A User Friendly Local Area Network Controller (G04)

By: Adam McNeil and Nikita Machchenko

Abstract

In this paper, we propose to research how to communicate real-time variables over a local area network (LAN). The target application is to have a large number of devices communicating with a central server that are all on the same network. We will research technologies like WebRTC, RPCs, and raw UDP sockets. We focus on creating the best Quality of Experience (QoE) in the following areas:

1. Starting the application
2. Communicating in the network

Introduction

There would be many applications for user-friendly communication in a local network, such as interactive presentations at conferences or classrooms. The application would be simple enough that it could be run on any smartphone. Since nearly everybody has a smartphone, the application could be widely adapted and used in any setting with a local network. It could also replace outdated technologies like clickers used in college classrooms.

The problem that we are trying to solve is very similar to what popular FPS games do. One of the main assumptions that we are making is that all devices are connected to the local area network (LAN). This is the primary assumption that makes this project different from other communication of the network. This assumption greatly reduces the complexity of the problem. It has implications for both joining the system's network and communication in the network.

Starting the Application

In order to provide the best QoE the user must be able to quickly and simply use the application. This would entail two different areas:

1. Distributing the controller (as a web or native application)
2. Connecting to the network

The user should not deal with entering an IP address manually. Ideally the user code scans a code with their phone that would transfer the IP address of the server to the client. We may need to rely on a centralized server that the local server registers its IP address with. The client then could connect to the local server by communicating with the remote server.

Communicating

Because the controller will communicate over the LAN, it will decrease the latency of the application compared to an application that relies on a central server. All of the data will be displayed and processed on the central server. Because of this almost all of the data flow will be from the client to the server. The server will only occasionally send acknowledgments to the client for specific join and leave messages. Because we want inputs from the client to be processed as

quickly as possible we will send updates to the server as soon as they are made. We will not care if the server receives the update because the user will likely already have made an input that made the last input irrelevant. This may cause the server/network to be overloaded causing updates to be delayed. We would like to research the limitations of this type of communication. We would produce prototypes and then profile and compare the differences.

Related Works

The largest factor that would contribute to the quality of experience for our application would be a low-latency communication channel. The biggest challenge to this is the high amount of net work created by our application that is all direct to the same location. We hope to mitigate this problem by using a fast communication channel and communicating efficiently over the network.

UDP

UDP is an unreliable "hope for the best" form of communication used on the internet. To keep the throughput high it does not perform any congestion control [13]. Novel schemas for congestion control have been proposed [13] but we may see the best results if we simply use a naive approach of raw UDP socket. We would also be able to tune your application to use the communication channel efficiently as they did here [15]. Performance Adaptive UDP (PA-UDP) is designed for bulk data transfer in a local area network [14]. PA-UDP puts data transfer limits on applications to reduce the number of dropped packages [14]. Though it is not ideal to limit the transfer rate this may be necessary for a large number of clients.

WebRTC

The state-of-the-art for real-time transfer over the web is Web Real-Time Communication (WebRTC) [1, 9]. WebRTC is an open-source project that is maintained by Google [10, 9]. Large companies like Google, Facebook, and Discord use the protocol for messaging apps. WebRTC is designed for audio and video communication over the web. The protocol uses UDP because of the need for high throughput [2]. The protocol enables communication between browsers [2]. Because of this, it would make deployment of the application simple. The user would need to visit a website and connect to the server. The development would simply be making a webpage that creates a web socket to initiate communication However, previous research has shown it does not always provide the best Quality of Experience (QoE) [1]. This research was not specific to a LAN, so they needed to deal with other factors that our application will not have to deal with namely, FireWalls and NAT [1]. This would be important to keep in mind because we will create a large amount of traffic. Due to the large amount of traffic that our application would have, the WebRPC might become overwhelmed.

RPC

Remote procedure calls (RPC) are a way of calling functions on another computer. There are different types of RPC: [11]

- Synchronous. The client makes a call and waits for a reply.
- Non-blocking. The client makes a call but it continues with its processing.
- Batch-mode. A client sends multiple non-blocking calls in a group.
- Broadcast. A client sends a message to multiple servers and then receives all

the resulting replies.

- Callback. A client makes a non-blocking client-server call.

The RPC protocol can use TCP or UDP at the network layer making it ideal for our application. Because of this RPCs can be used to do low latency communication. RPCs were used in a data center in a paper called “Overload Control for µs-scale RPCs with Breakwater”. They demonstrated that under a normal load of RPC calls the latency could be as low as 20ms [3]. This latency would be sufficient to sustain real-time communication. There are also implementations of gRPC that can be used on the web [8]. This means that the controller could be deployed as a native app or on the web.

DNS

The domain name system is a system that the internet uses to translate readable domain names to IP addresses [7]. There is a process of registering a domain name with a trust organization that ensures that the name is unique making the domain name globally routable. This means that many applications rely on the DNS server to be trustworthy. We would be able to leverage this technology to allow the clients access to the network. We however do not need the domain name to be globally routable because applications will only be joining from within the local network.

Timeline

- March 7- Create simple applications to test the remote controller. This should be protocol agnostic.
- March 14 - Create script that connects server with WebRTC.
- March 21 - Profile and collect data on WebRTC remote.
- March 28 - Create script that connects to server with RPCs.
- April 4 - Profile and collect data on RPC remote.
- April 6 - Submit Midterm report.
- April 11 - Polish the deployment process for the above controllers.
- April 18 - Collect data with as many clients as possible.
- April 25 - Work on final report.
- May 11 - Submit final report.

