A Scalable WebRTC Platform based on Open Technologies

Pelayo Nuño, Francisco G. Bulnes, Juan C. Granda, Francisco J. Suárez, Daniel F. García
Department of Computer Science
University of Oviedo
Campus de Gijón, Spain
{nunopelayo,bulnes,jcgranda,fjsuarez,dfgarcia}@uniovi.es

***Abstract*—WebRTC is a joint effort of the IETF RTCWeb and the W3C WebRTC working groups for real time communications through the web. This paper proposes a WebRTC platform for interactive multimedia communication. The platform is based on open technologies and provides an ubiquitous conferencing service which may be used to support multi-point e-learning/e-meeting activities. The platform also takes advantage of the potential of cloud computing to achieve scalability. The iterative design of the platform architecture is exposed and some preliminary results in scalability analysis are outlined.**

***Index Terms*—Real-Time Communicactions, WebRTC, Protocols, Open standards, Multimedia, Cloud computing, System integration**

I. Introdution

Real-Time Communication (RTC) services are usually built around two planes. The signaling plane is used to locate endpoints and establish and tear down media sessions. The data plane is responsible for delivering multimedia data, such as audio and video.

WebRTC [1] is a set of JavaScript APIs supporting the data plane of RTC services in web browsers. It provides peer-to-peer communication with audio, video and other real-time data types by using standard protocols. Desktop and mobile web browsers widely support WebRTC, making the development of real-time communication services possible without any additional software. WebRTC also supports royalty-free codecs, secure data transport with the Secure Real-time Transport Protocol (SRTP), inter-stream synchronization, NAT traversal mechanisms and so on. WebRTC lets the programmer choose any signaling mechanism, so it is possible to use existing signaling protocols to interoperate with legacy telephony and voice over IP systems.

WebRTC includes mechanisms for Interactive Connectivity Establishment (ICE), allowing communication between users even if they are behind firewalls or one or more layers of NATs. However, WebRTC is only aimed at point-to-point communication between two browsers, so multi-point communication is out of the scope of the standard. Multi-point Control Units (MCUs) are then required to enable conferences with more than two endpoints. This brings the possibility to deploy various conferencing models with WebRTC depending on the signaling topology, ranging from star topologies to highly distributed multi-point topologies.

This paper proposes a platform for interactive multimedia communication based on WebRTC and other open standards and technologies. The platform provides an ubiquitous and scalable conferencing service, supporting multi-point activities such as e-learning sessions, e-meetings and product demonstrations.

The remainder of the paper is organized as follows. In Section II, related work on existing WebRTC implementations is discussed. The architecture of the proposed platform is described in Section III. Details of the scalability evaluation and some preliminary results are shown in IV. Finally, Section V contains the concluding remarks and outlines future work.

II. Related Work

WebRTC has several outstanding advantages over other alternatives in real time interactive multimedia communications. It is supported by almost all modern browsers, so users do not need to install or download any additional software. In [2] a deep analysis of the standard can be found, including details of the available communication topologies and performance metrics.

WebRTC has contributed to successful applications in the fields of tele-health [3], [4], online assistance in e-businesses [5], distance learning [6]–[8], all of them following the native peer-to-peer approach (P2P). WebRTC does not support IP multicast, so multi-point scenarios [9] should be supported by multi-unicast deployments, using MCUs to forward media streams between participants. Examples of WebRTC MCUs can be found in [10]–[12], providing services such as media transcoding and mixing.

Integration of videoconference systems on the cloud has been an important research topic in the last years [13], [14]. In [15] a cloud videoconference system for mobile devices, where the cloud resources improve quality and scalability, is described. In case of multi-point videoconferences, advantages and challenges of cloud MCUs are shown in [16]. Cloud versions of MCUs are also proposed in [17], [18]. Finally, optimization of resources in videoconference systems over multi-provider hybrid clouds is presented in [19].

This research has been partially funded by the Spanish National Plan of Research, Development and Innovation under the project MINECO-15-TIN2014-56047-P

978-1-5386-4599-4/18/$31.00 © 2018 IEEE

III. Platform Architecture

Similarly to other RTC services, the proposed platform defines a data plane and a signaling plane. The signaling plane follows a star topology and uses the Session Initiation Protocol (SIP) to negotiate media sessions between participants, whereas RTP is used for data plane. These planes are independent, meaning that SIP and RTP packets can travel through the network following different routes. The planes can be even established between different entities. For instance, parameters of a multimedia session can be negotiated in a SIP server and data streams can be exchanged through an MCU.

