May 2020

DETERMINING WIRELESS BANDWIDTH CAPABILITIES FOR AUTOMOBILE HEAD UNITS

Fredrik Hubinette

DETERMINING WIRELESS BANDWIDTH CAPABILITIES FOR AUTOMOBILE HEAD UNITS

ABSTRACT

This paper describes techniques that relate to estimating latency in buffered systems and, more particularly, for evaluating wireless bandwidth capabilities in automobile head units (e.g., infotainment systems) that are configured to process and display data wirelessly received from mobile computing devices (e.g., smartphones, smartwatches). For example, an automobile head unit may wirelessly receive display data at a given resolution level from a mobile computing device, and may display the received data at a display device of the head unit. The techniques disclosed herein provide a method to test or determine if a particular automobile head unit has sufficient wireless bandwidth capabilities to support receiving and displaying data at one or more resolution levels. The techniques utilize information associated with required bandwidth and available bandwidth to simulate the buffering behavior of a particular setup under evaluation, and may generate various data points in the form of estimated latencies. The techniques then generate a histogram of these latencies, from which one or more thresholds can then be established to determine whether a particular automobile head unit demonstrates sufficient performance while processing received display data.

DESCRIPTION

As indicated above, the present paper describes techniques for evaluating wireless bandwidth capabilities in automobile head units that are configured to process display data that is wirelessly received from mobile computing devices. Over time, an automobile head unit may be wirelessly paired with one or more different mobile computing devices that are configured to output data for display on the head unit at one or more resolution levels. For example, the automobile head unit may support display resolution levels of 480p, 720p, and/or 1080p, and may be configured to wirelessly receive display data encoded at one or more of these resolution levels from mobile computing devices that are wirelessly coupled to the head unit. To determine whether a particular head unit performs with sufficient performance when processing such display data, techniques are needed to measure wireless throughput between the head unit and the mobile computing device to reflect how good the actual user experience is while using the head unit to view displayed data that is provided by the mobile computing device. In many cases, measuring an average bit rate may be insufficient in accurately assessing performance.

In some cases, a test setup may execute a test (e.g. a 10-minute test) that tries to saturate the available wireless bandwidth between the automobile head unit and the mobile computing device, such as by implementing a test program on the mobile computing device that sends a high volume of display data to the head unit. The test program provides as much display data as it can to the head unit, at one or more display resolutions, and the test program may record how much data it was able to send once per second. At the end of execution of the test program, the program or a user of the program may determine whether or not the head unit's performance is sufficient for the resolutions that the head unit claims to support.

A common way to make this determination is to use percentiles. For instance, for a supported resolution level of 480p, the head unit may be required to be able to handle 12Mbit/s of display data 98% of the time. While this type of assessment may work, it may fail to capture how well the head unit is actually performing, because it doesn't necessarily take into account exactly how many Mbit/s the head unit can handle 98% of the time. For example, a head unit that can process 100Mbit/s of display data 97% of the time, but only 11Mbit/s of data the other

3% of the time, is probably better from a performance perspective than a head unit that can process 13Mb/s of display data 100% of the time. For instance, the following two figures illustrate a few real-world examples (showing data in Mbit/s at one sample per second):

