Low Latency Camera Feed Development

This work was supported in part by the NSF REU program and the donation from nVERSES CAPITAL
Researchers Introduction

Brayden Casaren  Sebastian Clarke  Rohit Karthickeyan  Ayush Iyer
Our Mission

➢ To find the method(s) of reducing latency to a minimum in a unicast camera to computer connection over a network
The Importance of Low Latency Cameras

The modern day world requires low latency cameras for many applications including:

- Self driving cars
- Live Sporting Events and television
- Interviews and News
- Security and Surveillance
- Remote Work Operations (like Zoom)
Project Design

Testing Setup Diagram
Packets

Frame 1518: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
Internet Protocol Version 4, Src: 10.41.250.42, Dst: 10.41.1.11
Source Port: 80
Destination Port: 38868

This shows the raw value of the sequence number (tcp.seq_raw), 4 bytes

Packets: 1858 · Displayed: 1858 (100.0%)
GPS, PPS, and PTP

GPS: Global Positioning System

PPS: Pulse per second

PTP: Precision Time Protocol
Creating Artificial Video

Created videos of varying noises (0%-50%) which were streamed from our secondary node to our primary over VLC

- This setup will allow us to simulate how noise affects our actual camera.
Visualizing our data

❖ Using Matplotlib to create the histograms of our data (byte occurrences)

❖ Useful for visualizing data particularly for comparison and pattern recognition (more than million pieces of data)

❖ Want to gain information of how noise affects packets and eventually use for identify when changes occur in our video
We changed the ways our histogram worked in a couple of ways but ended up deciding on this:

- Measures RGB pixel values instead of bytes
- Creates three separate graphs for each color (RGB) of how often certain values occurred
More changes we made

Through camera settings made the noise observed very little

Changed LED setup to take up more of the camera’s view

Used FFMPEG to get video into one MJPEG or and its constituent JPEG files
## Calculating Latency

A screenshot of Wireshark showing packet traces and latency calculations. The table contains columns for **No.**, **Time**, **Source**, **Destination**, **Protocol**, **Length**, and **Info**. Each row represents a packet with details about its transmission time and destination IP address.
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>196</td>
<td>18:06:08.945515</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>1514</td>
<td>80 → 38668 [ACK] Seq=249009 Ack=392 Win=6864 Len=1448 TVal=3...</td>
</tr>
<tr>
<td>197</td>
<td>18:06:08.945602</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>1314</td>
<td>80 → 38668 [ACK] Seq=250457 Ack=392 Win=6864 Len=1248 TVal=3...</td>
</tr>
<tr>
<td>198</td>
<td>18:06:08.946447</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>68</td>
<td>80 → 38668 [PSH, ACK] Seq=251705 Ack=392 Win=6864 Len=2 TVal=2...</td>
</tr>
<tr>
<td>199</td>
<td>18:06:08.977719</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>1514</td>
<td>80 → 38668 [ACK] Seq=251707 Ack=392 Win=6864 Len=1448 TVal=3...</td>
</tr>
<tr>
<td>200</td>
<td>18:06:08.978810</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>1514</td>
<td>80 → 38668 [ACK] Seq=253155 Ack=392 Win=6864 Len=1448 TVal=3...</td>
</tr>
<tr>
<td>201</td>
<td>18:06:08.979934</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>1514</td>
<td>80 → 38668 [ACK] Seq=254683 Ack=392 Win=6864 Len=1448 TVal=3...</td>
</tr>
</tbody>
</table>

Source Port: 80  
Destination Port: 38668  
[String index: 1]  
[TCP Segment Len: 1448]  
[relative_segment_number]  
Sequence number: 3513273  

