Analyzing SMPTE ST 2110 Streams Using EBU's Open-Source Software

By Ievgen Kostiukevych, Willem Vermost, and Pedro Ferreira

Abstract

The transition to IP-based studios is probably the biggest technology change broadcasters have faced over the past 30 years. The European Broadcasting Union (EBU) Live IP Software Toolkit (LIST) provides a software-only off-line analyzer that is able to verify the conformance of streams to the SMPTE ST 2110 standards. This article starts by explaining the typical LIST off-line analyzer workflow, then focuses on its architecture designed so that it is accurate, performant, and flexible enough to plug in analysis modules for future needs. Finally, we demonstrate the LIST's major features, step by step, with real-world examples.

Keywords

Analyze, conformance, container, dematerialized, measure, network compatibility model, virtual receive buffer

Introduction

The ultimate goal of the transition to IP-based production facilities is to enable flexible, scalable, and shareable live video production platforms. The industry is moving away from its dedicated SDI technology toward the use of IP in its core production platform. Despite all of the advantages that are foreseen, this is a huge challenge for both customers and vendors, namely, regarding how to test and monitor signals. It is also an opportunity, as this shift makes it possible to build tools based on software running on off-the-shelf hardware. Even accurate test and measurement tools can be built without the need for specific field-programmable gate array (FPGA)-based implementations. The added value of this shift can only be achieved by moving toward pure software running on off-the-shelf hardware and eventually virtualized servers. However, video and audio signals are highly sensitive to timing variations. Changing from a circuit-based network to a statistically multiplexed network brings the challenge of keeping the packet delay variation within the limits of the isochronous application running on that network.

This article starts by explaining the typical LIST off-line analyzer workflow, then focuses on its architecture designed so that it is accurate, performant, and flexible enough to plug in analysis modules for future needs.

The proof of the pudding is in the eating. The development of the European Broadcasting Union (EBU) Live IP Software Toolkit (LIST), a suite of software tools designed to measure and verify the state of IP networks, helps to tame these challenges, giving a push for the industry to move faster toward this goal.[1]

Analyzer

As the industry started moving toward IP, several stakeholders pointed out the need for a tool that enabled the verification of the early implementations. One use case is to validate a device in terms of its compatibility with SMPTE ST 2110.[2]

What if a user wants to validate the video stream generated by a camera? For that purpose, the user connects the camera and a video monitor to a network switch and verifies that the video is displayed correctly on the monitor. In order to analyze it, the user connects a capture device—say a PC with an appropriate network interface card (NIC)—to the switch and instructs the switch to mirror the ports to which the camera and the PC are connected. The user then runs an appropriate capture software, for example, tcpdump[3] or Wireshark,[4] and records a short period of traffic, for example, 1 sec. The user then loads the resulting packet capture (pcap) file in the off-line analyzer, which will process it and present the results to the user. Another use case is to periodically check devices even while they are operating. A similar procedure can be used, mirroring the relevant ports in the switch and capturing network traffic samples.

Workflow

The workflow is as follows: the user captures network traffic from a relevant link on the network, and this traffic is recorded as pcap files that are then

Digital Object Identifier 10.5594/JMI.2019.2899712
Date of publication: 25 April 2019

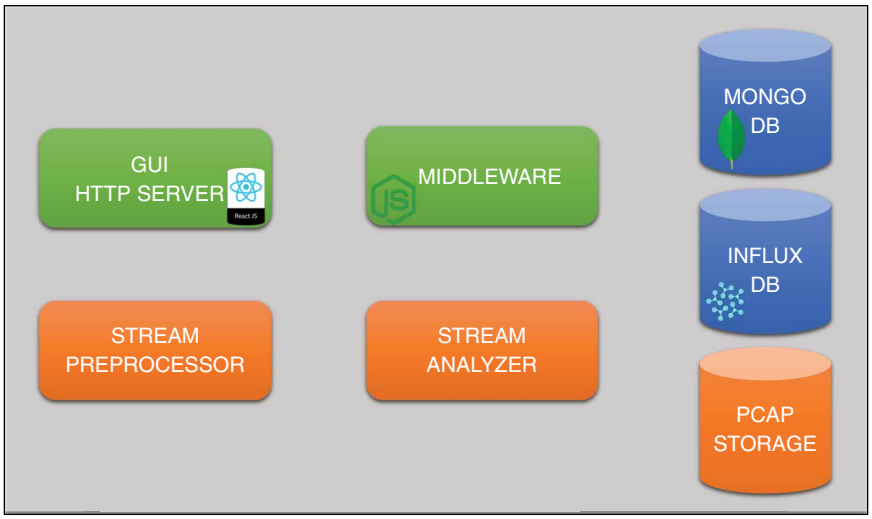

FIGURE 1. Block diagram.