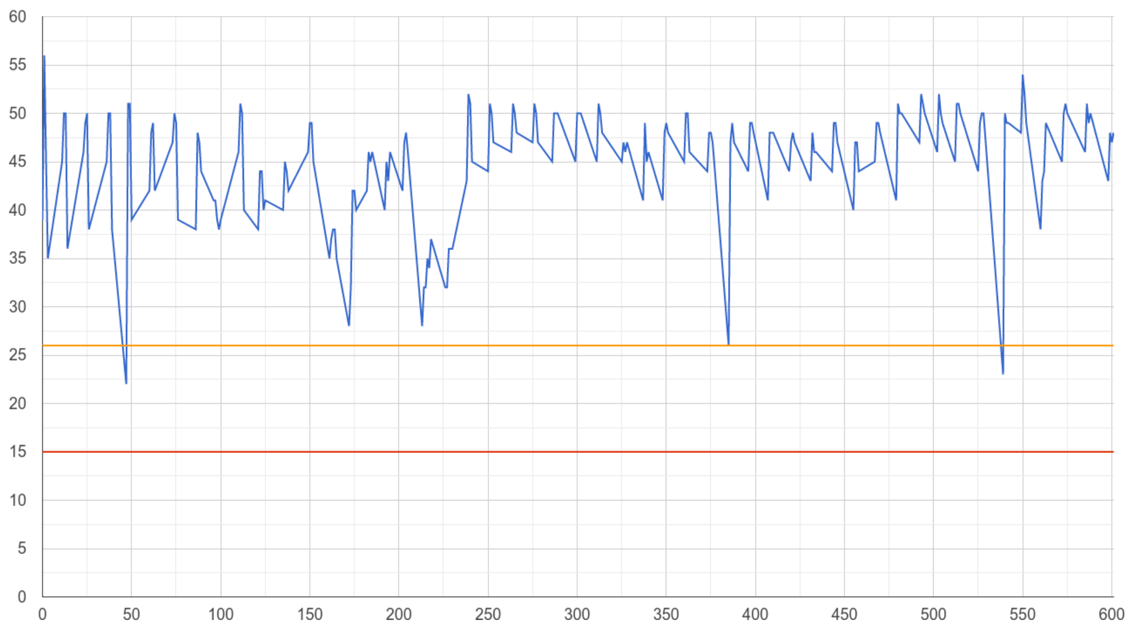

FIG. 1: Bandwidth Results #1

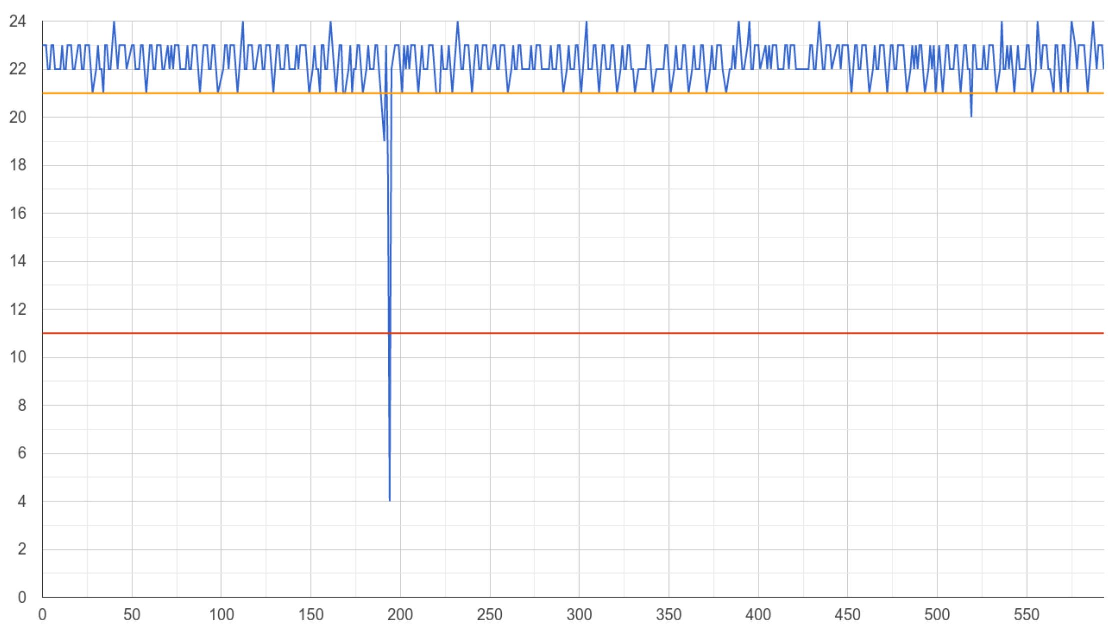

FIG. 2: Bandwidth Results #2

In the two charts shown in FIGS. 1 and 2, the 97th percentile (yellow line) is fairly similar. In FIG. 1, the yellow line indicates that a first head unit of this example is capable of processing 26Mbits/s of display data 97% of the time for a particular resolution level. In FIG. 2, the yellow line indicates that a second head unit of this example is capable of processing 21Mbit/s of display data 97% of the time for this particular resolution level. These processing bandwidths are similar. Clearly, however, the first head unit associated with the results of FIG. 1 has much higher performance than the second head unit associated with the results of FIG. 2. For the most part, the first head unit of FIG. 1 is capable of processing display data at rates much higher than 26Mbits/s, while the second head unit of FIG. 2 is capable mainly of processing display data at rates in the range of 21-24Mbits/s. Thus, reliance on average bandwidth, or bandwidth based on percentiles, fails to provide a clear picture regarding actual processing bandwidth for display data at various resolution levels.

Using techniques of the present paper, a test setup may measure how much data needs to be sent over a period of time, as well as how much data can actually be sent. A test program may then simulate the buffering that happens in the network layer. For example, if the test program executing on a mobile computing device is unable to send a certain quantity of display data to the head unit at a given time, the data may be buffered for later transmission. Simulating the buffering that occurs provides a probability distribution of latencies that will more accurately measure how well a head unit can perform than the use of bandwidth percentiles outlined above. The test program, or a user (e.g., administrator or certification tester) may then choose one or more cut-off points based on those probabilities.

FIG. 3 shown below illustrates the framework system in which the test program executes.

