VR2Gather: A collaborative social VR system for adaptive multi-party real-time communication

Irene Viola, Jack Jansen, Shishir Subramanyam, and Ignacio Reimat
Centrum Wiskunde en Informatica, The Netherlands

Pablo Cesar
Centrum Wiskunde en Informatica and TU Delft, The Netherlands

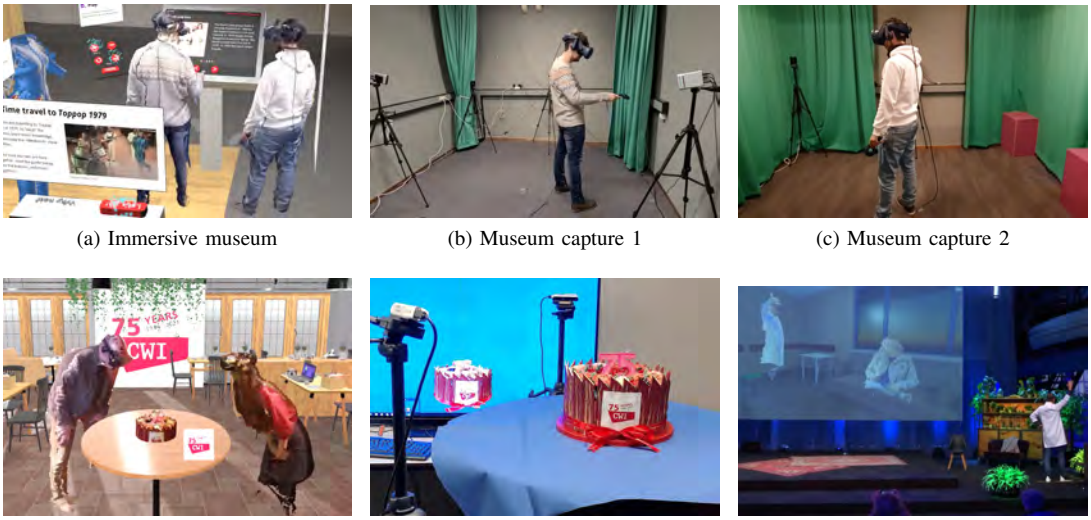
***Abstract*—Virtual Reality telecommunication systems promise to overcome the limitations of current real-time teleconferencing solutions, by enabling a better sense of immersion and fostering more natural interpersonal interactions. Many solutions that currently enable immersive teleconferencing employ synthetic avatars to represent their users. However, photorealistic reconstructions have been shown to increase the sense of presence with respect to synthetic avatars in tele-immersive scenarios. In this paper, we present VR2Gather, a costumizable end-to-end system to transmit volumetric contents in multi-party real-time communication. We present the architecture and evaluate the costs and benefits of using different modules and transport mechanisms in terms of CPU usage, latency, and bandwidth. Moreover, we report the user experience based on applications the system has been used for, and how it was customised to meet the requirements using different acquisition and rendering modules.**

THE FUTURE of media communication is immersive. Virtual Reality (VR) technologies have the potential to overcome the limitations of current telepresence systems by offering enhanced realism, better sense of presence, higher degree of interactivity, and more naturalness in remote social communication. Several commercial solutions are currently available for immersive teleconferencing, including Mozilla Hubs[1] and Meta's Horizon Worlds[2]. They all employ synthetic avatars to represent their users. However, recent studies show that a photorealistic reconstruction of users in tele-immersive scenarios allows participants to sense emotions and communicate effortlessly while opening new commercial opportunities [1]. Still, further work is needed to provide customizable solutions that enable different use cases (entertainment, cultural heritage, healthcare...), while meeting the demanding performance requirements for collaboration and communication. This is exactly the objective of VR2Gather, the contribution of this article.

A promising technology for capturing photorealistic representations of the users is using volumetric media [2]. However, such media contents require large amounts of data in order to faithfully represent digital humans, several orders

[1] https://hubs.mozilla.com/
[2] https://www.meta.com/horizon-worlds/

 Published by the IEEE Computer Society



(a) Immersive museum (b) Museum capture 1 (c) Museum capture 2

(d) Shared cake celebration (e) Cake capture (f) Remote consultation on stage

Figure 1: Examples of use cases developed with VR2Gather.

of magnitude bigger than traditional images and videos [3]. For example, an uncompressed 30 frames per second (fps) point cloud video with one million points requires around 5 Gigabit per second (Gbps). This may be attainable within a single location with current hardware using direct fiber-optic connections, but for remote communication we need to lower the bandwidth requirements by orders of magnitude. This can be done by lowering the frame rate or the point count, or through using compression techniques, or a combination of these. This inevitably leads to trade-offs between visual quality, end-to-end latency, and resource consumption.

An overview of volumetric video streaming approaches and systems can be found in [9]. We provide a comparison of existing tele-immersion systems in Table 1. A few systems provide support for real-time communication with varying degrees of adaptation and multi-user support. However, to the best of our knowledge, none of the existing systems supports and natively implements different transport protocols, which is a key aspect of our system, and what makes it adaptable to different scenarios. Some examples include systems based on the transmission of RGBD images [1], mesh-based reconstruction of humans using multiple Kinect cameras for gaming [4], Free Viewpoint Video (FVV) navigation [7], and heavyweight solutions using up to eight cameras with stereo depth estimation to capture users [5]. Recently, Jansen et al. engineered a pipeline for multiparty transmission of photorealistic volumetric contents over low-latency Dynamic Adaptive Streaming over HTTP (DASH), demonstrating its feasibility in enabling real-time VR communication [8]. The work allowed for up to four participants to meet in a shared virtual space. VR2Gather extends this work by providing a customizable approach in terms of capturing, transmission and rendering for volumetric video conferencing based on point clouds.