Confidentiality of communications is achieved only if both the signaling and data planes are encrypted. The data plane is encrypted by using the secure profile of RTP. For the signaling plane, SIP does not provide encryption, and secure transport protocols, such as TLS, must be used.

Several technologies have arisen around VoIP that can be aplied to every real time communication service, including collaborative and e-leaning platforms. Some of them are analyzed below.

It is quite feasible to develop a telephone service with the SIP/RTP protocols and a VoIP PBX (*Private Branch eXchange*) such as Asterisk. Asterisk allows communication between terminals (extensions) with one or more lines connected to the Public Switched Telephone Network (PSTN). This way, terminals can establish calls with other terminals in the PTSN. Other funcionalities provided by Asterisk are voicemail, call on hold, interactive voice response (IVR), etc.

Asterisk uses SIP, among other protocols, for call signaling. It behaves as a B2BUA (*Back-to-Back User Agent*) in a VoIP call, operating between the participants both in the signaling and data planes. Configuring Asterisk for data to flow directly between participants (*pass-through* mode) is also posible. In this case, Asterisk operates between participants only in the signaling plane, but multimedia data flows between participants directly.

The functionality offered by Asterisk can be extended using two APIs. First, AMI (*Asterisk Manager Interface*) can be used to control Asterisk from other applications or services. This API allows to initiate calls, configure voicemail, redirect calls, etc. Second, AGI (*Asterisk Gateway Interface*) can be used to trigger other applications when events in current calls arise. A version of this API, named FastAGI, allows triggering applications in remote machines through TCP *sockets*.

A. Initial Design

The construction of the platform follows and iterative process, with new designs adding flexibility and reliability, but also complexity, to the platform.

The initial and simplest platform design is shown in Fig. 1. This design supports multi-point conferences between participants. An Asterik PBX plays the role of a SIP signaling server and gives access to the conference. Participants make calls to the conference SIP URI to join it using a WebRTC client with SIP signaling. These calls are forwarded to a Kurento server

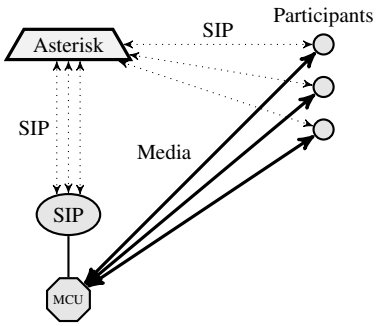

Fig. 1. Initial architecture design.

acting as an MCU that merges audio and video data streams of all participants. Kurento can not use SIP directly, so a SIP wrapper must be developed to process calls on its behalf. The wrapper is a server that interacts with Kurento through its API and with the other entities of the platform within the SIP signaling plane. Since a terminal has to register in Asterisk in order to receive calls, the Kurento SIP wrapper must be registered in Asterisk.

The Kurento SIP wrapper has full control of the conference. It must support multiple concurrent SIP calls, one per participant. It must reject calls from non authorized participants and even exclude disturbing participants during the conference. The inclusion and exclusion of participants also implies the reconfiguration of the data plane. The MCU must send multimedia streams only to active participants.

Although Asterisk may also act as an MCU, it is not as efficient as a dedicated MCU such as Kurento. In fact, Kurento is more specialized in multimedia data processing. Besides, a design with a signaling server and a dedicated MCU results in more flexible, increasing modularity and scalability.

Figure 2 shows the signaling messages needed to join a conference. Asterisk is configured in *pass-through* mode for the data to flow directly between the participant and the MCU.

The participant must call `sip:conf@domain.com`, the URI conference in the example. This URI is associated with an extension that must be configured in the Asterisk dialplan to point to the Kurento SIP wrapper, so the call is forwarded to the MCU. The participant sends a SIP INVITE message to initiate the call. If the call initiation successes, the call is established between the participant and Asterisk. Thus, Asterisk may send the participant the call dial tone or information about errors (e.g. an invalid extension).

Once the call between the participant and Asterisk has been established, the latter makes a call to the Kurento SIP wrapper. When the wrapper receives the INVITE message from Asterisk, it decides if the participant is authorized to join the conference or nor, so the authorization is done in two steps: in Asterisk (using the dialplan) and in the SIP wrapper (accepting or rejecting incoming calls).