loaded into the tool. Then, the tool automatically detects the individual streams and performs the analysis, presenting the user with a summary of the results (a traffic-light that informs the user if the streams are valid or not) and thorough analysis, including graphs that show the evolution of relevant measurements throughout the duration of the capture. One of the most relevant pieces of information is the stability of the device's media clock with regard to the grandmaster clock. The off-line analyzer displays a graph of the deviation of the device's media clock relative to the master clock.

It is also capable of decoding the video streams (SMPTE ST 2110-20), presenting the individual frames or fields, decoding the audio streams (SMPTE ST 2110-30) allowing the user to listen to them via the web browser, as well as decoding the ancillary streams (SMPTE ST 2110-40). To fulfill the limits of packet delay variation defined in SMPTE ST 2110-21, the network compatibility model (C) as well as the virtual receiver buffer model (VRX) were implemented.

Architecture

The LIST off-line analyzer is composed by a web-based single-page application and a backend server. The LIST

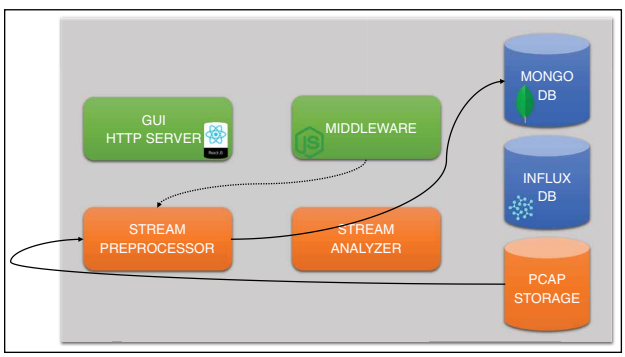

FIGURE 2. Preprocessing.

build scripts generate a Docker[5] container that includes all the server-side components. Therefore, it can be run locally on the user's machine or deployed as a dematerialized instance in any cloud provider. We provide that docker container in an online registry, as well as an instance running on a public cloud.

The block diagram of the tool can be seen in **Fig. 1**. The user interface is written in JavaScript and based on React.[6] The middleware that coordinates the workflows is based on node.js.[7] The analyses of the data are made by the LIST Core libraries, which are developed in C++.

There are two separate databases. We use MongoDB[8] for nontime-related data, ranging from user data to static stream information; in addition, we use InfluxDB[9] for time-series data. This database is used to store information about every packet on the stream.

When the user uploads a file, it is transferred from the browser to the pcap storage. Then, the middleware instructs the stream preprocessor to analyze the pcap file (**Fig. 2**). The preprocessor parses all the packets in the pcap file, detects the individual streams, and uses heuristics to identify the kind of essence and to detect the streams' characteristics, such as frame rate, resolution, audio sampling rate, and so on. This information is stored in MongoDB.

Next, the middleware instructs the stream analyzer to process all the essence (**Fig. 3**). The analyzer reads the stream information from MongoDB to know what parameters to use, then parses all the packets from the pcap file and stores all time-related information in InfluxDB. The video frames and the audio streams are stored in the pcap storage, ready to be served to the browser. If the detection heuristics do not correctly identify the stream characteristics, the user can change the settings and the stream analysis process is run again for the specific streams.

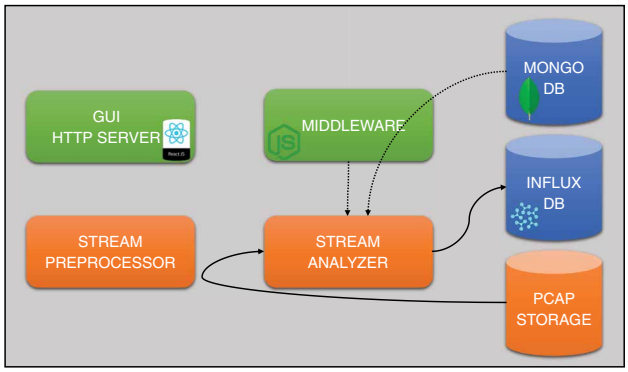

FIGURE 3. Processing.