The first step in optimizing volumetric video streaming is through efficient compression solutions that can operate in real time; despite the emergence of new standards for point cloud compression [10], no standardized codec has yet real time capabilities. Due to the high demands of volumetric videos, recent works have explored a number of optimization techniques for both offline and real-time transmission of volumetric video. Relevant examples include the proposal of a Multipoint Control Unit (MCU) for real-time delivery of point cloud streams [11], viewport prediction to facilitate the delivery of the most relevant upcoming packets [6], and cloud or edge rendering [12]. VR2Gather uses and extends the Motion Picture Expert Group (MPEG) anchor codec [13] to compress in real time tiled streams

Table 1: Overview of current state-of-the-art tele-immersion systems

System	3D format	Acquisition	Transport	Adaptation	Multi-user	Open source
Zioulis2016 [4]	Mesh	Kinect v2	RabbitMQ	Network	Yes	No
Orts-Escolano2016 [5]	Mesh	NIR depth	Direct TCP	No	No	No
Son2020 [6]	Mesh	Photogrammetry	WebRTC	Motion prediction	-	No
Carballeira2021 [7]	FVV	Stereo cameras	RTP - UDP	User Network	Future work	No
Gunkel2021 [1]	RGB-D	Kinect v2 RealSense	WebRTC	Network	Yes	No
Jansen2020 [8]	Point cloud	Azure Kinect RealSense	DASH	Network	Yes	No
VR2Gather	Point cloud	RealSense Azure Kinect Samsung S20 Ultra 5G	Direct TCP SocketIO DASH	User Network	Yes	Yes

of point clouds at different quality metrics and provides three transmission mechanisms: HTTP adaptive streaming (HAS) using low latency chunked HTTP with fragmented ISOBMFF/MP4 [8] for alleviating the incurred latency, SocketIO using the WebSocket protocol, and a direct TCP connection for single-quality peer-to-peer transmission. It provides as well an implementation for tiling and multi-quality user adaptive streaming (see [14] for details).

VR2Gather is comprised of different modules to capture, encode, transmit, and rendering volumetric videos in real time. These modules can be switched to enable different 3D capturing devices and camera configurations, network protocols, and rendering environments. The contributions of the paper can be summarised as follows:

- We present an end-to-end system for volumetric real-time communication and interaction that can enable social VR experiences. The system is customizable, so that different capturing setups, transport protocols, and rendering applications can be selected. The system is open source and available here: https://github.com/cwi-dis/VR2Gather
- We analyse the trade-offs in using different transport protocols in terms of resource consumption, framerate, and latency. The measurements can serve as guidelines to enable different modules in the system, to ensure the best performance under given constraints.
- We demonstrate several use cases for VR2Gather, reporting the user experience and highlighting how different configurations can meet the specific requirements of each of them.

SYSTEM OVERVIEW

This section provides the architecture of VR2Gather, describing the modules and detailing the APIs between them that allow for different configurations as required by different use cases. The intention is that the selection of modules is not done at run time but at design time, matching performance and capabilities of the modules with the use case at hand.

Requirements

The main requirement driving the design of the pipeline is that it should be practically usable in a variety of live, real-time, settings. This leads to a requirement that the pipeline can be integrated in other software: unlike a videoconferencing pipeline, a tele-immersion pipeline will always have to be embedded into a bigger VR application, with other components implementing the virtual world aspects, user navigation, interaction with VR objects and more.

To cater for different types of VR applications the pipeline must be usable with different capturing devices, different user topologies (2 user, N-user symmetric, broadcast-like), different network setups (dedicated LAN, high-bandwidth internet, home internet via router from provider) and more modalities than only volumetric video and spatial audio (haptics, motion capture, gaze capture). This leads to a requirement that the pipeline is designed in a *modular* fashion, to enable selecting the right set of functionality for the application under study. It also leads to a requirement that the modules are *parameterisable* to allow adapting to the real-world constraints of the deployment scenario.

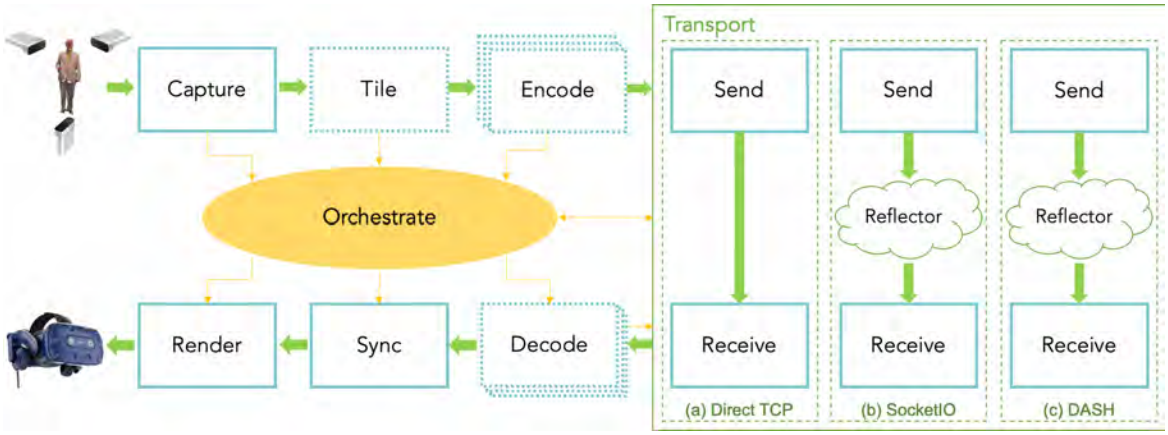


Figure 2: Functional architecture of the point cloud pipeline. Dotted lines denote modules that can be enabled or disabled, whereas the dashed lines denote modules that can be swapped.

A final requirement is that the pipeline is instrumented, and that the instrumentation data for all instances of the pipeline within a session can be combined for analysis.

System Architecture

The functional architecture of the system is depicted in Figure 2. Point clouds are captured, fused and time-stamped, (optionally) split into tiles, (optionally) compressed, transferred over the network, (optionally) decoded, synchronised and rendered. The figure depicts only the point cloud pipeline, a similar pipeline is used for conversational audio and there are optional pipelines for other modalities. Green arrows indicate the flow of point clouds, yellow arrows indicate control flow. Modules depicted with a dotted border are optional, and all modules depicted as boxes have an API which allows them to be replaced by other implementations. All data, especially point cloud data, is reference counted and uses shared pointers, to enable minimal copying and thereby decrease latency.