References

[1] N. I. Ariffin, M. A. S. Hamdan and S. F. Kamarulzaman, "Internet of Things Intercommunication Using SocketIO and WebSocket with WebRTC in Local Area Network as Emergency Communication Devices," 2023 IEEE 8th International Conference On Software Engineering and Computer Systems (ICSECS), Penang, Malaysia, 2023, pp. 268-273, doi: 10.1109/ICSECS58457.2023.10256297.

[2] J. Cui and Z.-Y. Lin, "Research and Implementation of WebRTC Signaling via WebSocket-based for Real-time Multimedia Communications," Feb. 2016, doi: https://doi.org/10.2991/iccsae-15.2016.72.

[3] Cho, A. Saeed, J. Fried, S. J. Park, M. Alizadeh, and A. Belay, "Overload Control for µs-scale RPCs with Breakwater," in *Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, USENIX Association, Nov. 2020, pp. 299–314. Available:https://www.usenix.org/conference/osdi20/presentation/cho.

[4] S. Agarwal, "Real-time web application roadblock: Performance penalty of HTML sockets," 2012 IEEE International Conference on Communications (ICC), Ottawa, ON, Canada, 2012, pp. 1225-1229, doi: 10.1109/ICC.2012.6364271.

[5] Marwan Fayed, Lorenz Bauer, Vasileios Giotsas, Sami Kerola, Marek Majkowski, Pavel Odintsov, Jakub Sitnicki, Taejoong Chung, Dave Levin, Alan Mislove, Christopher A. Wood, and Nick Sullivan. The Ties That Un-Bind: Decoupling IP from Web Services and Sockets for Robust Addressing Agility at CDN-Scale. In Proceedings of the 2021 ACM SIGCOMM 2021 Conference, SIGCOMM '21, page 433–446, New York, NY, USA, 2021. Association for Computing Machinery.

[6] Fayed, Marwan & Bauer, Lorenz & Giotsas, Vasileios & Kerola, Sami & Majkowski, Marek & Odintsov, Pavel & Sitnicki, Jakub & Chung, Taejoong & Levin, Dave & Mislove, Alan & Wood, Christopher & Sullivan, Nick. (2021). The ties that un-bind: decoupling IP from web services and sockets for robust addressing agility at CDN-scale. 433-446. 10.1145/3452296.3472922.

[7] Mou, Chengjin. (2023). DNS is the Internet Pivotal Basics and Fundamental. International Journal of Advanced Network, Monitoring and Controls. 7. 11-23. 10.2478/ijanmc-2022-0012.

[8] dreamsofcode-io, "grpcalculator-rs," GitHub, 2025. [Online]. Available: https://github.com/dreamsofcode-io/grpcalculator-rs

[9] WebRTC, "WebRTC," https://webrtc.org/ (accessed Mar. 1, 2025).

[10] Miguel Isabal, "8 Powerful Applications Built Using WebRTC," United World Telecom, Dec. 2020. Available:https://www.unitedworldtelecom.com/learn/webrtc-applications/

[11] Gillis, Alexander, "Remote Procedure Call (RPC)", TechTarget, May 2024. Available:https://www.techtarget.com/searchapparchitecture/definition/Remote-Procedure-Call-RPC

[12] S. Srivastava and P. K. Srivastava, "Performance analysis of Sun RPC," 2013 National Conference on Parallel Computing Technologies (PARCOMPTECH), Bangalore, India, 2013, pp. 1-9, doi: 10.1109/ParCompTech.2013.6621405.

[13] M. Aboubakar, P. Roux, M. Kellil and A. Bouabdallah, "A novel scheme for congestion notification in IoT low power networks," 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), Bordeaux, France, 2021, pp. 932-937.

[14] M. Masirap, M. H. Amaran, Y. M. Yussoff, R. A. Rahman and H. Hashim, "Evaluation of reliable UDP-based transport protocols for Internet of Things (IoT)," 2016 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), Penang, Malaysia, 2016, pp. 200-205, doi: 10.1109/ISCAIE.2016.7575063.

[15] O. Romanov, V. Mankivskyi and I. Saychenko, "Analysis of Influence of UDP Parameters on QoS in IP Network," 2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, Ukraine, 2020, pp. 551-556, doi: 10.1109/PICST51311.2020.9467892.

Business Plan

OVERVIEW

Mnemo enables devices on a local network to exchange real-time variables with a central server. By leveraging technologies such as raw UDP sockets, WebRTC, and RPC, Mnemo delivers near-instantaneous updates for interactive presentations, classroom clickers, and other live events. Users join easily—by scanning a QR code that automatically connects them without any manual IP entry—while the lightweight design minimizes latency.

HOW BIG A DEAL IS Mnemo?

Interactive applications are increasingly vital in education, conferences, and live events. With nearly every smartphone on hand, a low-cost, low-latency system like Mnemo could redefine real-time interactivity in these settings. In an era where milliseconds matter, our approach can dramatically improve user experience compared to traditional web solutions, opening the door to a broad market including educational institutions, corporate training, and event organizers.

FINANCES

Initial investments for Mnemo are effectively $0. Our self-hosted approach leverages existing development devices and a lean infrastructure, allowing us to offer a simple, quick-to-deploy interface without significant upfront costs. For our minimum viable product (MVP), we focus on rapid onboarding and ease of use while ensuring a robust, secure LAN-based system.

Mnemo is designed not only to serve interactive presentations but also to open diverse revenue channels:

Once the MVP is validated, additional investments in infrastructure and marketing will support scaling, ensuring that Mnemo remains a low-cost, high-performance solution with sustainable growth potential.

DETAILS

The plan is to generate revenue through recurring subscription fees from educational institutions and individual speakers, while also securing strategic partnerships and licensing deals with universities and training centers. Additionally, offering custom solutions for large-scale events can command premium pricing.