Heuristics

ST 2110 streams are not self-descriptive, that is, unlike some media formats, they do not transport information about the media characteristics, such as resolution, frame rate, number of audio channels, and so on. For that reason, ST 2110-10 determines that Session Description Protocol (SDP) should be used to provide that information. However, when monitoring random streams on the network, it may be difficult to access the corresponding SDP file. For that reason, LIST implements a set of heuristics that allows it to determine most of the parameters of the streams.

The first step is to distinguish among video, audio, and ancillary data. Each kind of stream exhibits specific patterns for two characteristics: the variation of the timestamp difference between consecutive packets, and the number of packets between packets with the marker bit set. This makes it relatively straightforward to segregate among media kinds.

The second step is to infer media-specific information. For ancillary data, only the rate needs to be calculated. This can be easily computed by measuring the RealTime Protocol (RTP)[10] timestamp delta between packets of consecutive frames and taking into account that the RTP clock runs at 90 kHz.

For audio streams, LIST assumes that the audio sampling rate is 48 kHz. Then, measuring the RTP timestamp delta, it calculates the packet time. Finally, using the packet size, it calculates the number of channels and the bit depth. Since several combinations of the latter can result on the same packet size, LIST chooses the most common combination (e.g., two channels, 24 bits; over three channels, 16 bits).

Finally, the heuristics for video streams. LIST calculates the frame/field rate based on the RTP timestamp deltas, like for ancillary data. Then, it analyzes the ST 2110-20 headers to detect additional information. The presence of a field marker implies that this is an interlaced stream. The maximum line number signaled in the header indicates the height of the frame. The offset and length provide information about the frame width. The pixel format is assumed to be YCbCr.

Practical Test Results with EBU LIST

As a practical test of our algorithms and the software, we performed a high-precision capture and analysis of a 1080p50 ST 2110-20 stream generated by the Embrionix emSFP with our lab setup. The stream capture was done by a high-performance server with the NIC synced to Precision Time Protocol (PTP). The analysis was run on the web appliance of the EBU LIST, available at http://list.ebu.io. The results feature a number of measurement graphs described later.[11]

The graph shown in **Fig. 4** is derived from the PTP messages exchanged between the sender device and the master clock. By emulating the behavior of the sender PTP clock and based on the timestamps and payload of the PTP messages, we can derive the corrections that were made to the internal clock of the sender by its PTP stack. This analysis excludes any smoothing done by the internal Proportional-Integral Controller or any other mechanism used by the sender's clock to smooth adjustments.

FIGURE 4. Sender's PTP offset graph.

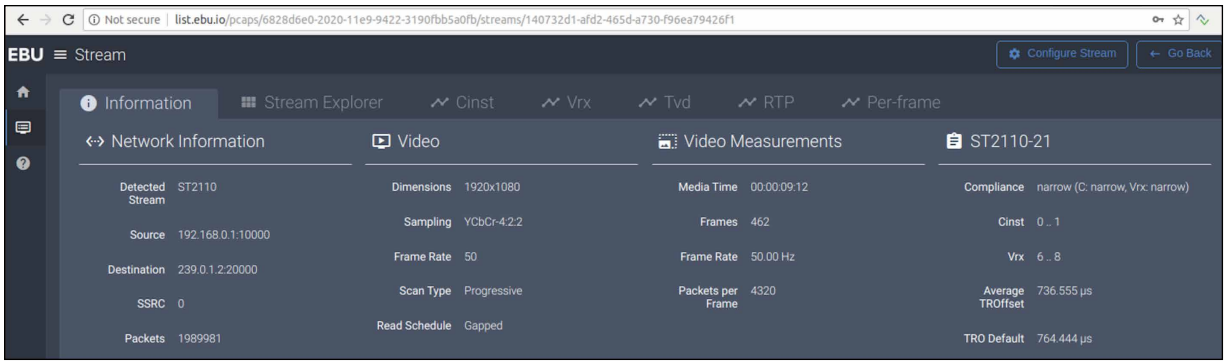

FIGURE 5. Stream information.

