Computer Standards & Interfaces 44 (2016) 234–242

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

Materialising a new architecture for a distributed MCU in the Cloud

Pedro Rodríguez *, Álvaro Alonso, Joaquín Salvachúa, Javier Cerviño

Departamento de Ingeniería de Sistemas Telemáticos, Avenida Complutense n 30, "Ciudad Universitaria", 28040 Madrid, Spain

ARTICLE INFO

Article history:
Received 15 December 2014
Received in revised form 7 September 2015
Accepted 8 September 2015
Available online 18 September 2015

Keywords:
Cloud Computing
Video conferencing
Scalability
WebRTC
MCU

ABSTRACT

New technologies are making videoconferencing more ubiquitous than ever. This imposes a big challenge for scaling software MCUs, the traditional videoconferencing servers. We propose, implement and test an architecture for a distributed MCU designed to be deployed in a Cloud Computing environment. The main design idea is to break monolithic MCUs into more simple parts: broadcasters. These broadcasters can be deployed independently on the fly. This achieves a higher deployment granularity and flexibility. We describe the control architecture that allows this distribution and prove the viability of the system with a fully developed implementation.

1. Introduction

Traditionally, video conferencing communications have been implemented in native applications or dedicated hardware devices. However, in the last few years and thanks to technologies such as Adobe Flash[1] and HTML5[2] with its real-time communications standard WebRTC [1], these applications are moving towards the Web. That implies that any user with a PC, a smart phone or a tablet, a compatible Web browser and an Internet connection can use a videoconferencing service in order to communicate with others in real-time.

This improved accessibility and ease of use encourages the presence of video conferencing systems in more Internet applications and services. Social networks, television applications and on-line video game platforms are rapidly adopting real-time communications. This trend also applies to business environments, where on-line meetings, long-distance interviews, call centres and conference streaming are increasingly adopted.

This evolution towards the Web imposes new flexibility and scalability requirements to video conferencing infrastructures. Dedicated servers with fixed connections limits are becoming obsolete in a fast-paced Web environment where new popular sites are experiencing exponential growth in the number of concurrent users. At the same time, the surge in Cloud Computing systems has allowed developers to improve scalability on traditional services. Having more computing power on demand on a pay-per-use basis has set in motion a new ecosystem of new Web applications that react to the increasing traffic.

The convergence of these developments has given rise to new types of systems that offer video conferencing as a Cloud service. By taking advantage of Cloud technologies, they can adapt to large variations in the number of users. This is particularly important in video conferencing between multiple participants. Here, communication usually occurs in virtual rooms, conceptual places on the Internet where users can communicate with others.

Software multipoint control units (MCUs) are at the core of the server side of scalable video conferencing systems. MCUs have been used for years in these systems to address the signalling and real-time transport of user video and audio content. However, they can also support advanced functionality such as recording, video and audio transcoding, video composition and audio mixing. Nowadays, MCUs commonly serve many virtual rooms with multiple users in each of them. Hence, Cloud video conferencing services focus on solving the problem of adapting the number of MCUs to a varying number of users.

In this paper we propose an architecture to achieve this scalability with very high flexibility and granularity. Scaling dynamically is a challenge in a traditional MCU. The machine in which the MCU is deployed imposes a limit that cannot be raised without interrupting the running sessions. Furthermore, the highest granularity that one can achieve is that of a video conferencing session: a session being the atomic unit of deployment in an MCU system.

In conclusion, a video conferencing room has to be managed by a single MCU and it can never be split into different MCUs. This involves a very significant limitation in the deployment and the scalability of this type of system, a factor that can be improved in terms of efficiency in the management of the resources and costs.

The solution we propose focuses on the most illustrative example, which is the scenario in which the MCU forwards each video and audio stream from every user to the rest of participants in the same

* Corresponding author at: Av Complutense 30, Escuela Ingenieros de Telecomunicación UPM Edificio B, Despacho B323 Madrid, Madrid 28040 Spain.
E-mail address: prodriguez@dit.upm.es (P. Rodríguez).

[1] http://get.adobe.com/es/flashplayer/.
[2] http://dev.w3.org/html5/spec/.

http://dx.doi.org/10.1016/j.csi.2015.09.004
0920-5489/

room. The main task in this case is to copy every packet and send it to all participants. We have also defined the component that is responsible for carrying out this task within the MCU: the *OneToMany* processor. Our solution divides an MCU into multiple *OneToMany* components, one per audio and video stream that needs to be sent to the users. Moreover, we define the details we need to take into account for different layers of communication as well as general considerations to manage all the components in this system.

This work is built on top of the ideas proposed in [2] the distributed OneToManys (dOTMs), an atomic software component that receives a media stream published by a user and multiplexes it to its subscribers. In a video conference room with multiple participants there will be a OneToMany (OTM) for each user publishing a stream. The OTMs are managed by a control layer and each one is totally independent of the others. Thus, OTMs that belong to users connected to the same virtual room can be distributed between different machines. An immediate advantage of this architecture is that when you need more resources to attend to the demand in a videoconferencing session, you can add a new machine on-the-fly.