If the participant is authorized to join the conference, the

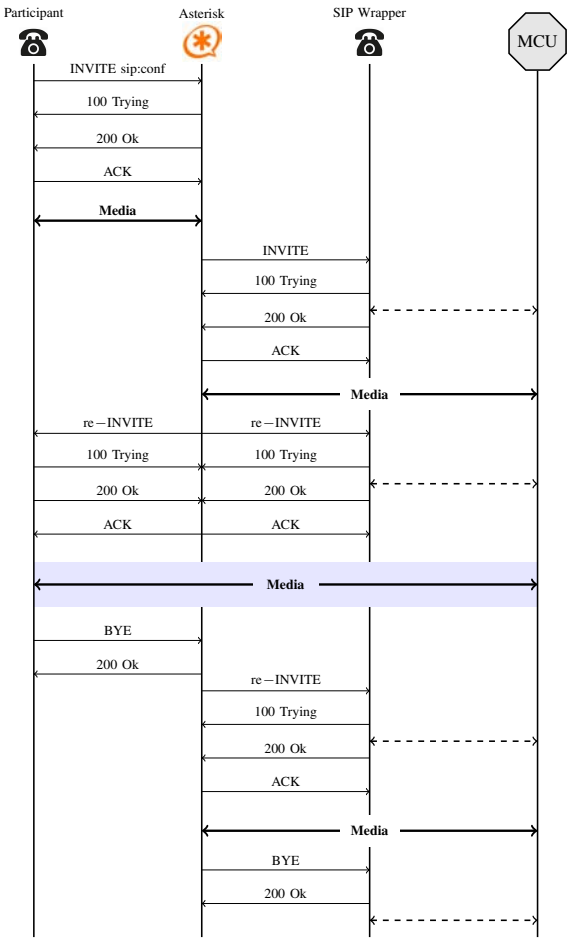

Fig. 2. Messages to join a conference.

wrapper uses the Kurento API to get the multimedia configuration (SDP description) to be used in the media negotiation during the call establishment. As a result, the SIP wrapper is responsible for keeping a call with each participant, and the Kurento server is responsible for multimedia data exchange.

When Asterisk has established the two calls between the participant and the SIP wrapper, it reconfigures both calls to make data flow directly between the participant and the SIP wrapper. Thus, the data plane follows a star topology as shown in Fig. 1.

This initial design for the platform has several shortcomings that may reduce scalability:

1) Asterisk is the entry point to the conference and all the calls must go through it, so it becomes a single point of failure. Besides, although configured in *pass-through* mode, the workload that Asterisk must deal with during the joining and leaving of participants in the conference is significant. During these periods, Asterisk sends and receives multimedia data, so this peak workload might limit quality of service.
2) The Kurento server, playing the role of an MCU, receives and merges multimedia data streams from all participants, so its workload could be high. In fact, with a star topology, and depending on the strategy for the data distribution followed, the workload of the MCU could increase exponentially [20].
3) The SIP wrapper is also a single point of failure that could compromise the signaling plane in the conference.

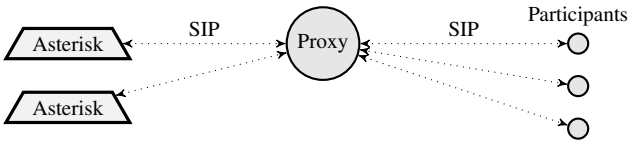

Fig. 3. SIP Proxy.

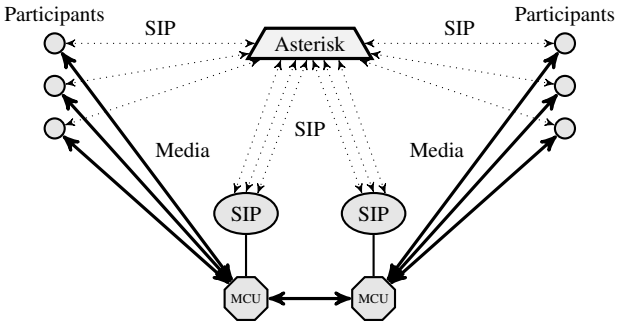

Fig. 4. Elastic architecture.