The capture module produces a stream of timestamped point clouds. The capture module is also responsible for aligning and fusing point clouds if they have been captured from multiple cameras. The optional tile module will split point clouds into tiles, and feed these streams of point cloud tiles into the optional encoder (or encoders), which in turn produce a stream of compressed timestamped data packets to feed the transport send modules. Each encoder can optionally produce multiple output streams for multiple quality levels.

For the transport module, we provide three implementations; however, it is obviously possible to add implementations for different protocols:

- The TCP transport modules allow efficient transfer of compressed or uncompressed point clouds over a direct TCP connection. It handles single-sender-single-receiver only, and no NAT traversal (Network Address Translation, as used by most home routers). It does handle multiple tile streams and selection of quality levels.
- The SocketIO transport modules use the central component of the orchestrate module to reflect media streams to other participants. It handles multiple receivers and NAT traversal, but all streams are delivered to all receivers so it is not suitable for multiple quality levels.
- The DASH transport modules implement low-latency DASH by uploading fragments and the accompanying DASH Media Presentation Description (MPD) to an ingestion server. The ingestion server can optionally use a CDN (content distribution network) to increase fan-out, or function as delivery server itself. The DASH receive module periodically retrieves the MPD, and selects the streams corresponding to the required tiles and quality levels and retrieves the fragments. Low latency DASH uses chunked HTTP transfer and ISOBMFF/MP4 fragmenting to allow using DASH for use cases

that are traditionally implemented using other protocols (e.g., WebRTC, RTP).

Adding other protocols, such as WebRTC, should not be too difficult provided that an implementation is available that is suitable for use from C# in Unity.

The receive modules feed compressed timestamped data packets to the (per tile stream) decode modules for decompression. The synchronise module will synchronise all of these streams across all modalities, and enqueue the point cloud tiles for synchronous rendering. The render modules will now synchronously render the individual point cloud tiles.

The orchestrate module handles communication of metadata between modules within one instance of the system and between the instances in a session.

Implementation

VR2Gather is implemented as a framework Unity application that implements the full VR experience for each user, plus some small centralised cloud-based components to handle orchestration, session management and media stream fan-out. To this framework Unity modules, scripts and scenes can be added to create the full VR experience. Figure 3 shows the layered software architecture. Modules depicted on a yellow background are available as open source; for the modules with a white background, we are considering options for making them available or providing alternatives.

Most modules are implemented in C#, some in C and C++. Implementation of optional modules is done by providing a zero-cost pass-through version of the module. All modules are portable to Windows, MacOS and Linux. The full application structure is governed by the Unity scene and object graph, with the orchestrate module providing overall control, and coordination and communication between the different instances of the VR2Gather application within the session. The render module is implemented in the standard Unity way, consisting of a shader implemented on the GPU and a control module in C#.

The capture, tile, encode and decode modules are implemented in C and C++, with the API usable from C# (in addition to other languages). This is done for reasons of efficiency (encoder and decoder) as well as to allow interfacing to vendor-specific hardware drivers (capturer). These modules are available as the *cwipc* software suite and are more fully described in [15]. All these modules share an efficient in-core point cloud format. Capture modules are available for Microsoft Azure Kinect and Intel Realsense cameras as well as for various disk-based formats. Cwipc provides an encoder and decoder for the MPEG Anchor codec [13].

The DASH transport modules and reflector are implemented in C++ and supplied by Motion Spell[3]. They are based on the GPAC suite, extended to allow transmission of compressed point cloud data over low-latency DASH.

PERFORMANCE COMPARISON

This section reports a number of experiments, under different conditions, demonstrating how VR2Gather can be used for real-time communication and collaboration. The data here can be used to help in the decision whether a specific transport mechanism is suitable for a specific use case. These experiments should be seen as a first step in performance evaluation; things like bandwidth fluctuation or the effect of fan-out and CDN usage on latency are not taken into consideration.

Experimental design and methodology

As described in the previous section, VR2Gather allows several transport mechanisms to be chosen. Depending on the transport mechanisms, several modules can be enabled: the point cloud content can be partitioned into tiles to allow for parallel encoding and decoding; in case of sufficient bandwidth, the encode and decode module can be skipped to reduce the latency; multi-quality encoding can be performed to allow for adaptation at the receiver's side. The goal of our experiment is to understand how each of the modules contributes to resource consumption. To do so, we performed two sets of experiments: a unidirectional transmission with pre-recorded point clouds, to ensure that the conditions are identical for all pipeline variants and that the results can be easily reproduced, and a bidirectional transmission with real-time

[3]https://www.motionspell.com

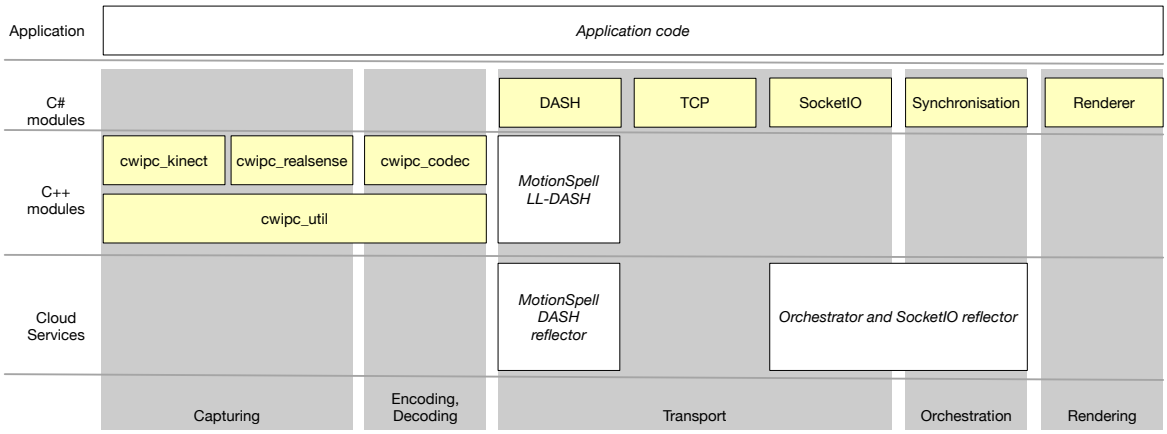


Figure 3: Software architecture of VR2Gather applications

capturing of two human subjects, to show the performance of our system in a live setting.