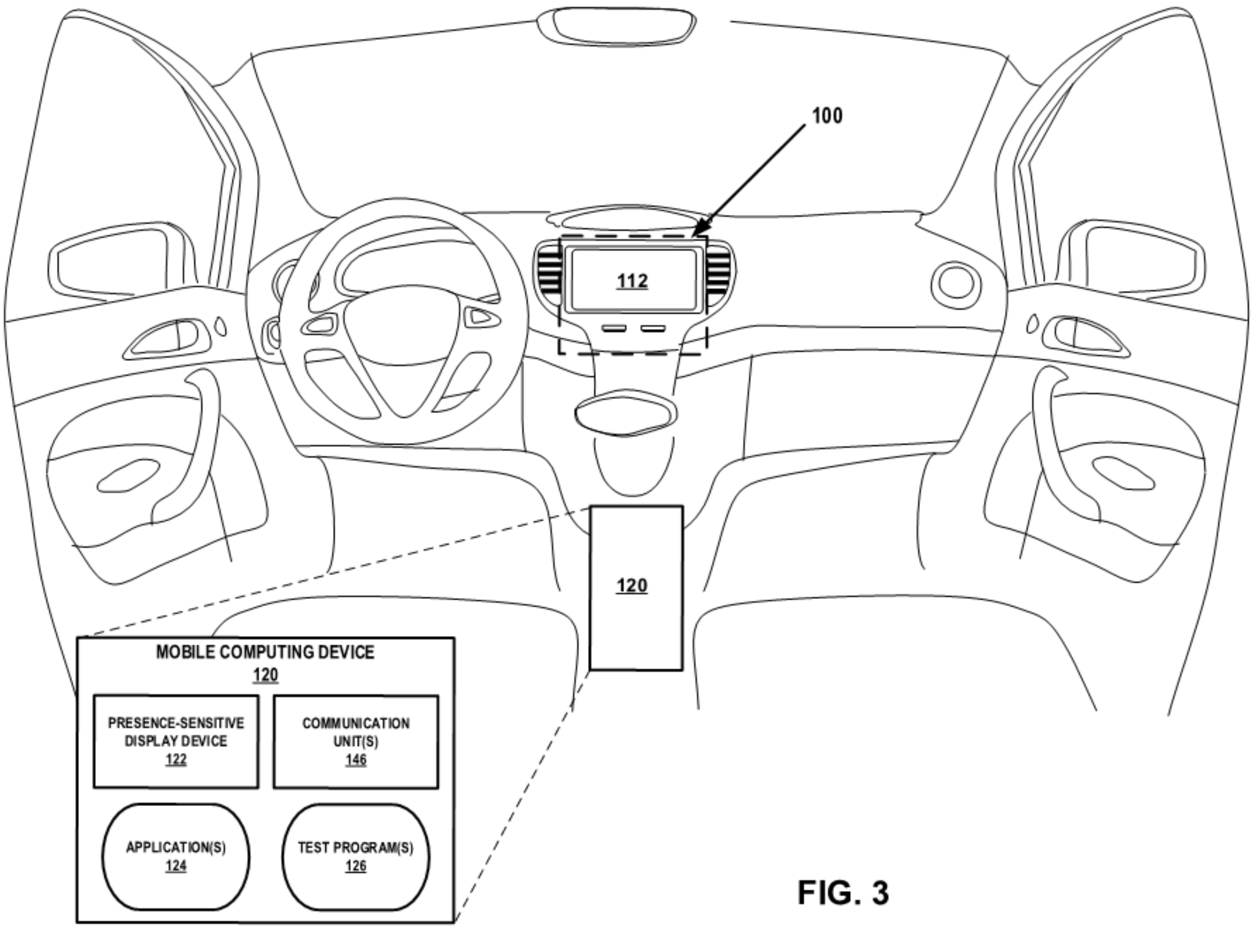

FIG. 3

FIG. 3 illustrates an example automobile having a head unit 100. The head unit 100 includes a display device 112 that is configured to display data that is transmitted (e.g., via a wireless connection) to head unit 100 by a mobile computing device 120. As illustrated in FIG. 3, mobile computing device 120 (e.g., a smartphone or smartwatch) may include a presence-sensitive display device 122 and one or more communication units 146. Mobile computing device 120 may also use one or more processors (not shown) to execute applications 124 and one or more test programs 126. During execution of applications 124 and/or test programs 126, mobile computing device 120 may use communication units 146 to send display data associated with applications 124 and/or test programs 126 to head unit 100, such as via one or more wireless connections.

In many cases, what really matters to the end user, such as the automobile operator of using mobile computing device 120, is how long it takes for the video frames sent by mobile computing device 120 to appear on display device 112 of head unit 100. If the data latency is low, the data will display quickly on display device 112, and the end user will likely feel quite satisfied using mobile computing device 120 with head unit 100. On the other hand, if the data latency is high, mobile computing device 120 may begin buffering data that needs to be sent to head unit 100, and the display of received data at display device 112 of head unit 100 will appear slow and choppy to the end user, which is not preferred. In the general sense, as long as applications 124 and/or test programs 126 use less wireless bandwidth to transmit display data than what is available, latency issues should be minimized. However, the question becomes: what happens when that's not the case?

