# An architecture for time-critical IP broadcasting in the cloud

Miguel Poeira*, Pedro Santos†, Alexandre Ulisses‡, Daniel Costa§, Pedro Ferreira¶ and Rui Amor‖
MOG Technologies
Email: *miguel.poeira@mog-technologies.com, †pedro.santos@mog-technologies.com, ‡alexandre.ulisses@mog-technologies.com,
§daniel.costa@mog-technologies.com, ¶pedro.ferreira@mog-technologies.com, ‖rui.amor@mog-technologies.com

***Abstract*—Today's live remote TV production is still based on very expensive and specific equipment. Besides that, covering big events is a complex challenge, both from technological and logistic point of view. In this paper, we present a high-level architecture to virtualize live IP production to the cloud, in order to provide production teams with more flexible tools at reduced cost. While we outline the basic components for a modular distributed system, we also address some problems in the time-critical domain.**

***Keywords*—cloud computing, time-critical, soft real-time, broadcasting, TV production**

## I. Introduction

Live remote TV production, due to its distributed nature, requires broadcasters to deploy equipment and human resources to different places, severely increasing production's costs. With the evolution of virtualization technologies and the efforts developed in order to provide solutions based on elastic, adaptive, controllable cloud environments, it is possible to consider this technological stack for time-critical scenarios, such as a live TV production.

In this paper, we present a high-level architecture to virtualize remote production using a cloud-based infrastructure, outlining the main components for a distributed system that takes advantage of that infrastructure.

## II. State-of-the-art

Today's most used professional broadcasting solutions for live coverage rely on Outside Broadcasting vans, production trucks that are both expensive and complex to use [1], and satellite links or 3G/4G networks, depending on the available equipment and the venue location. Live remote coverage of an event can be as simple as one camera recording an interview, or a complex scenario, such as covering the Olympics. This wide range of scenarios, from small to big venues, with the possibility to need to cover more than one location at the same time, it is not only a technological challenge, but also a logistic challenge.

### A. Soft Real-time Applications

A time-critical application is a specific "system or mode of operation in which computation is performed during the actual time that an external process occurs, in order that the computation results can be used to control, monitor, or respond to the external process in a timely manner" [2].

While operating a soft real-time application, such as the one we address here, it is important to understand that, although the deadline for a given task may be missed, doing it will degrade the Quality of Service (QoS), upon which professional broadcasters set very hard constraints. In order to achieve the desired QoS, broadcasters usually rely on specific hardware, which is very expensive to buy, operate and maintain [3].

### B. Internet Media Consumption

Media consumption, via the Internet, is largely established and has a huge viewer base, such as Youtube and Twitch, services that may peak at more than 1 Tbps [4]. These platforms are used mainly for non-professional broadcasting. Any user may set up an account and start a live broadcasting session right away, as they are served as a Software-as-a-Service system. The problem is that those are closed platforms, in the sense that it is not possible to freely control the environment on which the system runs. Likewise, by only being able to stream to closed platforms, it becomes restrictive, both in a technological and business sense.

In a professional broadcasting, there is the need to control and monitor the whole system's life-cycle, from design, to provisioning and run-time monitoring. This means that the running environment must lay on a Platform-as-a-Service architecture that supports real-time QoS specification, dynamic Service-Level-Agreement negotiation, on-demand resource provisioning and QoS event monitoring [5]. That said, it is not feasible to use already market-established platforms, such as Youtube, to perform professional TV production.

## III. Scenario

A scenario to handle live remote TV production for the professional broadcasting industry, where operations are still very tied to hardware processing [6], requires the processing of high-volume data within a very limited time frame, which usually requires very specialized hardware. Nevertheless, the importance of delivering software-based processing, which was previously done by hardware specific material, is a key aspect to avoid vendor lock-in [7], [8].

### A. Considerations

Since we are proposing a virtualized environment to run a scenario where reliability is an important part of the industry's business, it is important to keep the same quality of experience

that end-users already have. This means that the environment where these scenarios are deployed needs to be well-controlled and deterministic, in a sense that assures repeatability of processes' results. That predictability is what defines a real-time system [9].

Because we are talking about a soft real-time system, where low latency is a key aspect to whole system's success, it is important to recognize Workflow Management Systems, such as SWITCH [10], [11], as an important part of the ecosystem that is needed to support these kind of scenarios. This means that not only it is important to understand the conditions within the applications' deployment, but it is also necessary to understand how to monitor and dynamically manage the different subsystems that cooperate for the system's success.

In a data-driven scenario as the one we present, it is very important to be able to operate in an active manner. This means being reactive upon data imperfections (delayed, missing and out-of-order data) and pro-active to seamlessly reconfigure without human intervention.