This shows the raw value of the sequence number (tcp.seq_raw), 4 bytes
<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>223</td>
<td>18:06:09.0443808</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>1514</td>
<td>80 → 38668 [ACK] Seq=283167 Ack=392</td>
</tr>
<tr>
<td>224</td>
<td>18:06:09.0443893</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>1514</td>
<td>80 → 38668 [ACK] Seq=284615 Ack=392</td>
</tr>
<tr>
<td>225</td>
<td>18:06:09.0445102</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>1514</td>
<td>80 → 38668 [ACK] Seq=286063 Ack=392</td>
</tr>
<tr>
<td>226</td>
<td>18:06:09.0446411</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>1514</td>
<td>80 → 38668 [ACK] Seq=287511 Ack=392</td>
</tr>
<tr>
<td>227</td>
<td>18:06:09.0447582</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>1514</td>
<td>80 → 38668 [ACK] Seq=288959 Ack=392</td>
</tr>
<tr>
<td>228</td>
<td>18:06:09.0450801</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>1514</td>
<td>80 → 38668 [ACK] Seq=290407 Ack=392</td>
</tr>
<tr>
<td>229</td>
<td>18:06:09.0451181</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>1514</td>
<td>80 → 38668 [ACK] Seq=291855 Ack=392</td>
</tr>
<tr>
<td>230</td>
<td>18:06:09.0452460</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>1514</td>
<td>80 → 38668 [ACK] Seq=293303 Ack=392</td>
</tr>
<tr>
<td>231</td>
<td>18:06:09.0453643</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>1514</td>
<td>80 → 38668 [ACK] Seq=294751 Ack=392</td>
</tr>
<tr>
<td>232</td>
<td>18:06:09.0454871</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>1514</td>
<td>80 → 38668 [ACK] Seq=296199 Ack=392</td>
</tr>
<tr>
<td>233</td>
<td>18:06:09.0455933</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>1514</td>
<td>80 → 38668 [ACK] Seq=297647 Ack=392</td>
</tr>
<tr>
<td>234</td>
<td>18:06:09.0457770</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>1514</td>
<td>80 → 38668 [ACK] Seq=299905 Ack=392</td>
</tr>
<tr>
<td>235</td>
<td>18:06:09.0459026</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>1514</td>
<td>80 → 38668 [ACK] Seq=300543 Ack=392</td>
</tr>
<tr>
<td>236</td>
<td>18:06:09.0460265</td>
<td>10.41.250.42</td>
<td>10.41.1.11</td>
<td>TCP</td>
<td>1514</td>
<td>80 → 38668 [ACK] Seq=301991 Ack=392</td>
</tr>
</tbody>
</table>

Frame 237: 122 bytes on wire (976 bits), 122 bytes captured (976 bits)

0000 98 03 9b 3b 81 cf 00 40 8c c6 a4 97 00 00 45 00
0010 00 6c 84 00 40 00 40 00 06 a7 04 a0 0a 29 fa 0a 29
0020 01 0b 00 50 97 0c 89 c7 12 64 98 2a 47 aa 80 10
0030 06 b4 4d 30 30 01 00 00 00 86 a0 0a 12 13 f2 54 d0 32
0040 98 04 8a 00 28 a2 8a 00 28 a2 8a 00 28 a2 8a 00
0050 28 a2 8a 00 28 a2 8a 00 28 a2 8a 00 28 a2 8a 00
0060 28 a2 8a 00 28 a2 8a 00 28 a2 8a 00 28 a2 8a 00
0070 ff d9
hexfile = open("output_0018.txt", "r")
packetsfile = open("P223.txt", "r")
hex = ""
img = ""
packets = []
file = packetsfile.read()
file = file.split()
for i in file:
    hex = hex + i.upper()
hex = hex[hex.index("FFD8") :]
image = hexfile.read()
image = image.split()
for i in image:
    img = img + i.upper()
if(img[:len(hex)] == hex):
    print("Match")
else:
    print("No Match")
Reducing Latency:

Viable methods:

Changing Codec, Camera can run MJPEG and H264
Changing Framerate, Camera can operate between 0-30 fps
Changing Compression amount, Camera defaults to 30 on scale of 0-100
Switching Codec (H264)

- 3 Types of frames: I, P, and B
- Frame referencing and motion vectors with P and B frames to reduce size
- Calculated latency to be around 34 ms for just I frames, 60 ms for GOV of 3
THANK YOU

Any Questions?