Three transport protocols were selected to be evaluated in the unidirectional experiment, referred to as Direct TCP, SocketIO, and DASH. As baseline (TCP.0), we consider a variant of the VR2Gather system with disabled tiling, encoding and decoding, using Direct TCP as the protocol, which allows us to compute the performance of the system in ideal conditions with high bandwidth. We subsequently enable encoding (TCP.1) and tiling (TCP.2) to progressively monitor the overhead added by the modules in terms of latency and CPU. As Direct TCP only allows for a point-to-point connection between two users, it is not suitable for multi-party communication; thus, we test SocketIO and DASH with encoding and tiling enabled (SocketIO.2 and DASH.2) to evaluate the costs of switching transport protocol. Then, we investigate the impact of enabling the multi-quality module in terms of latency, CPU usage, and bandwidth for both sender and receiver. As SocketIO does not allow for selecting which packets to receive, we instrument the multi-quality module in DASH. We select two encoding qualities, and test the performance when the high or low quality tiles are selected by the receiver (DASH.3 and DASH.4, respectively). For the bidirectional live transmission, we compare the baseline liveTCP.0 with respect to liveDASH.2.

For each condition under test, we measure the latency, the CPU usage at the sender and receiver side as relative to the baseline TCP.0, and the required bandwidth at the sender and receiver. We are not reporting GPU and memory use: the GPU is not used except for rendering and memory use is fairly constant.

Experimental setup

For the unidirectional experiment, we select the *Loot* sequence from the 8iVFB v2 dataset [3], pre-scaled with a voxel size of 2.5 to reach a point count of $150,000$ points, tiled statically into four tiles following the method described in [16], and played back looping from an SSD at $15\,\mathrm{fps}$. The point count and framerate were selected as they give an acceptable visual and temporal quality while maintaining a low encoding latency; moreover, they sufficiently resemble the values we can achieve in our live capture setup with 3 cameras, which ensures consistency among the measurements. Such values have been validated in terms of QoE in previous work [14]. To comply with the real time requirements, we selected the MPEG Anchor [13] to encode the point cloud content, as the encoding for the target point count could be completed in around $200\,\mathrm{ms}$. For the high quality encoding, an octree bit value of 9 was selected, leading to roughly the same point count, whereas for the low quality encoding, the octree bit was set to 7, which led to a point count of $50,000$ points. Despite its greater encoding efficiency, we could not use the current MPEG standard for dynamic point clouds V-PCC, as the required encoding time was $30\,\mathrm{s}$ for a point cloud of that size.

Additionally, we perform two fully interactive measurements, where two human subjects were each wearing a head mounted display (HMD),

and were engaging in a conversation while their point clouds were captured using a 3-camera setup. Audio was captured via the HMD microphone. The users could move freely in a space of about $5\,\mathrm{m}^2$ each and were virtually about $4\,\mathrm{m}$ apart. The virtual world was created in Unity, and consisted of a space with some minimal furniture and objects. Interaction and teleportation were possible but not used during the experiments.

The machines used were two Windows 10 Enterprise gaming PCs. The sender (vrbig) is a 4-core i7-770K running at 4.2GHz, 32GB memory, NVidia GeForce GTX 1080 Ti graphics card, whereas the receiver (vrsmall) is a 12-core i9-7920X running at 2.9GHz, 128GB memory, dual NVidia GeForce GTX 1080 Ti graphics cards. These machines were selected because, while serious enough, they are definitely not top of the line by today's standards and therefore the measurement setups give an indication of what can be done with commonly available hardware. The machines were directly connected with a dedicated gigabit Ethernet link for the TCP.0 and liveTCP.0 measurements, and with a normal shared Ethernet for internet access and for all other measurements. The DASH and SocketIO reflectors were running on another machine on the shared Ethernet. This second setup is comparable enough to a normal (internet and cloud-based) deployment that we feel confident to state that the latency results reflect what would be measured in a production setup (if we add the ping/RTT times between sender, reflector and receiver).

For the interactive experiments the machines were each fitted with a Vive Pro HMD rendering at 90Hz and 3 Microsoft Azure Kinect cameras. The cameras used the hardware synchronization cable, and point clouds were captured using the *cwipc_kinect* capturer, which builds upon the Azure Kinect Sensor SDK. Two users were recruited and instructed to strike a conversation with each other, as well as move around, to test the capabilities of the system. The session lasted for about a minute.

The pipelines are instrumented to record all relevant parameters to a log file, and system parameters such as CPU and memory use are recorded by a separate application. Correspondence of each system clock is also recorded, so after an experiment run has finished all measurements of the two computers can be combined and analysed.

Results

Because we have fixed the point count, the primary pipeline parameter of interest with respect to user-perceived quality is the *pipeline latency* between sender and receiver. Users are able to see themselves with a fairly fixed latency that depends on the cameras, capturer, renderer and HMD. This latency is around $350\,\mathrm{ms}$. As the hardware and software setup is identical for both users, they view the other participant with an additional *pipeline latency* added. Latencies were measured per frame, and any clock differences between sender and receiver were compensated for. Resource consumption measurements were taken once per second and also include contribution by background processes and such, and should therefore be seen more as indications than exact measurements. Table 2 reports numeric results for our experiments. They were collected over 60-second sessions after discarding the first and last 10 seconds, to get rid of the session startup and termination anomalies. CPU results are shown as relative to the baseline TPC.O scenario, to improve readability and comparison between the transport mechanisms. Figure 4 shows the pipeline latency and its contributing factors over a session, for selected experiments. This graph allows seeing how the latency contributions develop over time, how much jitter there is and how regular or irregular that jitter is.