FIGURE 6. Stream explorer.

Figure 5 shows an overview of the stream information. We can confirm that the video format was identified correctly and that some additional information was extracted or calculated. The EBU LIST detected that the read schedule is gapped and that there are 4320 packets per video frame. Regarding ST 2110-21 compliance, we can see that C_{INST} and VRX varied between 0 and 1, and 6 and 8, respectively, which classifies the stream as compliant with the narrow profile. We can also see that the average TR_{OFFSET} is slightly below the $TRO_{DEFAULT}$ for this format.

The graph shown in **Fig. 6** demonstrates the stream explorer, which allows us to see each frame of the captured stream. It is apparent that the capture started halfway through one frame, and we can also confirm that the stream was correctly captured, with no visible impairments on the image. In **Fig. 7**, we can see the graphic representation of the C_{INST} analysis. On the top, the histogram shows the distribution of the C_{INST} values and we can see that more than 60% of the time the C buffer was empty, while it had one packet in the remaining time. This can be confirmed in the timeline view, on the bottom.

Figure 8 shows two graphs. The one on the top is the calculated VRX over time, assuming that the first packet of each frame should arrive at the ideal time. This ideal time is the alignment point of a frame plus the default TR_{OFFSET}, as defined in ST 2110-21. On the other

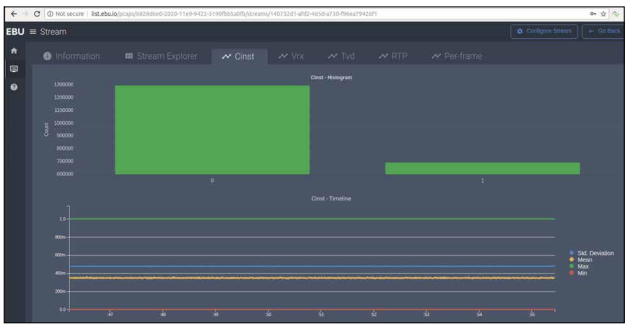

FIGURE 7. C_{INST} analysis.

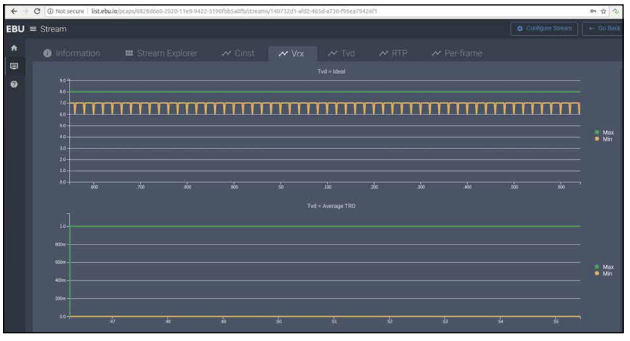

FIGURE 8. VRX analysis.

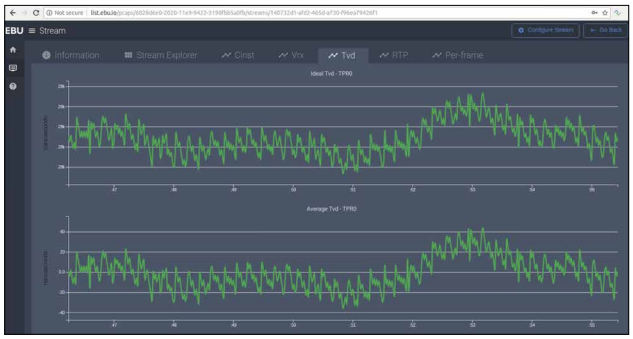

FIGURE 9. T_{VD} analysis.

hand, the second VRX graph is plotted with TR_{OFFSET} = measured/averaged. The difference between the two graphs can be seen as a "DC component," in this case of around six packets. In fact, if we go back to **Fig. 5**, we see that, on average, the first packets arrive 27.8885 µs earlier than they should. If we take into account that for this format, the number of packets per frame and read schedule, the interpacket spacing is 4.4444 µs, we can easily derive that we should have 27.8885/4.4444 ≈ 6 packets in the buffer at all times.