In the next section we extend the motivations introduced here and we analyse the requirements for achieving the goals of this research. In Section 3 we summarise the current work related to the problem. Then in Section 4 we present our solution by describing the architecture and its execution flow in a videoconferencing scenario. In Section 5 we show the implications of this architecture and, using a real open-source implementation of dOTMs (Licode[3]), we demonstrate that the new model does not negatively affect the performance of the communications. Finally, in Section 6 we show the main conclusions of this work and we introduce some lines of research for future work related to it.

2. Scenario and problem description

In this section we enunciate the problem this work aims to solve. In order to do that, we describe the scenario where the problem takes place, define the terms we will use throughout the paper and elaborate the problem description by providing an example on a real software MCU.

2.1. Scenario

Video conferencing with multiple participants has traditionally been provided by MCUs, as opposed to peer-to-peer solutions. The lack of availability of IP Multicast [3] for end-users impairs the use of p2p for multi-party real time communications as the number of participants dictates the upload bandwidth needed. In such cases, an MCU-based application level multicast is used. The centralised architecture means that there is a bottleneck on the server side but, on the other hand, clients save uploaded bandwidth as they do not have to replicate the streams they are publishing for each participant.

A few years ago, videoconferencing systems were limited to native and desktop applications. But current new technologies also provide video conferencing on the Web browser. As Web applications are more easily accessible and impose fewer requirements in the devices, this enables a new and more dynamic way of communicating. However, it also imposes a great challenge for MCU deployments. The traditional dedicated and tightly scheduled infrastructures are not tailored to meet the fast changes in demand associated with Web applications. Cloud Computing, and its ability to provide resources on demand, has been widely used as a solution for scalability problems. Specifically when talking about video conferencing, the illusion of infinite computational power and bandwidth that the Cloud provides is very attractive for deploying resource demanding services. However, designing and deploying software in the Cloud is a challenge in itself.

Several works have tried to take advantage of the capabilities the Cloud provides applied to video conferencing. In [4] the authors provide a general overview on video conferencing services in the Cloud. By designing the conference as service oriented architecture, the final consumer application is separated from the video conference provider. Conferencing services are offered on request in a similar way that infrastructure is offered in the Cloud. The authors in [5] present an implementation that adheres to some of these principles. While the general idea on how to adapt video conferencing systems to the Cloud is clear in these works, it is not specified how to react to the variations on demand or how to provide the resources for the video conferencing capabilities dynamically.

On the other hand, since the term *Cloud Computing* was coined and its characteristics set out, there has been an abundance of studies on how to take different kinds of applications *to the Cloud*, taking advantage of its strengths and highlighting the risks. According to [6] the ability to scale quickly and efficiently is one of the top ten obstacles (and opportunities) for the growth of Cloud Computing and the applications that rely on it. Furthermore, the authors in [7] argue that cost and efficiency is one of the risks businesses have to take into account when transitioning to Cloud services. The authors in [8] emphasise the importance of taking into account the different requirements of the applications in terms of quality of experience (QoE) when adapting their Cloud deployment.

To sum up, Cloud Computing can be a very powerful tool for creating scalable video conferencing services. However, to maximise its value (economic and otherwise), the Cloud deployment has to be carefully designed, implemented and adapted to the service in order to provide a successful experience with minimal cost.

This paper proposes a way to improve the efficiency when it comes to deploying and providing resources for multiple participant video conferencing in Cloud systems.

To illustrate this, we will focus on the scenario of a multi-party video conference where an arbitrary number of users sends and receives real time audio and video streams. A use case of this scenario is, for instance, a business meeting in which all participants share their audio and video. These participants will send all the traffic through an MCU in order to save upload bandwidth as explained before. In the following subsections we will provide the definitions that will be used throughout the paper and will further explain the scenario of the problem.

2.2. Definitions

The following terms will be used throughout this paper. These are illustrated in Fig. 1:

- *Participant*: A device that takes part in a video conference. It can receive and send multimedia streams. Generally, it will represent the final users of the system.

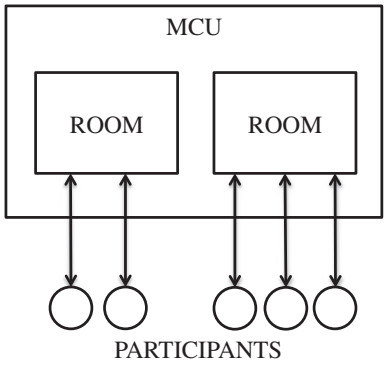

Fig. 1. Agents in an MCU-based video conference.

[3] https://github.com/ging/licode.