One latency contribution that cannot be measured directly and must be deduced from the presented contributions is the "missing contribution": network transit time. We cannot measure this due to the way TCP/IP is implemented in Windows and other operating systems: a transmit system call may return immediately after storing the data in a system buffer, so before the data has been received by the other party. This makes it impossible to determine the upload time for the SocketIO and DASH protocols. We have determined independently that the total network transit time is indeed very similar to the missing latency contribution, using a WireShark session in parallel to an experimental run.

For uncompressed direct TCP Ethernet connection (TCP.0) the quality is obviously maximal,

protocol	encoding	no. tiles	multi-quality	rec. quality	latency (ms)	CPU (relative)		bandwidth (Mbps)	
						sender	*receiver*	*sender*	*receiver*
TCP.0	✗	1	✗	-	40.6 ± 2.0	1.00	1.00	281.8	281.4
TCP.1	✓	1	✗	-	278.7 ± 3.6	16.78	1.98	19.0	19.0
TCP.2	✓	4	✗	-	90.6 ± 5.9	12.04	2.47	20.3	20.3
SocketIO.2	✓	4	✗	-	109.0 ± 6.0	12.39	2.54	20.4	20.4
DASH.2	✓	4	✗	-	257.6 ± 35.3	12.27	2.92	20.4	20.5
DASH.3	✓	4	✓	high	273.2 ± 41.5	20.11	2.86	24.9	20.3
DASH.4	✓	4	✓	low	232.6 ± 24.2	20.35	2.14	24.9	4.8
liveTCP.0	✗	1	✗	-	87.4 ± 11.1	9.30	2.17	212.0	211.6
liveDASH.2	✓	7	✗	-	562.0 ± 67.1	30.50	5.71	16.4	16.7

Table 2: Pipeline performance in terms of latency (ms), relative CPU, and bandwidth (Mbps) for both sender and receiver side, for each of the protocol variants under test: TCP, SocketIO, and DASH, without encoding and tiling (.0), with encoding (.1), with tiling (.2), with multi-quality encoding and high quality selected from the receiver (.3), with multi-quality encoding and low quality selected from the receiver (.4).

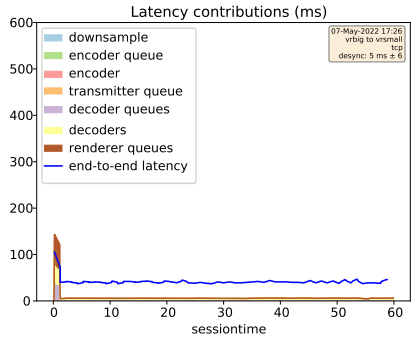
the latency minimal, but the bandwidth used is very high. A single uncompressed point cloud of $150,000$ points is $19.2\,\mathrm{Mbit}$, so it will spend about $20\,\mathrm{ms}$ being transmitted over a gigabit ethernet. This number is corroborated by the latency figure here and the graph in Subfigure 4 (a).

When we compress the point clouds (TCP.1), we see a significant bandwidth reduction of more than 90%, but at the expense of serious increase in latency. As the packets spend about $2\,\mathrm{ms}$ on the network, we can deduce that it takes around $250\,\mathrm{ms}$ to compress and decompress a point cloud on the hardware we used. By introducing tiling (into 4 tiles, TCP.2) we are able to compress and decompress the partial point clouds in parallel. Latency drops, but not by a factor of four: the tiles have different numbers of points and for each point cloud the encoding and decoding latency of the largest tile will determine the overall latency.

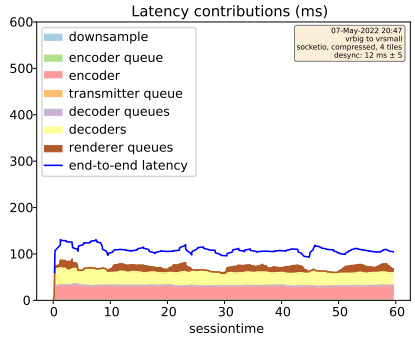
Introducing SocketIO as transport mechanism, in stead of direct TCP (SocketIO.2) obviates the need of a direct TCP connection, thereby allowing both sender and receiver to be behind a firewall or NAT router. This enables use over the internet at large, at the price of only $20\,\mathrm{ms}$ extra latency. Subfigure 4 (b) shows the latency contributions of each module. We can see the contribution of the encoder and decoder to the latency. All tile streams are transmitted synchronously over a single SocketIO connection, so the per-tile compressed point clouds arrive synchronously, but because each tile takes a different amount of runtime to be decoded, there is also a small contribution by the renderer queue, while the decompressed tiles wait for the last decompressor to finish.

Switching to the DASH transport protocol does increase the latency from 110 to $260\,\mathrm{ms}$ due to protocol overhead (DASH.2). There are no good numbers yet in the literature for acceptable latency for interactive 6DOF volumetric video, we plan to do this work in the future, but $260\,\mathrm{ms}$ is below the threshold acceptable for normal 2D video conferencing [17].