The keystone of our foundational architecture is the Reference Architecture of Joint Task Force on Networked Media [12].

### B. Scope

TV production world is very broad and the endless different scenarios that broadcasters face each day make it as interesting as complex. With so many different interactions and workflows, we believe that there is no one-size-fits-all solution. So, we are defining the scope of this work as an cloud's distributed system architecture for live IP production, where producers may switch cameras from a given number of input streams. This way, we recognize the diversity that exists in broadcaster's world, but focus our attention in live scenarios.

We define out of scope how contents are delivered to and from our cloud system, as we believe both topics are broad enough and too complex to address within our work's scope.

## IV. High-level Architecture

In order to virtualize TV live production, we propose a cloud-based distributed system, as well as an IP-based content transport fabric. This system should be able to receive a given number of input streams, generate control versions of that streams, enable video switching and deliver the output in several different formats. This converged workflow enables teams to produce once, deliver everywhere [13].

By taking advantage of cloud's benefits, we are proposing an architecture that is:

- Built on **modular** components and blocks, that can be re-utilized to build new nodes and functionalities.
- **Expansible**, which means that it can be adapted to new scenarios on-the-fly.
- **Scalable**, that is, operation's needs are not bound by hardware stock availability or ease of deployment.
- Based on **state-of-the-art technology**, battle-tested by other industries.
- **Compliant** with industry standards and compatible with already-established components.
- **Collaborative**, because it enables people to easily collaborate on the same project, wherever they are located.
- Enables fast, transparent **updates and bug fixes**, which lets the production team abstract on technical details.

### A. Assumptions and Requirements

As we designed this architecture, we defined some requirements that should be met. These are hard constraints in the sense that not being able to ensure them may result in system's failure.

We assume the knowledge of Workflow Management Systems and redundancy and high-availability techniques, in order to understand how the system should keep itself up and running upon failure. Also, the deployment environment must provide built-in mechanisms to provide resiliency against "imperfections" on data, including missing and out-of-order data.

We also assume that the network for the deployed environment provides the required resources in order to ensure the QoS that enables the system to work as expected in a professional broadcasting soft real-time scenario. It is also out of scope of our work how the system should be deployed and in what ways it can be configured and personalized. We believe that there is a lot of work to do in this field, including the use of visual human-friendly tools to deploy a whole broadcasting infrastructure.

### B. Components

In this section, we present the five basic components that compose our architecture for live IP production. These are the foundational components, which we believe that will be present in the majority of the scenarios.

Each component is a "Node" on the network and it may be virtual or physical. Each node is based on a layered structure proposed by the Networked Media Open Specifications (NMOS) [14] and it has a specific purpose, which is well-described at section V and summarized hereunder:

- Input Distributor is responsible to receive the incoming streams and distribute them to other nodes of the system.
- Proxy Transcoder generates low-resolution control streams to be presented in a Web GUI, which will enable live IP production.
- Video Switcher switches from the input streams, the desired output.
- Output Transcoder serves the output from the Video Switcher node to "outside" of the system boundaries.
- Business Logic orchestrates the whole system and provides a REST API to control it.

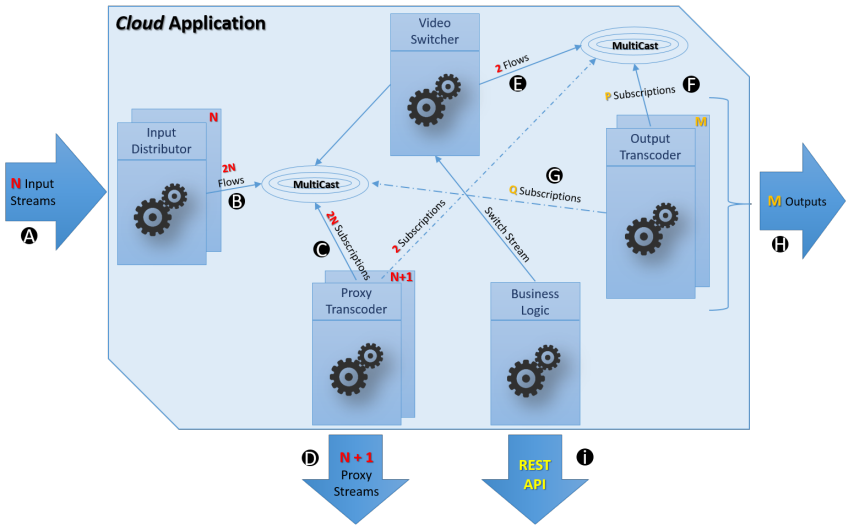


Fig. 1: Components and its interactions

### C. Data Flow

The transport of audio and video (AV) through the system is a critical issue, not only because it is a challenge to keep delay to a minimum, but also because of timing and synchronization issues. So, in our architecture we took two major decisions: to process AV as two separate entities and deliver them using multicast.