B. Elastic Design

A new elastic design provides solutions to the aforementioned shortcomings of the initial design:

1) A SIP Proxy distributes incoming calls among a set of Asterisk PBXs using a *round-robin* policy (see figure 3).
2) Several MCUs, and their corresponding SIP wrappers, act as one virtual MCU, as shown in Fig. 4, providing an scalable approach to an increasing number of participants.
3) A second SIP proxy acts as a backup proxy.

Using several MCUs poses a new scenario where some issues related to load balace must be solved. These issues are adressed in the following sections.

1) MCU Instantiation: New MCU instances should be provisoned in a cloud provider to support the increasing workload associated with the number of participants in the conference. This can be done through the Asterisk FastAGI API, triggering a control application (controller) when specific call events arise, and using the corresponding scripting API of the cloud provider. Figure 5 shows the elements involved in MCU provisioning. When serveral Asterisk PBXs are used all of them must invoque the same controller.

The steps to follow when a new participant joins the conference are: 1) The Asterisk dialplan is responsible for the participant authorization and triggering the controller; 2) The controller decides if the new participant is assigned to an active MCU or a new MCU must be provisioned; 3) If the participant

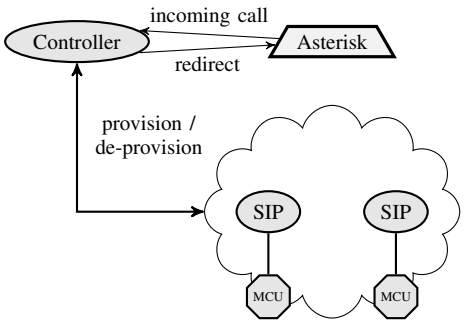

Fig. 5. MCU provisioning.

is assigned to an active MCU, the controller redirects the call to the corresponding SIP wrapper (using the extension defined in the dialplan); and 4) If a new MCU is necessary a script is executed to provision it. As a consequence, a new SIP wrapper is registered in Asterisk (through the proxy if using several Asterisk PBXs) and finally the controller redirects the call to the new SIP wrapper. Policies for provisioning MCUs in advance are mandatory to reduce the initial latency perceived by participants asigned to new MCUs.

2) Load Balance: The conference controller is also responsible for load balance between MCUs within the virtual MCU. The complexity of load balance algorithm depends on the flexibility of the multimedia configuration of participants. If all the participants have been asigned the same multimedia configuration (media, codecs and codecs configuration), then a simple round-robin algorithm could be used in the controller. In any case, including MCU monitoring channels could improve decision making for load balance in the controller.

3) Data plane reconfigurable topology: An MCU can carry out several actions on the streams coming from participants: simply forward them to the other participants, filter streams (i.e. using voice detection or floor-control) or mix and send them to all the participants. The data plane topology is a star when a single MCU is used in virtual MCU. In this case, all participants are connected to the MCU and the streams are homogeneusly processed. With more than one MCU in the virtual MCU, stream forwarding, filtering and mixing must be coordinated among MCUs. This is carried out using communication channels among MCUs (*trunks*) in a hybrid data plane topology as shown in Fig. 6. A custom communication protocol between MCUs and the controller is also necessary to reconfigure the topology after an MCU becomes active or inactive in the virtual MCU.

IV. Scalability Evaluation

Audio and video streams demand a lot of resources, so those elements of the platform receiving and sending the highest number of media streams are more prone to become a bottleneck. Since, each participant sends and receives one video stream and one audio stream, the resources required by participants are negligible compared to other entities.

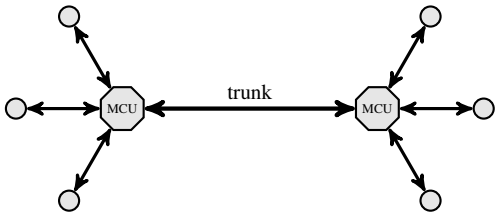

Fig. 6. Hybrid topology.

The MCUs are the entities that forward media streams between participants and between other MCUs, so they are supposed to require a high amount of resources. However, conferences usually include some floor-control mechanism to limit the number of streams flowing through the platform (for instance, to avoid media overlapping). As a result, only a few participants are allowed to send media streams at any time, so the load supported by MCUs is limited. They must process a few data streams and forward them to participants.