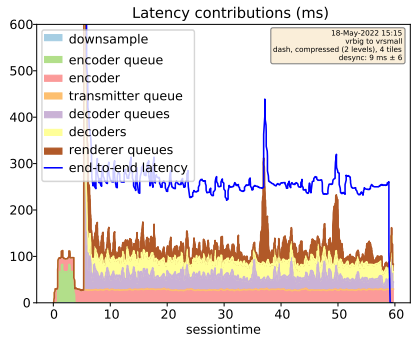
The next two lines show the advantage that can be gained by switching from SocketIO to DASH as transport mechanism: DASH allows uploading the point cloud stream in multiple quality levels (DASH.3 and DASH.4). The receiver can then select which quality level to download and decode, allowing an application to cater for lower download bandwidth, and lower decoder CPU use. This is especially useful for applications with more than 2 participants: tiles that are not in the field of view of the receiver can be downloaded at low quality. This is true as well for participants that are completely outside the viewport as for some of the tiles of participants that are in view: the tiles that are occluded by other tiles. Latency contribution for DASH.3 are shown in Subfigure 4 (c). The first observation that can be made is that the DASH pipeline is not stable during the first 10 seconds of the session. This is due to the DASH MPD file not being available until the first full fragment has been uploaded for all streams, and then the receiver taking some



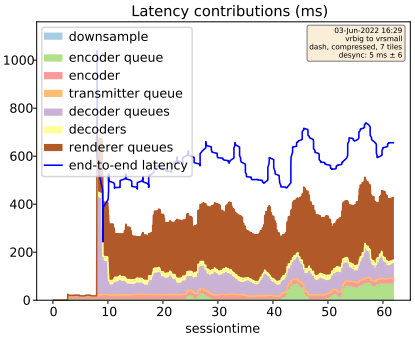
(a) TCP.0



(b) SocketIO.2



(c) DASH.3



(d) liveDASH.2

Figure 4: Latency contributions for several transport mechanism configurations.

more time to catch up. The second observation is that for DASH the decoder queue duration is a significant contribution to the total latency. This is due to the individual DASH streams being independent, unlike for SocketIO, so the arrival of compressed packets is not synchronised. Therefore, in addition to the renderer queue latency there is also a decoder queue latency. The spike at $t = 37$ is due to some random network event, but we can see that the system catches up quickly. The synchroniser is parameterised, so the strategy for catching up or dropping frames and giving priority to audio can be controlled by the application.

The last two rows of Table 2 show the performance of the pipeline for bidirectional live sessions. It can be observed that the latency measurements increased by roughly a factor of two. Using DASH leads to larger latency values with respect to direct connection by roughly a factor of 6, and to a larger CPU usage, especially at the sender side. DASH enables both tiling and encoding, which naturally lead to added latency, as shown in the pre-recorded experiment. An overview of the latency contributions can be seen in Subfigure 4 (d). Jitter and renderer queue latency are quite a bit higher than for the prerecorded case. This is due in part to the live participants walking around and such, which results in the tiles having unbalanced point counts. This in turn leads to some tile streams taking more time in the encoder and decoder, which in turn makes the other tile streams wait in the renderer queue, because the synchroniser enforces synchronised playout of the tiles. The latency is right at the edge of what [17] considered acceptable for 2D video. The benefits of using DASH with respect to TCP can be seen in terms of bandwidth expenditure: whereas the baseline necessitates 212Mbps in order to enable the communication, with DASH we are able to reduce it by a factor of 13, to 16Mbps.

USE CASES

This section describes different use cases, demonstrating the possibilities opened by the VR2Gather platform. It showcases three case studies: an immersive museum experience, a shared cake celebration, and a remote doctor consultation. The section demonstrates the customization capabilities for realizing each of the

use cases and reports the achieved user experience.

The first experience is around cultural heritage[4]. Current museum experiences offer very limited, if not null, interaction with the artifacts on display. Moreover, space constraints and concerns over the fragility of the artifacts often mean that several pieces of the collections are not accessible to the visitors. The VR2Gather platform can be used to make such pieces virtually accessible to the users. Furthermore, new experiences can be devised so to allow interactivity that would not be possible in a physical setting: think of pulling apart complex objects to see each individual part in detail, or wearing historical costumes that are usually behind a glass. The latter was the focus of our demonstrator, in which a costume from the collection, historically worn for a pop performance, was presented to the museum visitors. Each user could interact with the costume and wear it while recreating the historical performance, for example by playing instruments and singing on stage. Figure 1 (a) shows two users interacting with a virtual information panel in the museum, and (b) and (c) show that same moment in the real world.

The second experience is around connecting with others while being apart[5]. Geographic remoteness, travel limitations, social distancing, and carbon footprint concerns all can lead to being unable to celebrate life events with loved ones. This was the case for the 75 anniversary national research institute for mathematics and computer science in the Netherlands, which, while supposed to be happening in person, was forced to be held online. We successfully demonstrated how our VR2Gather platform could be used to bring people together and share a virtual cake slice. Our setup included a specific capturing system for the cake itself along with the users, which were able to chat and interact in real time while being located in different cities. Figure 1 (d) and (e) show the setup in operation.

The last experience is around remote consultation with doctors[6]. The possibility of meeting remotely in an immersive environment opens new possibilities for healthcare. Being able to have remote consultations reduces the amount of time patients spend travelling to the clinic and waiting for their appointment, which might be additionally difficult or impossible for people with mobile impairments. Moreover, it can allow the patient to receive urgent care advice in real time while waiting for the healthcare personnel to be dispatched on site. For remote consultation, be it with general practitioners or mental health specialists, it is particularly important to have a faithful reconstruction of the patient with respect to a synthesised avatar. However, patients might not have immediate access to volumetric capturing hardware or stable internet connections. Figure 1 (f) shows the ”doctor” participating in the session on-stage while a third person viewpoint of the virtual world is projected for the benefit of the conference audience.

Customization of the Modules

A key contribution of this article is the customization capabilities of VR2Gather. Each of the experiences described above used the same platform (see the System Overview section), but using different modules and configurations for meeting the requirements.

The immersive museum experience was offered *in situ*, so that museum visitors can explore immersive narratives together: in that case, a physical connection can be devised in order to offer the best possible visual quality and lowest latency. As users can navigate freely in the virtual space and see themselves along with the other visitors, it is recommended to use a multi-camera setup. Furthermore, the VR2Gather platform can also be used to connect users remotely, so that friends can experience an afternoon at the museum while being in different geographic locations, offering enhanced social experiences. In this case, depending on the available camera setup for each user, a (multi-tiled) multi-quality approach would be the best to ensure high framerate and low bandwidth expenditure at the receiver side.