Processing AV as separate entities gives the flexibility to only address entities that are relevant within a given context. This means that, from a given source, it is possible to easily discard the video flow when there is the need to only receive the audio, for instance.

All AV data will be served in our system using multicast delivery. By achieving "one-to-many delivery" it is possible to abstract nodes from time-consuming operations (such as keeping sessions) and keep latency to a minimum. We suggest different multicast addresses for different flows, as it makes it much more easier to filter the content to subscribe.

When a node fails, for instance, it should be possible to get another up and listening to the same multicast address, without the need to reconfigure the other nodes in the system. Moreover, nodes don't need to be aware of failures in other nodes.

Furthermore, changes in a deployed application, such as adding one more output format, are straightforward and transparent to the end-user as one just needs to configure a new Output Transcoder node and put it online.

### D. Node Communication and Layered Architecture

Each node has the layered architecture that NMOS proposes, where the high-level layers are common to all nodes and allow interoperability, and the lower level layers are responsible for the specific content processing of that node and its control logic, which handles requests and updates.

All communication and exchange of information between nodes is done using the HTTP Transactions API that NMOS proposes, using the Business Logic node as the centralized server for discovery and registration purposes.

## V. Distributed System's Components

In this section we provide more details about the nodes that compose the proposed architecture (Fig. 1). Although we are giving more details about how the nodes should interoperate, we are keeping this description as high-level as possible, mainly because we want to stay agnostic about how to deliver content into the system and how to distribute its output. The same applies to how to transport media content inside the system. These are operational decisions, which should be evaluated within a given context and a well-defined purpose.

### A. Input Distributor

The Input Distributor is the entry-point of the distributed system. It must de-multiplex the input stream (if necessary) and multicast the resulting flows, so that other nodes may subscribe them.

Each Input Distributor node is responsible for a single input stream. That means that the number of Input Distributor nodes that exist is, at any given point of the system's lifetime, the same as the input streams that the system is receiving (Fig. 1A).

A typical input stream is composed of video and audio. Because of the de-multiplexing that is done and the data flow that was adopted, from N input streams 2N flows are derived, which are multicasted separately (Fig. 1B).

It is not defined how the routing should be done, as the operations team should evaluate in a case-by-case basis.

### B. Proxy Transcoder

Each Proxy Transcoder node is responsible to serve a low-resolution control stream that may be consumed by an external client or service. Proxy Transcoder has a 1:1 relationship with

Input Distributor, which means that each Proxy Transcoder is responsible for the flows of one Input Distributor (Fig. 1C).

The same way, one Proxy Transcoder should also subscribe the resulting flows from the Video Switcher node, in order to provide a control version of the output result. That means that for N input streams, we get N+1 deployed nodes (Fig. 1D).

### C. Video Switcher

The Video Switcher node is unique in the system and will multicast the resulting AV pair of flows of the switching operation (Fig. 1E). Because of that, it needs to know all multicast addresses that Input Distributors are feeding. Video switching, by itself, it's a pretty complex research topic, which [15] and [16] address in much more detail, and would require, alone, a dedicated paper.

### D. Output Transcoder

Each Output Transcoder node must subscribe multicast addresses that are being fed by Video Switcher (Fig. 1F), or one Input Distributor (Fig. 1G), and deliver them to outside of the application, with a specific format and for a specific platform. This means that, although the output of the system is editorially unique, the system may have more than one Output node, in order to deliver different formats. So, the number of nodes required is equal to the number of different outputs that may be produced (Fig. 1H).

As it was defined before, transporting audio and video in separate gives flexibility, given that it allows to serve as output an audio-only version (for instance) and thus it is only needed to subscribe the audio flow (and completely ignore the video one).

By the same logic, an Output node that delivers the output stream with the same characteristics of the Input's may exist, in order to enable cascade scenarios.

### E. Business Logic

The Business Logic node is the brain of the whole system. It is responsible for the orchestration of the other nodes, playing a three-folded role. It enables other nodes to register its capabilities, assets and tasks; by keeping a centralized database, it also enables nodes to query and discover all other nodes and their associated functions and flows; at the same time, it serves a REST API that enables end-users to control and monitor the whole system (Fig. 1I).

## VI. Conclusion

This paper has described a high-level architecture to operate an IP broadcasting session in the cloud. The presented distributed system is designed to cope with industry's standards and current operations, while addressing the flexibility required to cope with the multitude of possible scenarios.

