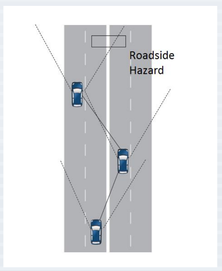
# Virtual Front View: Enhancing Driving Safety with Multi-hop Video Streaming

**Cameron Cairns, Matthew Morikawa, Corina Putinar, Shawn Shojaie, Doron Zehavi**

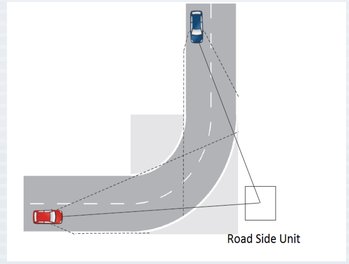
Faculty Mentors: Professor Dipak Ghosal, Professor Premkumar Devanbu

## Motivation

Virtual Front View is a method to extend the visual range of the driver by providing a video stream of downstream traffic and/or road conditions that may not be visible to the driver. The video stream maybe generated by a Road Side Unit (RSU) or by an downstream vehicle and the video stream may be received using single-hop or multi-hop using vehicle-to-vehicle (V2V) communication. The motivation is to improve driving safety by extending the visual range of the driver. This project is motivated by a multi-disciplinary National Science Foundation (NSF) project that is currently underway and led by Prof. Michael Zhang (CEE), Prof. Chen-Nee Chuah (ECE), and Prof. Dipak Ghosal (CS).




*This is a multi-hop example; where Car A is getting video stream from Car C using Car B as an intermediary.*




*This is how a Road Side Unit (RSU) can be used to relay a video stream to a car in range.*

## Project Scope

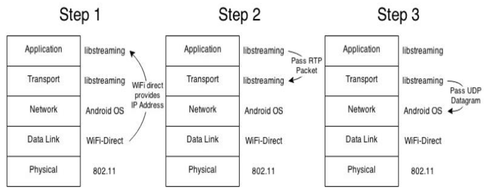
Within the broader scope, the goal of our project was to develop a prototype Android application that can used to demonstrate the feasibility of the idea using standard Android devices. An additional goal of the project was to characterize the packet loss and delay in single hop communication and its impact on the quality of the video stream. Towards this end we adopted Andriod Wi-Fi Direct for ad-hoc communication between the Android devices. For video streaming, we adopted the libstreaming library, which also comes with standard Android OS. Libstream library uses the standard Real Time Streaming Protocol (RTSP) which uses Real Time Protocol (RTP) for streaming the video data.

## Architecture

Our Android applications is based on the standard client server architecture. The architecture uses the standard TCP/IP network protocol stack as shown below. The application processes use libstream which handles the top two layers of the network protocol stack, namely, the application layer and the transport layer functions. Wi-Fi Direct use the IEEE 802.11 protocol and implements data link and physical layer protocols.

Wi-Fi Direct retrieves the network layer identifier, the IP Address, and passes it to the application process which makes it accessible to the libstream library. At the sender side, the libstream library provides functions to convert video into RTP packets which are transported using UDP/IP. At the receiver, libstream library functions are used to render the video based on the received video packets.

*This diagram exhibits the architectural steps our program takes using the five layer network protocol stack.*




## Implementation: Server

```
// Sets the port of the RTSP server to 8988
Editor editor = PreferenceManager.getDefaultSharedPreferences(
                getApplicationContext()).edit();
editor.putString(RtspServer.KEY_PORT, String.valueOf(8988));
editor.commit();


// Configures the SessionBuilder
SessionBuilder.getInstance().setSurfaceView(mSurfaceView)
              .setPreviewOrientation(0).setContext(this) // Orientation
              .setVideoQuality(new VideoQuality(resX, resY, 20, bitrate)) /Video Quality
              .setAudioEncoder(SessionBuilder.AUDIO_NONE) // No audio
              .setVideoEncoder(SessionBuilder.VIDEO_H264); // Encoding: H264
// Starts the RTSP server
getApplicationContext().startService(
              new Intent(getApplicationContext(), RtspServer.class));
```