[4]https://www.dis.cwi.nl/news/2021-12-10-cwi-and-the-netherlands-institute-of-sound-and-vision-gave-a-sneak-peek-into-the-future-of-cultural-heritage/

[5]https://cacm.acm.org/news/256173-the-outlook-for-virtual-meetups/fulltext

[6]https://www.dis.cwi.nl/news/2020-11-10-dis-group-realises-worlds-first-volumetric-video-conference-over-public-5g-network/

The cakeVR experience instead used three volumetric participants captured with two different multi-camera hardware setups: for the cake, the Intel RealSense cameras were used, as their sensors allowed to capture non-corporeal data such as the flame on the birthday candle, whereas for the attendees, the Azure Kinect cameras were used as they offer better visual reconstruction quality. To ensure a stable connection with high quality and low latency, the transport configuration was set with a single-quality tiled stream using SocketIO. If download bandwidth is a concern, the multi-quality setup with DASH can be enabled instead.

For the last experience, remote consultation, a consumer-grade mobile phone was used to transmit a volumetric representation of the user using a public 5G network. To limit the bandwidth consumption, the visual communication was kept uni-directional, whereas the audio connection was bi-directional; that is, the doctor was able to see and hear the patient, whereas the patient could only hear the doctor's instructions, but not see them. The demonstrator represented the world's first volumetric video conference over public 5G network. The bandwidth requirements might be the most stringent in this use case. If upload bandwidth is limited, then a single-quality scenario would be the most advantageous. Enabling tiling would be needed to decrease the latency in case of a multi-camera setup, whereas for a single-camera case such as the mobile phone demonstrator, it is not necessary. On the contrary, if download bandwidth poses more constraints, a multi-quality solution might be more suitable.

User Experience

The three experiences have been deployed in the real world, with actual users. The reader is invited to self-assess the user experience provided by each of them watching the following videos:

- Museum experience with two users per session https://youtu.be/I7kY1cMZyD0
- Cake VR experience with two users and a cake per session https://youtu.be/KcRpp0s50RQ
- Remote Consultation with two users per session https://youtu.be/EuoRfy18_Fc

The first experience was demonstrated in collaboration with the Netherlands Institute for Sound and Vision, during VRDays 2021. The experience attracted more than 100 visitors. During the event, subjective data in the form of Net Promoter Score (NPS) [18] and System Usability Scale (SUS) [19] scores were collected, in order to understand user satisfaction and system usability. A total of 84 questionnaires were collected in total. Out of the 84 recorded participants, 3 were below 18 years old, 50 between 18 and 35, 23 between 36 and 50, while 8 users were above 50 years of age. From these 84 participants, 4 did not fill in the requested information for the computation of the SUS, while 3 participants omitted at least 1 question. Thus, the computation of the SUS relies on the remaining 77 users.

Among the 84 participants, 28.6% (24 out of 84) of them were promoters, 64.3% (54 out of 84) of them were passives, and 7.1% (6 out of 84) of them were detractors. Thus, the final NPS score of MediaScape XR is 28.6 - 7.1 = 21.5. According to Jeff Sauro's study, the NPS score of digital products ranged from -26 to 40, with an average of 15 [20]. Thus, it shows that the NPS score of the MediaScape XR application is above the benchmark, indicating a satisfying user experience. The SUS scores given by 77 participants is (M=75.3, SD=13.4). As validated by Bangor et al., the SUS score no fewer than 50.9 is considered as "ok", no fewer than 71.4 is "good", no fewer than 85.5 is "excellent", and no fewer than 90.9 is "best imaginable" [19]. From the results, 5 out of 77 participants (6.4%) rated it as "best imaginable", 9 out of 77 participants (11.7%) rated it as "excellent", 37 out of 77 participants (48.1%) rated it as "good", 22 out of 77 participants (28.6%) rated it as "ok", 4 out of 77 participants (5.2%) rated it as below "ok". The average SUS score of MediaScape XR across all users is 75.3, which falls under the category of "good".

CONCLUSION

In this paper, we presented VR2Gather, a system for volumetric real-time communication. We presented its modular architecture, and showcased the costs and benefits of employing different transport mechanisms to send and receive point cloud information. We described several use cases for the proposed systems, and discussed future directions.

ACKNOWLEDGEMENT

This work was supported through "PPS-programmatoeslag TKI" Fund of the Dutch Ministry of Economic Affairs and Climate Policy and CLICKNL, the European Commission H2020 program, under the grant agreement 762111, *VR-Together*, http://vrtogether.eu/, and the European Commission Horizon Europe program, under the grant agreement 101070109, *TRANSMIXR*, , https://transmixr.eu/. Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or Directorate-General for Communications Networks, Content and Technology. Neither the European Union nor the granting authority can be held responsible for them.