To address this question, test programs 126 may assess and graph how much display data is produced by mobile computing device 120 for wireless transmission to head unit 100. For instance, the following two figures illustrate a few real-world examples (again showing data in Mbit/s, one sample per second):

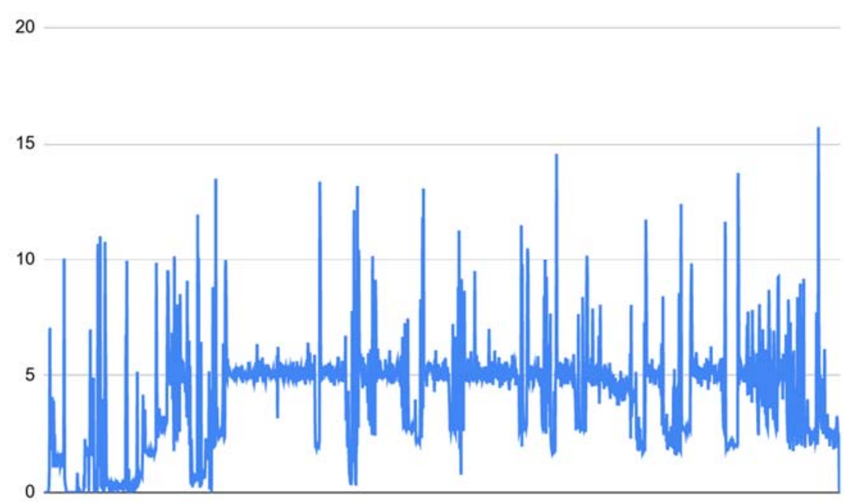

FIG. 4: Data Bandwidth Usage #1

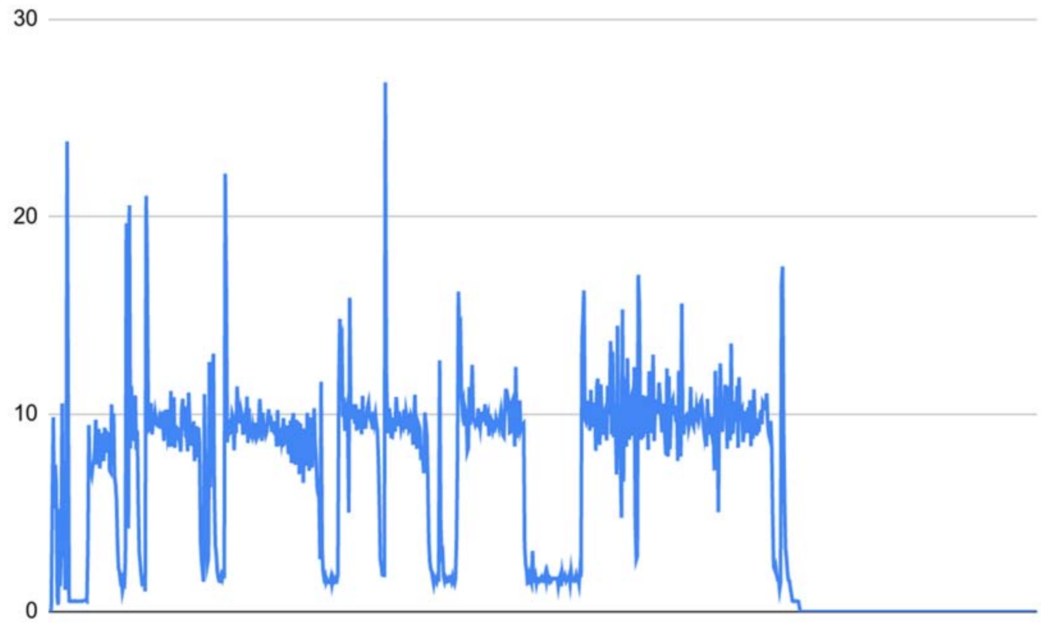

FIG. 5: Data Bandwidth Usage #2

As is shown in both FIGS. 4 and 5, both the bandwidth usage as well as the available bandwidth usage are both very noisy over the measured time interval. Also, the second bandwidth usage graph shown in FIG. 5 generally hovers around a higher value than the first bandwidth usage graph shown in FIG. 4. In certain cases, this may be due to the fact that applications 124 and/or test programs 126 are generating and transmitting display data at a higher resolution level in the example of FIG. 5.

However, by combining the information associated with transmitted data bandwidth and information associated with available bandwidth on the same chart, the test programs 126 or individuals using the test programs 126 are able to identify certain areas of overlap. FIG. 6, as shown below, illustrates an example chart that includes this combined information.