The idea of operating a broadcasting session through the cloud is a shift in the media and broadcasting industry. It gives production teams the freedom to create new and different workflows, more flexible, based on on-demand, pay-per-use, infrastructures, which not only enable resources to be better managed, but also provides organizations with a shorter time to market.

There is still a lot to study in the future, namely how the system should behave and adapt during runtime resource provisioning and how it is possible to scale this architecture to a full IP studio.

## Acknowledgment

This project was supported by European Union's Horizon 2020 program under grant agreements No. 643963 (SWITCH).

## References

[1] J. Jachetta, "Ip to the camera-completing the broadcast chain," in *Annual Technical Conference & Exhibition, SMPTE 2014*. SMPTE, 2014, pp. 1–29.

[2] J. Radatz, A. Geraci, and F. Katki, "Ieee standard glossary of software engineering terminology," *IEEE Std*, vol. 610121990, no. 121990, p. 3, 1990.

[3] S. Gogouvitis, K. Konstanteli, S. Waldschmidt, G. Kousiouris, G. Katsaros, A. Menychtas, D. Kyriazis, and T. Varvarigou, "Workflow management for soft real-time interactive applications in virtualized environments," *Future generation computer systems*, vol. 28, no. 1, pp. 193–209, 2012.

[4] K. Pires and G. Simon, "Youtube live and twitch: a tour of user-generated live streaming systems," in *Proceedings of the 6th ACM Multimedia Systems Conference*. ACM, 2015, pp. 225–230.

[5] M. Boniface, B. Nasser, J. Papay, S. C. Phillips, A. Servin, X. Yang, Z. Zlatev, S. V. Gogouvitis, G. Katsaros, K. Konstanteli *et al.*, "Platform-as-a-service architecture for real-time quality of service management in clouds," in *Internet and Web Applications and Services (ICIW), 2010 Fifth International Conference on*. IEEE, 2010, pp. 155–160.

[6] A. S. Davies, "The Silver Lining: Utilizing Cloud Computing in Broadcast Applications," *SMPTE Motion Imaging Journal*, vol. 120, no. 2, pp. 30–36, mar 2011.

[7] J. Footen and M. Ananthanarayanan, "Service-Oriented Architecture and Cloud Computing in the Media Industry," *SMPTE Motion Imaging Journal*, vol. 121, no. 2, pp. 22–30, mar 2012.

[8] N. P. J.D. Mitchell, M. Thorp, C.K.P. Clarke, "Making TV Programmes using IP Networks," *BBC R&D White Paper WHP281*, 2014.

[9] L. Abeni, G. Buttazzo, G. Lipari, and M. Caccamo, "Soft real-time systems: predictability vs. efficiency," *Massachusetts institute of technology-2005*, 2005.

[10] K. Evans, J. Trnkoczy, G. Suciu, V. Suciu, P. Martin, J. Wang, Z. Zhao, A. Jones, A. Preece, F. Quevedo, D. Rogers, I. Spasić, I. Taylor, V. Stankovski, and S. Taherizadeh, "Dynamically reconfigurable workflows for time-critical applications," in *Proceedings of the 10th Workshop on Workflows in Support of Large-Scale Science - WORKS '15*. New York, New York, USA: ACM Press, nov 2015, pp. 1–10.

[11] Z. Zhao, P. Martin, J. Wang, and A. Taal, "Developing and Operating Time Critical Applications in Clouds: The State of the Art and the SWITCH Approach," *Procedia Computer ...*, 2015.

[12] European Broadcasting Union, Society of Motion Picture and Television Engineers, and Video Services Forum, "Joint Task Force on Networked Media - Reference Architecture v1.0," 2015. [Online]. Available: http://jt-nm.org/RA-1.0/JT-NMReferenceArchitecturev1.0 150904 FINAL.pdf

[13] P. J. Cianci, *Technology and Workflows for Multiple Channel Content Distribution: Infrastructure implementation strategies for converged production*, 1st ed., S. M. Weiss, Ed. Focal Press, 2009.

[14] A. M. W. Association, "Networked Media Open Specification," 2016. [Online]. Available: https://github.com/AMWA-TV/nmos

[15] T. Kojima, J. J. Stone, J.-R. Chen, and P. N. Gardiner, "A Practical Approach to IP Live Production," *SMPTE Motion Imaging Journal*, vol. 124, no. 2, pp. 29–40, mar 2015.

[16] A. Rawcliffe, "Covering the Glasgow 2014 Commonwealth Games using IP Studio," *BBC R&D White Paper WHP289*, 2015.