**ServerActivity.java:** This class is only opened on the Android device that is serving the video stream.

## Implementation: Client

```
// Sets up the SurfaceView for displaying the video
mSurfaceView = (SurfaceView) findViewById(R.id.surface_view);
mSurfaceHolder = mSurfaceView.getHolder();
mSurfaceHolder.addCallback(this);

// Once the the surface is created (above), the MediaPlayer API is created
//     and sets the source of the video to the given IP, prepares and then
//     displays the video in the ClientActivity.
@Override
public void surfaceCreated(SurfaceHolder holder) {
        try {
                mMediaPlayer = new MediaPlayer();
                mMediaPlayer.setDisplay(mSurfaceHolder);
                mMediaPlayer.setDataSource("rtsp://" + mVideoIP
                                + ":8988");
                mMediaPlayer.prepare();
                mMediaPlayer.setOnPreparedListener(this);
                mMediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
                mMediaPlayer.setScreenOnWhilePlaying(true);
        } catch (IllegalArgumentException e) {
                e.printStackTrace();
        } catch (SecurityException e) {
                e.printStackTrace();
```

**ClientActivity.java:** This class is only opened on the Android device that is viewing the video stream.

## Experimental Setup

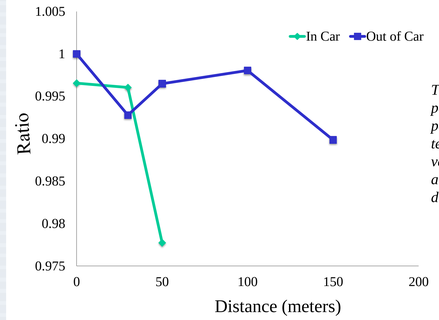
We used tcpdump on our test phones to watch the traffic that was transported over the wireless network. This was made simpler due to the fact that the Wi-Fi Direct protocol uses a non-standard network interface, and so no other traffic uses the same interface. This allowed us to assume that all traffic travelling over the WiFi-direct network interface was traffic generated by the Virtual Front View application.

We tested our application in a parking lot at different distance intervals. We choose 0, 30, 50, 100, 150 meter intervals. For each distance we ran our application for one minute and recorded the total packets observed by tcpdump at the sender and the receiver. This allowed us to determine the packet loss rate and how it is affected by the distance.

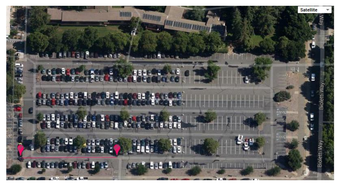
## Packet Analysis

Packet data was gathered to see the number of packets received and sent. The data shows that there is almost no packet loss until Wi-Fi Direct cannot connect anymore.

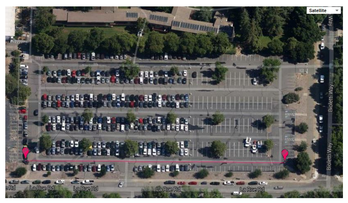
## Packet Analysis




*This is the ratio of packets received to packets sent when testing inside a vehicle and outside a vehicle at varying distances.*



*The maximum distance for the data gathered inside two vehicles is 50 meters*



*The maximum distance for the data gathered while walking, outside of vehicles is 150 meters*

## Future Work

For our next steps, we plan on optimizing the Virtual Front View code to reduce lag from 3 seconds to under 1 second. We also want to implement a multi-hop streaming protocol and thus be able to capture and show video data from cars that are further downstream to provide the driver additional time to avoid hazards.

## Acknowledgements

We would like to acknowledge Professor Premkumar Devanbu, Professor Dipak Ghosal, Ali Emara, and the Computer Science Department at the University of California, Davis