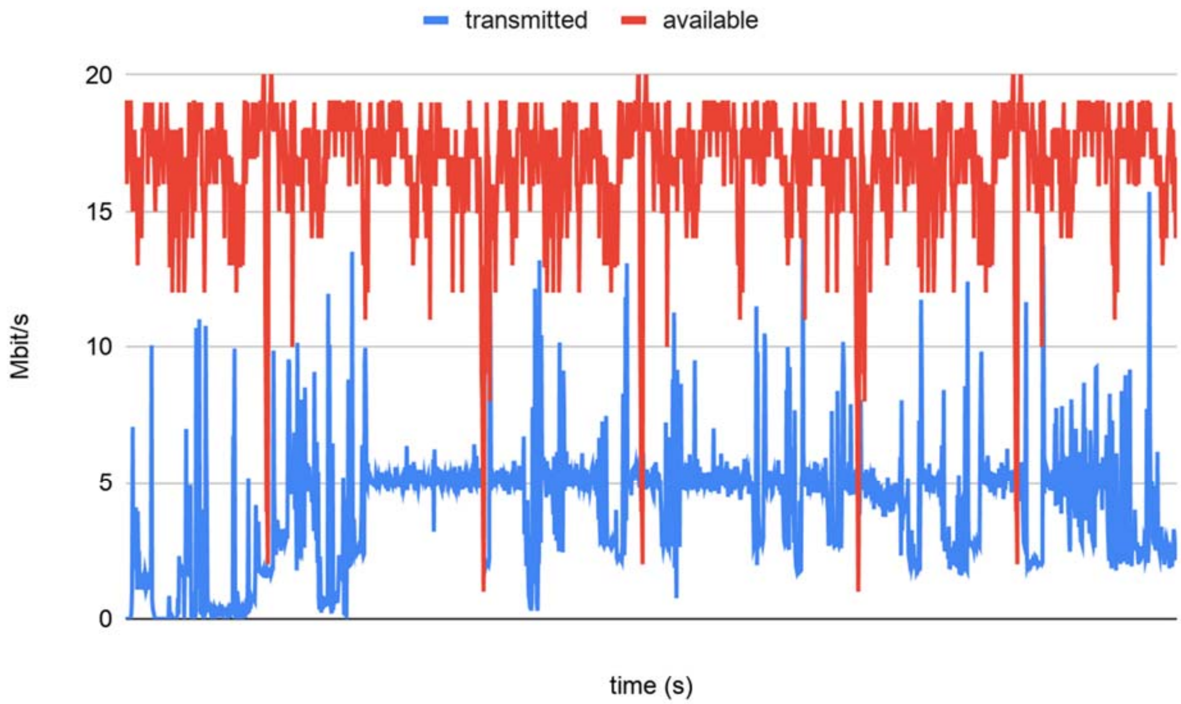

FIG. 6: Transmission Bandwidth Combined With Available Bandwidth

As indicated in FIG. 6, the areas of overlap are those areas in which blue data points (i.e., transmitted data bandwidth) overlay red data points (i.e., available bandwidth). When there is no overlap, the time it takes to send a frame from mobile computing device 120 to head unit 100 is simply *(size_of_frame) / (transmit_rate)*. However, when there is overlap, there will be data stored in the transmit buffer of mobile computing device 120 that has to be transmitted before device 120 can send a new frame, and the number of items within the buffer can grow if the overlap persists over time.

The test program 126 may determine, for each point in the graph of FIG. 6 above, how much data is in the data transmission buffer at any given point in time, based on the current available bandwidth. For example, test program 126 may utilize the following algorithm:

```
for t in time:
    buffer.add(transmitted[t])
    buffer.remove(available_bandwidth[t])  # clamped to zero
    yield buffer.currentSize() / available_bandwith[t]
```

By dividing by the available bandwidth in each point (except when the available bandwidth is zero), test program 126 is capable of determining an estimate of how long it would take for the buffer to empty, which is related a measurement of latency with respect to data that is transmitted from mobile computing device 120 to head unit 100. In various cases, the algorithm above, which determines the amount of data within the transmission buffer at a given point in time, is a good estimate of how much time data spends in the buffer (or how much time it takes for this data to be emptied from the buffer).

In some cases, the two graphs shown above in FIG. 6 are not overlapping, and the size of the transmission buffer will be zero in these cases. FIG. 7 shows an example of what the simulation of test program 126 would output in these cases, plotting estimated latency over time, where latency is estimated based on the algorithm shown above.