Media streams flow usually from participants to MCUs and vice versa directly, so the Asterisk PBXs and the proxy do not receive or send media streams, but only SIP messages. However, there are occassions when the load supported by these entities can not be ignored. For instance, conferences are usually scheduled, so many participants join in a short period of time. This may cause load peaks, specially in the Asterisk PBXs, since they must send audio streams to the participants with the dial tone until the SIP call is actually established.

In order to test the scalability of the platform the following testbed is used. A SIPp traffic generator emulates participants joining a conference. One or more Asterisk PBXs authenticate participants and provide access to the conference. A SIP proxy balances incoming SIP calls from participants to the Asterisk PBXs using round-robin load balancing. All the entities of the platform run on Ubuntu Server machines, with an Intel Core i7 CPU running at 3.6 GHz, with 4 GiB and with SSD storage.

Figure 7 shows the resource consumption in the Asterisk PBX and the percentage of calls that can be successfully established in a conference with only one Asterisk as the number of participants grows. All the participants join the conference at the same time, so Asterisk sends as many GSM audio streams conveying the dial tone as participants. Eventually, Asterisk stops sending the audio stream to the participant when the call is establish between the latter and the Kurento SIP wrapper. As can be seen in Fig. 7, the resource consumption grows with the number of participants almost linearly until saturation. At this point, the percentage of successful calls decreases preventing some participants to join the conference.

Figure 8 shows the same information as Fig.7 but when using a deployment with two Asterisk PBXs and a proxy balancing incoming calls between the two. The resource consumption are depicted for only one of the two Asterisk PBXs. As expected, the scalability of the platform is higher

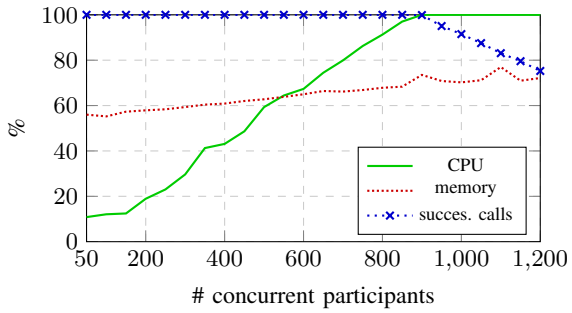

Fig. 7. Conference with one Asterisk PBX.

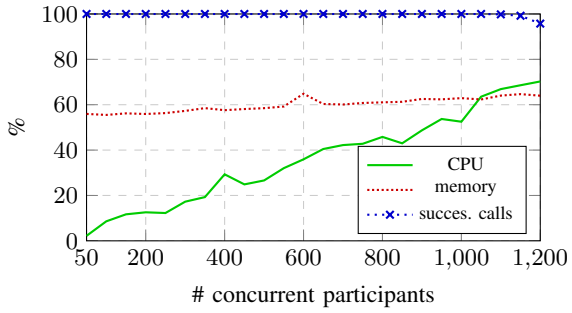

Fig. 8. Conference with two Asterisk PBXs in a load-balancing configuration.

than with a single PBX. The resource consumption is shared between the two PBXs. The final decrease on the percentage of successfull calls is caused by the SIPp reaching saturation.

As results in Fig. 7 and Fig. 8 prove, the platform may scale to hundreds of participants easily and even thousands when using load balancings between various Asterisk PBXs. This relies on the fact that the number of streams managed by the virtual MCU is limited or floor-controlled. However, it is unlikely that hundreds of participants join to a conference in a short period of time in order to make Asterisk send so many dial tone audio streams. The time while Asterisk sends the dial tone to a participant depends on the number of participants concurrently joining the conference and the time needed by the SIP wrapper to accept or reject an incoming call.

V. Conclusions

The architecture of an scalable platform for interactive multimedia communication based on WebRTC and supported by open technologies has been presented. The ubiquity provided by WebRTC and the interoperability with existing applications due to the fact of using standard protocols are other remarkable features of the platform.

Preliminary results prove that the platform may scale to hundreds or even thousands of users provided that the conference is floor-controlled.

Future work will be focused on assessing the performance of the virtual MCU in the platform when the number of media streams are not limited and the quality of experience perceived by users.

References

[1] WebRTC. (2015) Web real-time communications.

[2] A. Abell Lozano, V. Singh, and J. Ott, "Performance analysis of topologies for web-based real-time communication (webrtc)," Master's thesis, Aalto University, Aug 2013.