REFERENCES

1. S. N. B. Gunkel *et al.*, "VRComm: an end-to-end web system for real-time photorealistic social VR communication," in *Proceedings of the 12th ACM Multimedia Systems Conference*. Istanbul Turkey: ACM, Jul. 2021, pp. 65–79. [Online]. Available: https://dl.acm.org/doi/10.1145/3458305.3459595
2. J. v. d. Hooft, M. T. Vega, T. Wauters, C. Timmerer, A. C. Begen, F. D. Turck, and R. Schatz, "From Capturing to Rendering: Volumetric Media Delivery with Six Degrees of Freedom," *IEEE Communications Magazine*, vol. 58, no. 10, pp. 49–55, 2020. [Online]. Available: 10.1109/MCOM.001.2000242
3. E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i Voxelized Full Bodies - A Voxelized Point Cloud Dataset," *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006, Geneva, CH*, January 2017.
4. N. Zioulis, D. Alexiadis, A. Doumanoglou, G. Louizis, K. Apostolakis, D. Zarpalas, and P. Daras, "3D tele-immersion platform for interactive immersive experiences between remote users," in *2016 IEEE International Conference on Image Processing (ICIP)*. Phoenix, AZ, USA: IEEE, Sep. 2016, pp. 365–369. [Online]. Available: http://ieeexplore.ieee.org/document/7532380/
5. S. Orts-Escolano *et al.*, "Holoportation: Virtual 3D Teleportation in Real-time," in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, ser. UIST '16. New York, NY, USA: Association for Computing Machinery, Oct. 2016, pp. 741–754. [Online]. Available: https://doi.org/10.1145/2984511.2984517
6. J. Son, S. Gül, G. S. Bhullar, G. Hege, W. Morgenstern, A. Hilsmann, T. Ebner, S. Bliedung, P. Eisert, T. Schierl, T. Buchholz, and C. Hellge, "Split Rendering for Mixed Reality: Interactive Volumetric Video in Action," in *SIGGRAPH Asia 2020 XR*. Virtual Event Republic of Korea: ACM, Dec. 2020, pp. 1–3. [Online]. Available: 10.1145/3415256.3421491
7. P. Carballeira, C. Carmona, C. Díaz, D. Berjón, D. Corregidor, J. Cabrera, F. Morán, C. Doblado, S. Arnaldo, M. del Mar Martín *et al.*, "Fvv live: A real-time free-viewpoint video system with consumer electronics hardware," *IEEE Transactions on Multimedia*, vol. 24, pp. 2378–2391, 2021. [Online]. Available: 10.1109/TMM.2021.3079711
8. J. Jansen, S. Subramanyam, R. Bouqueau, G. Cernigliaro, M. M. Cabré, F. Pérez, and P. Cesar, "A pipeline for multiparty volumetric video conferencing: transmission of point clouds over low latency DASH," in *Proceedings of the 11th ACM Multimedia Systems Conference*. Istanbul Turkey: ACM, May 2020, pp. 341–344. [Online]. Available: https://dl.acm.org/doi/10.1145/3339825.3393578
9. I. Viola and P. Cesar, "Volumetric video streaming: Current approaches and implementations," in *Immersive media technologies*, G. Valenzise, M. Alain, E. Zerman, and C. Ozcinar, Eds. Elsevier, 2022.
10. S. Schwarz *et al.*, "Emerging MPEG Standards for Point Cloud Compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133–148, 2019. [Online]. Available: 10.1109/JETCAS.2018.2885981
11. G. Cernigliaro, M. Martos, M. Montagud, A. Ansari, and S. Fernandez, "PC-MCU: point cloud multipoint control unit for multi-user holoconferencing systems," in *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV '20. New York, NY, USA: Association for Computing Machinery, Jun. 2020, pp. 47–53. [Online]. Available: https://doi.org/10.1145/3386290.3396936
12. D. Zhang, B. Han, P. Pathak, and H. Wang, "Innovating Multi-user Volumetric Video Streaming through Cross-layer Design," in *Proceedings of the Twentieth ACM Workshop on Hot Topics in Networks*. Virtual Event United Kingdom: ACM, Nov. 2021, pp. 16–22. [Online]. Available: https://dl.acm.org/doi/10.1145/3484266.3487396
13. R. Mekuria, K. Blom, and P. Cesar, "Design,

Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 4, pp. 828–842, 2017. [Online]. Available: 10.1109/TCSVT.2016.2543039

14. S. Subramanyam, I. Viola, J. Jansen, E. Alexiou, A. Hanjalic, and P. Cesar, "Evaluating the impact of tiled user-adaptive real-time point cloud streaming on vr remote communication," in *Proceedings of the 30th ACM International Conference on Multimedia*, ser. MM '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 3094–3103. [Online]. Available: https://doi.org/10.1145/3503161.3548220
15. I. Reimat, E. Alexiou, J. Jansen, I. Viola, S. Subramanyam, and P. Cesar, *CWIPC-SXR: Point Cloud Dynamic Human Dataset for Social XR*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 300–306. [Online]. Available: https://doi.org/10.1145/3458305.3478452
16. S. Subramanyam, I. Viola, A. Hanjalic, and P. Cesar, "User Centered Adaptive Streaming of Dynamic Point Clouds with Low Complexity Tiling," in *Proceedings of the 28th ACM International Conference on Multimedia*. Seattle WA USA: ACM, Oct. 2020, pp. 3669–3677. [Online]. Available: https://dl.acm.org/doi/10.1145/3394171.3413535
17. J. Tam, E. Carter, S. Kiesler, and J. Hodgins, "Video increases the perception of naturalness during remote interactions with latency," *CHI EA*, pp. 2045–2050, 1 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2223656.2223750
18. N. I. Fisher and R. E. Kordupleski, "Good and bad market research: A critical review of net promoter score," *Applied Stochastic Models in Business and Industry*, vol. 35, no. 1, pp. 138–151, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/asmb.2417
19. A. Bangor, P. Kortum, and J. Miller, "Determining what individual sus scores mean: Adding an adjective rating scale," *J. Usability Studies*, vol. 4, no. 3, p. 114–123, may 2009. [Online]. Available: 10.5555/2835587.2835589
20. J. Sauro, "The challenges and opportunities of measuring the user experience," *J. Usability Studies*, vol. 12, no. 1, p. 1–7, nov 2016. [Online]. Available: 10.5555/3040226.3040227

Irene Viola is a tenure-track researcher at Centrum Wiskunde & Informatica (CWI). Contact her at irene@cwi.nl

Jack Jansen is a researcher at Centrum Wiskunde & Informatica (CWI), Contact him at Jack.Jansen@cwi.nl.

Shishir Subramanyam is currently a PhD candidate at Centrum Wiskunde & Informatica with the Distributed & Interactive Systems Group. Contact him at subraman@cwi.nl.

Ignacio Reimat is currently a R&D Engineer at Centrum Wiskunde & Informatica (CWI) in Amsterdam. Contact him at nacho@cwi.nl.

Pablo Cesar leads the Distributed and Interactive Systems Group, Centrum Wiskunde & Infomatica (CWI) and is Full Professor with TU Delft, The Netherlands. https://www.pablocesar.me