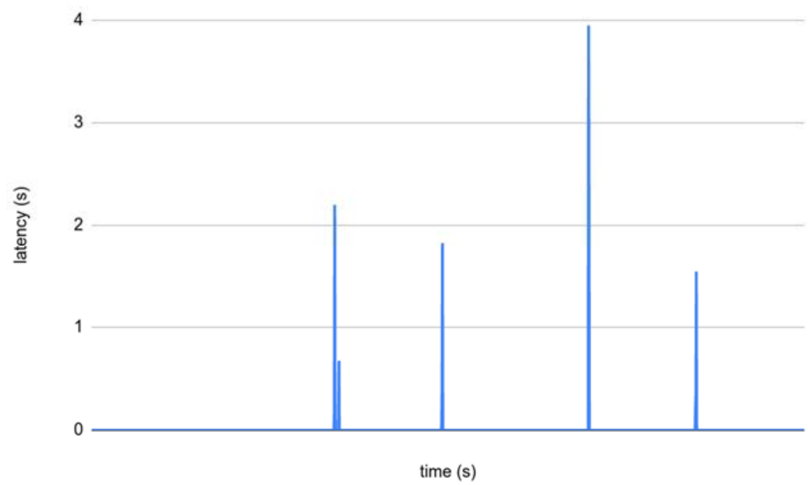

FIG. 7: Estimated Latency Over Time

FIG. 7 does not necessarily illustrate a high volume of data, although there are spot latencies of approximately 4 seconds at certain points, which may not be optimum. However, there are a large volume (e.g., 1600) of measuring points, so with only four spikes to go by in FIG. 7, it is hard to determine if the 4 seconds of latency occurs very often. This situation can be remedied based on the knowledge that the measured data comes from two completely different data sources associated with transmitted bandwidth and available bandwidth. In one given simulation of test program 126, it is assumed that these measurements start at the same time, but that is only one possible assumption. In one particular example, with 1600 data points, the output would generate 6400 spikes. In other examples, any different number of spikes could be generated using different data or a different number of data points, or using different assumptions regarding how the data points line up based on the times at which measurements are started.

In view of this large volume of data, rather than the use of line charts, cumulative distribution functions (CDF's) may be used instead. CDF's are usually displayed with the probability on the Y axis, but in alternate forms such as described herein, latency may be plotted on the Y-axis and probability may be plotted on the X-axis. If test program 126 executes the full set (e.g., 1600) simulations described above, test programs 126 may generate the following chart shown in FIG. 8.

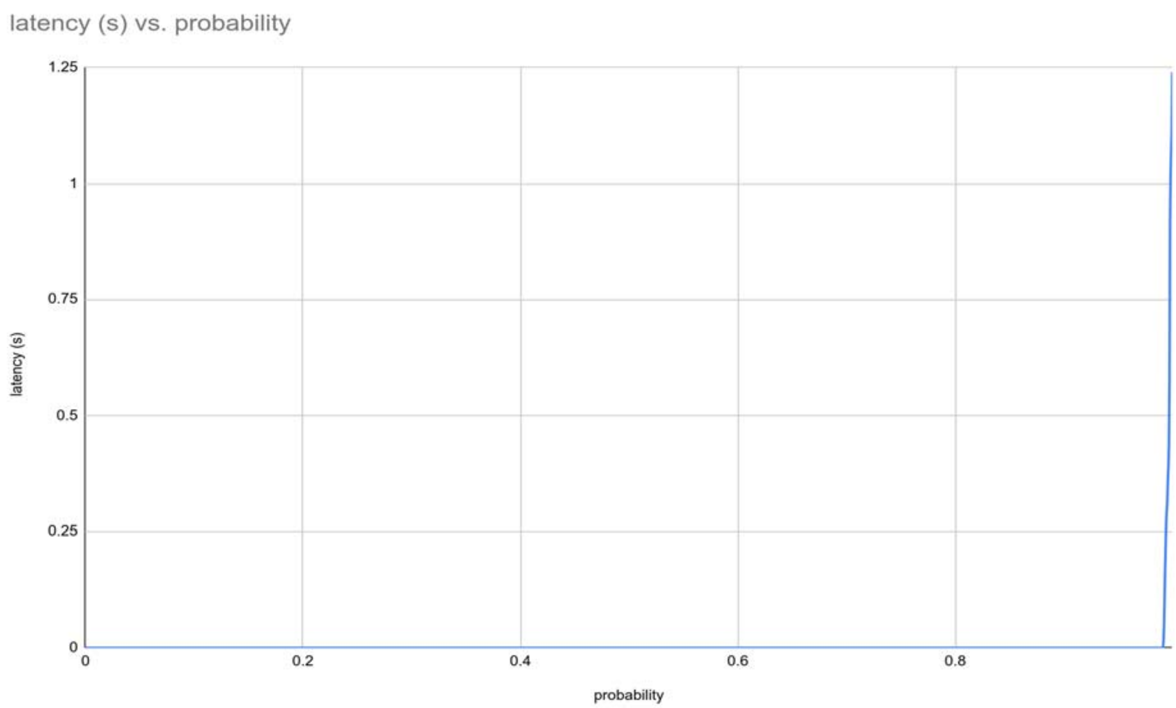

FIG. 8: Latency Cumulative Distribution Function

In order to see further relevant detail on the high-end of the probability axis, FIG. 9 is a version of FIG. 8 that zooms in on the last 10% of probability in FIG. 8 (i.e., the 90-100% probability range on the X-axis).