Figure 9 shows the variation of T_{VD}, the arrival time of the first of each frame. Again, the graph on top takes the ideal time as the reference, while the one on the bottom is based on the average arrival time, as if a DC component had been removed. This shows us that the maximum variation of T_{VD} is just above 40 ns, which is extremely good.

In **Fig. 10**, we can see the analysis of the RTP header timestamps. Starting at the bottom, we see the difference in timestamps between consecutive frames. The graph shows a constant difference of 1800 ticks, which is correct for a media clock at 90 kHz and a frame rate of 50. The middle graph shows the difference between the actual RTP timestamp and the calculated RTP timestamp for a frame captured at that time, according to the rules defined in ST 2110-10, § 7. Finally, the top graph shows the difference between the arrival time of the packet and what would be the

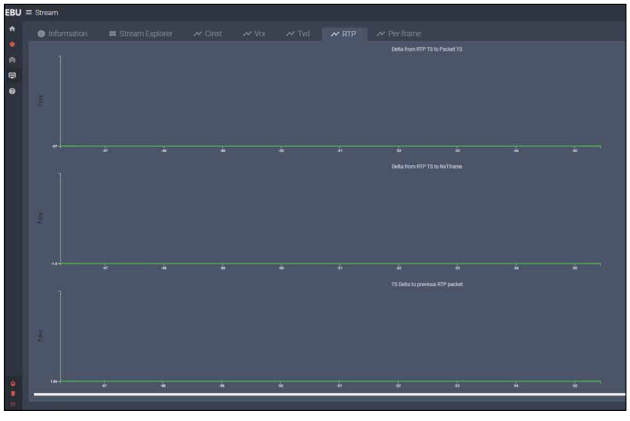

FIGURE 10. RTP timestamp analysis.

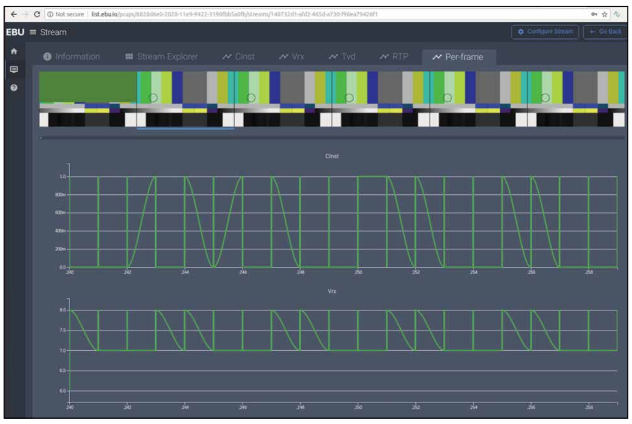

FIGURE 11. Per-frame C_{INST} and VRX analysis.

RTP timestamp for that time. The constant difference of −67 ticks (744.444 µs at 90 kHz) is consistent with the average T_{VD} shown before (736.555 µs) and within the resolution of 1 tick (11.11 µs). **Figure 11** shows the same information we can see on the previous C_{INST} and VRX information tabs, but shown for the duration of each frame, independently.

Finally, **Fig. 12** shows the analysis of an ST 2110-30 stream, including basic media information, a time-stamped delay factor (TS-DF)[12] graph, and a media player rendering the payload into an audio clip. The TS-DF graph shows the maximum value prescribed by AES67[13] as a yellow line and the actual measured values in green.

Conclusion

LIST is an open-source project led by the EBU in active development with contributions from our members. New features are under way. One of the most relevant is arguably a prototype of a live analyzer, which can capture and analyze traffic in realtime. This prototype was first demonstrated at International Broadcasting Convention (IBC)'18.

LIST uses Azure DevOps[14] as the Git repository, planning tool, and continuous integration (CI) platform. In addition, a public GitHub[15] repository is

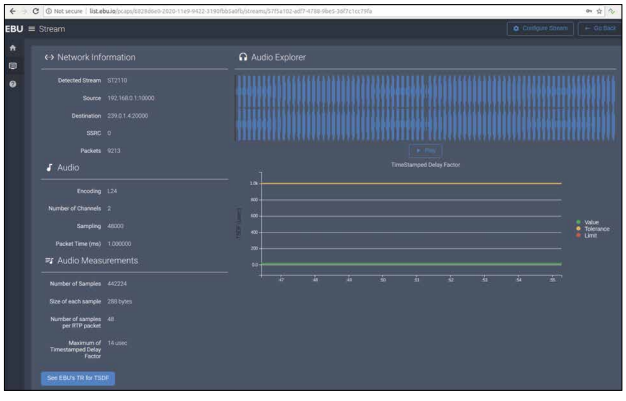

FIGURE 12. ST 2110-30 stream analysis.