[3] J. Jang-Jaccard, S. Nepal, B. Celler, and B. Yan, "WebRTC-based video conferencing service for telehealth," *Computing*, vol. In press, 2014.

[4] A. Puel, A. von Wangenheim, M. I. Meurer, and D. D. de Macedo, "BUCOMAX: Collaborative multimedia platform for real time manipulation and visualization of bucomaxillofacial diagnostic images," in *Proc. IEEE International Symposium on Computer-Based Medical Systems (CBMS'14)*, New York, NY, May 2014, pp. 392–395.

[5] N. Hongo, H. Yamamoto, and K. Yamazaki, "Web shopping support system for elderly people using WebRTC," in *Proc. International Conference on Advanced Communication Technology (ICACT'14)*, Pyeongchang, South Korea, Feb. 2014, pp. 934–940.

[6] A. Kokane, H. Singhal, S. Mukherjee, and G. R. M. Reddy, "Effective e-learning using 3D virtual tutors and WebRTC based multimedia chat," in *Proc. International Conference on Recent Trends in Information Technology (ICRTIT'14)*, Chennai, India, Apr. 2014.

[7] G. Eriksson and S. Hkansson, "WebRTC: Enhancing the web with real-time communication capabilities," *Ericsson Review (English Edition)*, vol. 89, no. 1, pp. 4–9, 2012.

[8] S. Loreto and S. Romano, "Real-time communications in the web: Issues, achievements, and ongoing standardization efforts," *IEEE Internet Computing*, vol. 16, no. 5, pp. 68–73, 2012.

[9] W. Elleuch, "Models for multimedia conference between browsers based on WebRTC," in *Proc. International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'13)*, Lyon, France, Oct. 2013, pp. 279–284.

[10] P. Rodríguez, J. Cerviño, I. Trajkovska, and J. Salvachúa, "Advanced videoconferencing services based on webrtc," in *Proceedings of the IADIS International Conference Web Based Communities and Social Media*, 2012, pp. 180–184.

[11] Licode. (2015) Open source WebRTC communications platform.

[12] L. L. Fernández, M. P. Díaz, R. B. Mejías, F. J. López, and J. A. Santos, "Kurento: a media server technology for convergent WWW/mobile real-time multimedia communications supporting WebRTC." in *Proc. IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'13)*, Madrid, Spain, Jun. 2013.

[13] V. Xhagjika, O. Escoda, L. Navarro, and V. Vlassov, "Load and video performance patterns of a cloud based webrtc architecture," in *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID*, 2017, pp. 739–744, cited By 0.

[14] D. Nguyen, K. Nguyen, S. Khazri, and M. Cheriet, "Real-time optimized nfv architecture for internetworking webrtc and ims," in *Proceedings of the 17th International Telecommunications Network Strategy and Planning Symposium*, 2016, pp. 81–88, cited By 1.

[15] Y. Wu, C. Wu, B. Li, and F. Lau, "vskyconf: Cloud-assisted multi-party mobile video conferencing," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Mobile Cloud Computing*, 2013, pp. 33–38.

[16] A. Alonso, P. Rodríguez, J. Salvachúa, and J. Cerviño, "Deploying a multipoint control unit in the cloud: Opportunities and challenges," in *Proceedings of the 4th International Conference on Cloud Computing, GRIDs, and Virtualization*, 2013, pp. 173–1788.

[17] P. Rodríguez, A. Alonso, J. Salvachúa, and J. Cerviño, "dOTM: A mechanism for distributing centralized multi-party video conferencing in the cloud," in *Proc. International Conference on Future Internet of Things and Cloud (FiCloud'14)*, Barcelona, Spain, Aug. 2014, pp. 61–67.

[18] P. Rodríguez, A. Alonso, J. Salvachúa, and J. Cerviño, "Materialising a new architecture for a distributed mcu in the cloud," *Computer Standards and Interfaces*, vol. 44, pp. 234–242, 2016.

[19] J. Cerviño, P. Rodríguez, I. Trajkovska, F. Escribano, and J. Salvachúa, "A cost-effective methodology applied to videoconference services over hybrid clouds," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 103–109, Feb 2013.

[20] J. C. Granda, D. F. García, P. Nuño, and F. J. Suárez, "An efficient networking technique for synchronous e-learning platforms in corporate environments," *Comput. Commun.*, vol. 33, no. 14, pp. 1752–1766, Sep. 2010.