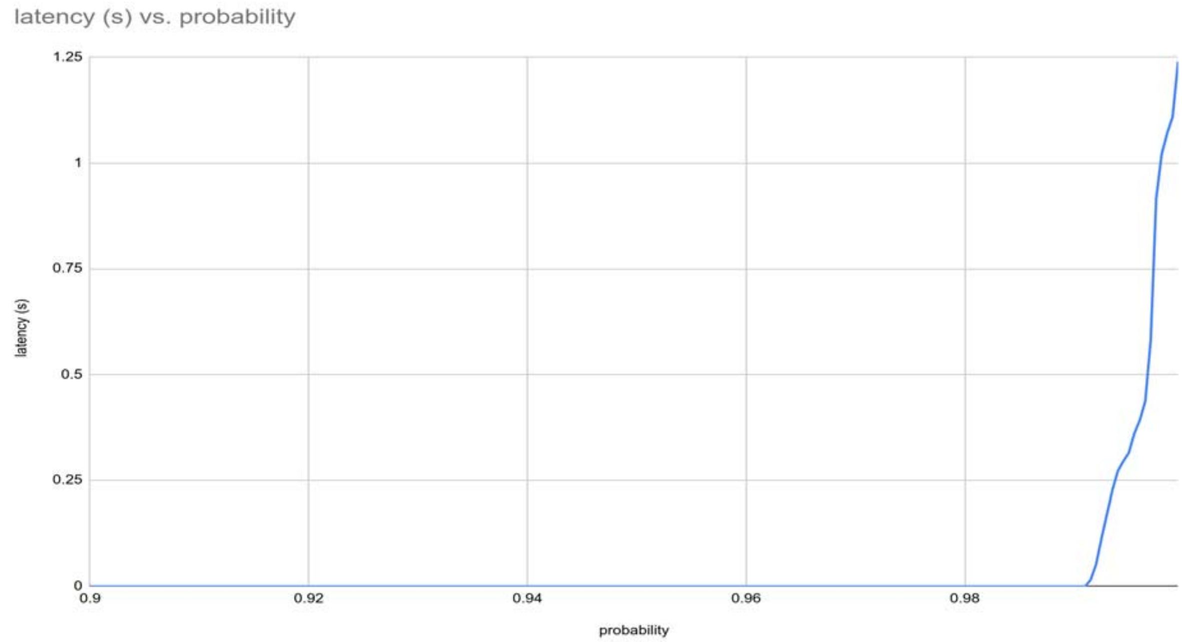

FIG. 9: Latency Cumulative Distribution Function

As shown in FIG. 9, for about 99% of the time, the latency of data transmission from mobile computing device 120 to head unit 100 is zero, not including the time that it takes to send frames to head unit 100. For this 99% of the time, the transmission buffer of mobile computing device 120 was empty when a new frame was placed into the buffer. It is also noted that the example of FIGS. 8 and 9, which illustrate a maximum latency of approximately1.25 seconds, is a different example from that of in FIG. 7, which illustrated a maximum latency of approximately 4 seconds.

Using the CDF's shown in FIGS. 8-9, test program 126 or a user (e.g., certification tester) may determine if any particular head unit, such as head unit 100, performs well enough, and without any problematic latency, to handle the processing of graphics data of a particular resolution that is sent from mobile computing device 120. For example, test program 126 or the user may select one or more particular cut-off points (e.g., a cut-off point associated with a rule that the latency needs to be less than some defined value at least 95% of the time).

In order to select one or more such cut-off points, test program 126 may execute test programs for various different types of head units that provide support for data transmissions at or more resolutions. Test program 126 may then plot CDF’s, similar to that shown in FIG. 9 above, for the tests executed with respect to these different head units. FIG. 10 below shows an example of such CDF’s zoomed in to the last 10% of probability (i.e., the 90-100% probability range on the X-axis).