frequently synchronized with the LIST master repository,[16] allowing open collaboration with external contributors, including the acceptance of pull requests. The CI system runs continuously, testing every commit and automatically pushing new releases to the Docker registry[17] and updating the online instance on the public cloud. This project has been playing a fundamental role in the industry, showing that it is possible to initiate a transition away from specialized and closed platforms into systems based mostly on off-the-shelf hardware, paving the way to the holy grail of the dematerialized[18] facilities.

References

1. European Broadcasting Union (EBU), "Live IP Software Toolkit (LIST)." [Online]. Available: http://tech.ebu.ch/list
2. SMPTE, ST 2110 Suite of Standards. [Online]. Available: https://www.smpte.org/st-2110
3. tcpdump. [Online]. Available: https://www.tcpdump.org
4. Wireshark. [Online]. Available: https://www.wireshark.org/
5. Docker. [Online]. Available: https://hub.docker.com/
6. React, "A JavaScript Library for Building User Interfaces." [Online]. Available: https://reactjs.org/
7. Node JS. [Online]. Available: https://nodejs.org/en/
8. MongoDB. [Online]. Available: https://www.mongodb.com/
9. InfluxDB. [Online]. Available: https://www.influxdata.com/
10. Real Time Protocol, "RTP: A Transport Protocol for Real-Time Applications." [Online]. Available: https://www.ietf.org/rfc/rfc3550.txt
11. W. Vermost, I. Kostiukevych, P. Ferreira, "Essential Measurements and Compliance Evaluation in SMPTE ST 2110 Based Facilities Using Commodity of the Shelf IT Hardware," in *Proc. NAB Broadcast Eng. Inf. Technol. Conf. 2019*, Las Vegas, Apr. 2019.
12. EBU-UER, "A Proposed Time-Stamped Delay Factor (TS-DF) algorithm for measuring Network Jitter on RTP Streams," Jan. 2010. [Online]. Available: https://tech.ebu.ch/docs/tech/tech3337.pdf
13. Audio Engineering Society. [Online]. Available: http://www.aes.org/
14. Azure DevOps. [Online]. Available: https://azure.microsoft.com/en-us/services/devops/
15. Github. [Online]. Available: https://github.com/
16. LIST on Github. [Online]. Available: https://github.com/ebu/pi-list
17. LIST on Docker. [Online]. Available: https://hub.docker.com/r/ebutech/pi-list/
18. LIST as a Dematerialized Version. [Online]. Available: http://list.ebu.io

About the Authors

Ievgen Kostiukevych gathered nearly a decade of experience with AoIP integration and solution architecture before joining the EBU Technology & Innovation Team as an expert on media over IP, with an emphasis on audio over IP and the AES67/SMPTE ST 2110-30 interoperability challenges. Kostiukevych is a member of SMPTE and the Audio Engineering Society.

Willem Vermost is the topic lead on the transition to IP-based studios. With 20 years of experience in broadcast, he is an expert and project manager of international strategic, expert groups, and events. Vermost has a master's degree in electronic engineering and a master's in applied computer science. He worked on various proofs-of-concept, including the multiple award-winning Vlaamse Radio en Televisie (VRT) Live IP Proof of Concept (POC) and started the EBU LIST project that has grown to an international project.

Pedro Ferreira is a software engineer, consultant, and trainer. He is the lead developer of the EBU LIST project and a member of the EBU Academy Faculty. His interests, which range from high-performance networking and media processing software to cloud-based microservices and IP-based production systems, are at the core of his current research. Ferreira has an MSc in telecommunications and computers and more than 20 years of experience in leading development teams in the broadcasting industry. He has founded several companies and led the development of the first commercially available Material Exchange Format (MXF) software toolkit, as well as many ingest and capture systems in production around the globe.

Presented at the SMPTE 2018 Annual Technical Conference & Exhibition, Hollywood, CA, 22–25 October 2018.