Optimizing CDNs for VR Streaming - CS 538 Project Proposal

Jeff Byju, Aryan Gupta, Vinit Gupta, Siddharth Jain, Yahav Mangel

1 Project Overview

We propose a *single-connection, multi-response* transport layer protocol that lets a client open one logical UDP/UDT session to a "primary" edge server (e.g., a CDN node), while a pool of secondary servers cooperatively transmit the payload as if it all originated from that primary. The target workload is AR/VR-class interactivity where motion-to-photon (MTP) latency should be kept near or below $\sim$20ms for user comfort. This design keeps a single app-visible flow while amortizing server egress load across multiple machines and preserving one server IP/port at the client. In this proposed architecture, Virtual Reality (VR) live streaming will be implemented using ROS (Robot Operating System) Bag files as the data source.

2 Motivation

- **AR/VR streaming** stresses both bandwidth and tail latency. Prior work has improved VR streaming at the application (Loka) [1], perceptual/bandwidth (Stanford) [7], and CDN distribution levels (Blazing CDN) [6], but none address cooperative multi-server transport under a single session which is the core idea our project targets.
- **TCP's head-of-line blocking and strict in-order semantics** can inflate recovery latency under loss and jitter on high-BDP paths. The *UDT* (UDP-based Data Transfer) protocol [3] bypasses HOL blocking by providing reliability via selective NAK-based loss recovery, periodic ACK/ACK2 control, and a hybrid rate+window congestion control tuned for high-BDP networks, making it a better fit for interactive high-throughput flows. However, UDT was originally designed for bulk file transfer and lacks native support for frame deadlines, which are critical for real time streaming.
- **Clients benefit from the simplicity of a single logical flow, while providers benefit from parallel egress across multiple servers for throughput, resilience, and cost control**. Protocols such as Multi-Path TCP [12] apply this idea to general TCP flows, but do not attempt to use it for streaming. Applying this principle to CDN architectures would enable cooperative communication among CDN nodes, thereby allowing dynamic load balancing and more efficient distribution of VR streaming data.

3 System Architecture

3.1 Roles

- **Client:** Opens a single UDT session to a server (e.g., `frame_num:port`) and issues a `FRAME_REQ` per frame (or group of frames).
- **Primary (Coordinator):** Terminates the UDT handshake, assigns chunk indices for each frame, and distributes transmit grants to secondaries.
- **Secondaries (Workers):** Transmit assigned chunks directly to the client using a shared bus/port (multicast), transmitting information about the requested frame and session info. This preserves one logical transport from the client's perspective, yet allows multi-origin egress from cooperating servers.

3.2 Reliability and Coordination (UDT)

- **Reliability:** UDT adds sequencing, acknowledgments, and loss recovery on top of UDP.

- **Coordination:** The primary assigns non-overlapping chunk ranges; secondaries transmit those chunks; the primary aggregates feedback and triggers targeted repairs.
- **Single Flow View:** The client keeps one socket and one congestion-control context, even though multiple servers help send data.

  In our design, the client opens one UDT session to a service VIP while a primary coordinator assigns disjoint chunk ranges for each frame to secondary servers; from the client's view it remains a single flow. The primary aggregates ACK/NAK feedback and triggers targeted repairs, while secondaries transmit their assigned chunks directly.

3.3 Per-Frame Flow

1. Client sends a *frame request* over the existing UDT session.
2. Primary splits the frame into chunks and decides which servers will send which chunks.
3. Secondaries transmit their chunks directly to the client under the same VIP.
4. Client acknowledges or reports missing chunks; the primary coordinates any retransmissions.
5. Once enough chunks arrive (optionally with parity/FEC), the frame is reconstructed and rendered.

4 Implementation Plan

As an initial phase of the project, we will design and implement the core client-server architecture alongside a custom transport protocol based on UDT. The protocol will be developed entirely in user space to allow for rapid iteration and direct control over congestion management, reliability mechanisms, and flow coordination. To validate and test our implementation under realistic network conditions, we will utilize the Kathará network emulator. Kathará provides containerized, topology-aware virtual environments, enabling us to deploy the client and server components across isolated nodes and simulate controlled network scenarios. We will instrument the testbed with packet capture tools such as tcpdump and integrate Wireshark for real-time protocol tracing and telemetry analysis, and deploy the Kathará setup on a CloudLab VM.

5 Division of Work

We will divide work as follows:

1. **UDT-based Protocol Implementation** - Yahav, Vinit, Jeff, Siddharth
2. **Kathara Architecture Simulation and Testing** - Yahav, Vinit, Jeff, Aryan
3. **VR ROSbag Porting and Testing** - Aryan, Siddharth

References

[1] Loka: A Cross-Platform Virtual Reality Streaming Framework for the Metaverse (PMC, 2025)
[2] UDT Github Repository
[3] UDP-based Data Transfer Protocol (Wikipedia, 2006)
[4] AR and VR device shipments to skyrocket through 2026 (Emarketer, 2022)
[5] Easily record and store robotic application data with the S3 rosbag cloud extension (AWS Robotics Blog, 2020)
[6] How Video CDN Supports 360° and VR Streaming Experiences (BlazingCDN, 2025)
[7] Towards Retina-Quality VR Video Streaming: 15 ms Could Save Your Bandwidth (Stanford Research)

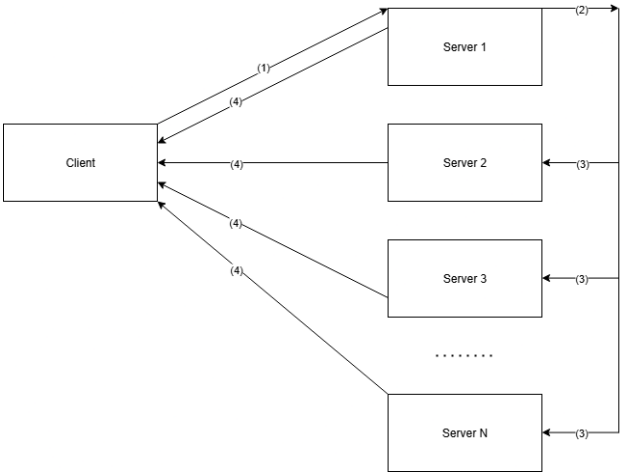

Figure 1: (1) Client initiates UDT request (2) Primary server shares request from step 1 on common bus (3) All involved servers (except primary) receive the client's request and prepare the data (4) All servers respond to the UDT request concurrently

[8] ros2/rosbag2 GitHub Repository (ROS2 Project, 2018)
[9] The Role of Edge Computing in OTT and Live Video Streaming (Dacast, 2025)
[10] VR Cloud Gaming UX: Exploring the Impact of Network Quality (arXiv, 2023)
[11] Content Delivery Networks (CDNs) and Live Streaming (WJARR, 2025)
[12] Design, implementation and evaluation of congestion control for multipath TCP (IEEE, 2011)