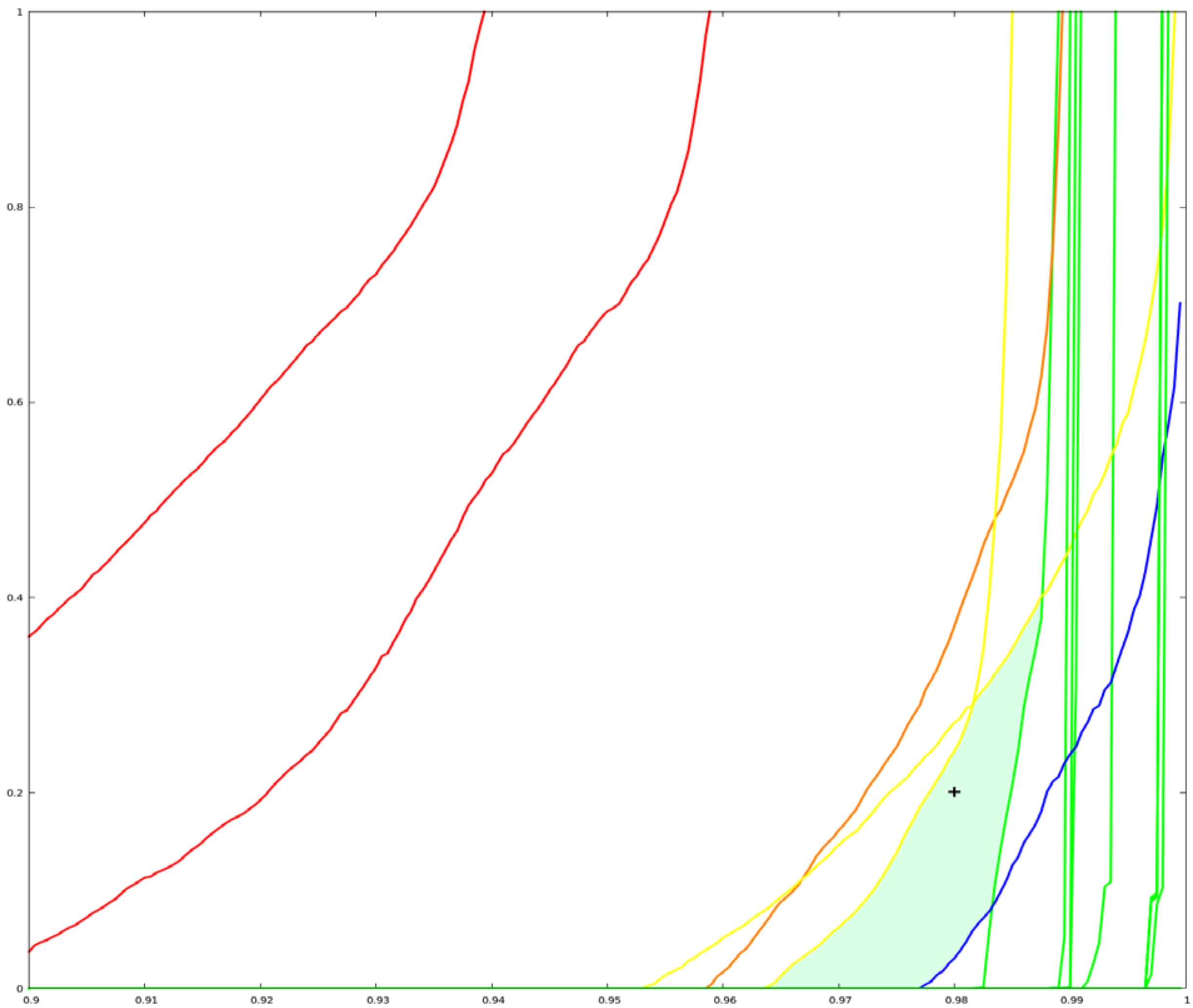

FIG. 10: Multiple Latency Cumulative Distribution Functions

As noted above, this chart of FIG. 10 shows the top 10% of the latency distributions calculated by various test runs of the test program 126. Each curve shown in FIG. 10 is associated with a test executed by test program 126 on a particular head unit and at a particular resolution level. To provide a few non-limiting examples, a red curve in FIG. 10 may be associated with a test on a head unit that uses a 2.4GHz wireless band and that receives display data transmitted from a mobile computing device at a resolution level of 1080p. A green curve in FIG. 10 may be associated with a test on a head unit that uses a 5GHz wireless band and that receives display data transmitted from a mobile computing device at a resolution level of 480p. An orange curve may be associated with a test on a head unit that uses a 2.4GHz wireless band and that receives display at a resolution level of 480p. A yellow curve may be associated with a test on a head unit that uses a 2.4GHz wireless band and that receives display at a resolution level of 720p. These are just a few examples, and any given head unit that supports display data at multiple different resolution levels may be associated with multiple curves in FIG. 10 that are associated with different tests executed by test program 126.

In various cases, based upon automated and/or manual test criteria associated with latency, the green and blue lines in FIG. 10 are associated with tests that are likely determined to be successful tests by test program 126, while the red lines and orange lines are likely determined to be non-successful, or failed, tests. The yellow lines are associated with tests have borderline results. Based upon an assumption that a slightly stricter criteria for successful test completion is more appropriate in many situations, test program 126 or a certification tester may determine that a cutoff point for any subsequent test executed by test program 126 should be somewhere in the light green shaded region in FIG. 10 for a successful head unit test at one or more resolution levels. For instance, test program 126 or the certification tester may suggest a cutoff point for a successful head unit test of 200 ms maximum latency, 98% of the time, which is indicated by the "+" sign located squarely within the green shaded region.

In different examples, any number of one or more different cutoff points located within the green shaded region may be selected. In addition, in certain cases, different cutoff points and associated pass/fail tests may be used for individual resolution levels (e.g., different cutoff points or tests for each of 480p, 720p, and 1080p). For each resolution that a particular head unit (e.g., head unit 100) supports, the generated CDF must meet the associated cutoff point for that resolution level, or the overall test will fail.

Thus, the techniques disclosed herein provide a method to determine if a particular automobile head unit has sufficient bandwidth to support receiving and displaying data at one or more resolution levels. The techniques generate a histogram of latencies, from which one or more thresholds can then be established to determine whether a particular automobile head unit demonstrates sufficient performance while processing received display data.

REFERENCES

1. United States Patent Application Publication No. 20190280954A1.

2. United States Patent Application Publication No. 20190238436A1.