UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

OBERDAN COSTA DOS SANTOS

Gravação de vídeo em sistemas de videoconferência

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Valter Roesler

Porto Alegre
2019

AGRADECIMENTOS

Meus primeiros agradecimentos vão, indubitavelmente, para o Sr. Vilmar Pereira dos Santos e para a Sra. Margarida Dias da Costa dos Santos, meus pais. Eles foram, e são até hoje, os meus primeiros mestres e amigos. Agradeço a vocês dois por toda a luta, tempo e tudo mais investidos, única e exclusivamente, em mim. Sei que não foi fácil e embora, às vezes, eu não demonstre tanto quanto deveria, saibam que vocês são meu maior orgulho e tudo que eu já conquistei e vou conquistar devo a vocês.

Quero agradecer também a minha namorada e melhor amiga, Ana Rosa Nipper Ramos, que praticamente vivenciou todos os anos que passei na graduação, seja me dando conselhos, motivação ou até mesmo um abraço apertado seguido da frase "vai dar tudo certo", nos momentos mais difíceis que passei nesses nove anos. Posso garantir que sem você ao meu lado eu não teria chegado aqui.

Agradeço ao meu orientador e também chefe, Valter Roesler, pela oportunidade de bolsa no projeto PRAV e pelo total suporte e disponibilidade no desenvolvimento do trabalho dessa monografia. Não posso deixar de agradecer também a toda equipe do GT-MCU e do GT-Videocolaboração pela ótima experiência e convivência até o presente momento. Em especial, ao líder dos GT's citados, Mario Cesar Gasparoni Junior, que além de ser uma excelente pessoa possui um conhecimento absurdo e está sempre disposto a dividi-lo com todos. Agradeço por todo o suporte nesse pouco mais de um ano de convivência.

Agradeço também as professoras, Renata de Matos Galante e Viviane Pereira Moreira, responsáveis pela minha primeira atividade de monitoria da graduação. Sem dúvidas foi um período de grande aprendizado que eu valorizo muito e sou grato até hoje.

Agradeço também, ao pessoal da Escola de Direito, desde a direção na figura da Sra. Ana Luiza Vianna, passando pela secretaria nas figuras da Sra. Yara Garcia de Freitas e do Sr. Francisco Delmar Lopes Matheus, pela minha primeira oportunidade como bolsista. Faço uma menção honrosa à segurança do prédio, Ana, uma pessoa muito especial que, por muitas vezes, ouvia-me e dava conselhos certeiros. Foi um curto período, mas levo um carinho muito grande de todos vocês.

Agradeço a todos professores que tive até o presente momento, desde o colégio, passando pelos cursos e pré-vestibulares até chegar e concluir a graduação. Vocês foram e são parte importante da minha construção pessoal e acadêmica.

Por fim, agradeço aos meus familiares e amigos que fizeram parte desse período da minha vida. Se hoje eu venci é por que vocês em algum momento me deram o suporte

necessário. Destaco meu padrinho, Paulo Ricardo Padilha, e meu amigo e “segundo pai”, Sérgio Dornelles, que acompanhavam minha mãe nos dias que prestei vestibular lá nos longínquos anos de 2009 e 2010. Agradeço também aos colegas e amigos de graduação: Fábio Petkowicz, Ricardo Sabedra, Pablo Soares, Rafael Calçada, Vinicius Scheffel e Rodrigo Okido, pela parceria e amizade construída durante a graduação.

RESUMO

Sistemas de videoconferência representam economia de tempo e recursos. Pode-se afirmar isso já que esses sistemas evitam deslocamentos e gastos com viagens. A disponibilização da gravação dessas reuniões permite o acesso assíncrono a quem não pode participar ou quer rever o vídeo. A proposta deste trabalho foi analisar o funcionamento da gravação de vídeo em sistemas de videoconferência e, além disso, foi efetuada a implementação de um sistema de gravação. E, para melhor avaliar essa implementação o seu desempenho foi analisado em diferentes *hardwares*. O escopo deste estudo pode ser dividido em três partes: contextualização do assunto, desenvolvimento da ferramenta e avaliação através de testes comparativos. Na primeira parte são apresentados conceitos que aproximam o leitor do assunto e também uma breve noção do mercado sobre o tema tratado. No desenvolvimento é mostrado toda a construção da ferramenta desde a arquitetura desenhada até os casos de uso trabalhados. Na parte final, são trabalhados todos os testes propostos na validação e então os resultados obtidos são apresentados através de tabelas. Sobre os resultados será possível observar que, indiferente dos *hardwares* ou das ferramentas utilizadas, a presença de transcodificação de vídeo é um fator determinante no desempenho de maneira negativa já que gera alto consumo de CPU. Por outro lado, sua presença traz um resultado que pode ser considerado positivo no tamanho dos arquivos gerados. Ficando, dessa forma, a cargo do usuário do sistema a decisão de mensurar o melhor custo-benefício para sua aplicação.

Palavras-chave: Videoconferência. Gravação de vídeo. Transcodificação.

Video recording in videoconferencing systems

ABSTRACT

Video conferencing systems save time and resources. This can be said as these systems prevent travel and travel expenses. Making the recording of these meetings available allows asynchronous access to those who cannot attend or want to review the video. The purpose of this work was to analyze the operation of video recording in videoconferencing systems and, in addition, the implementation of a recording system was implemented. And to better evaluate this implementation its performance was analyzed on different hardware. The scope of this study can be divided into three parts: subject contextualization, tool development and evaluation through comparative tests. In the first part are presented concepts that bring the reader closer to the subject and also a brief notion of the market on the subject. In development is shown all the construction of the tool from the architecture designed to the use cases worked. In the final part, all the tests proposed in the validation are worked and then the obtained results are presented through tables. Regarding the results, it will be observed that, regardless of the hardware or tools used, the presence of video transcoding is a determining factor in the negative performance since it generates high CPU consumption. On the other hand, its presence brings a result that can be considered positive in the size of the generated files. Thus, it is up to the system user to decide to measure the most cost-effective for their application.

Keywords: Videoconference. Video recording. Transcoding.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

a.k.a	*as known as*
AC3	*Audio Coding 3*
ACK	*Acknowledgement*
API	*Application Programming Interface*
AVC	*Advanced Video Coding*
AVI	*Audio Video Interleave*
BD	Banco de Dados
BFCP	*Binary Floor Control Protocol*
BSD	*Berkeley Software Distribution*
CPU	*Central Processing Unit*
CSRC	*Contributing Source*
FFmpeg	*Fast Forward* Mpeg
FLV	*Flash Video*
GE	Gerenciador de Escalabilidade
GNU GPL	GNU *General Public License*
GT	Grupo de Trabalho
HD	*High Definition*
HTTP	*HyperText Transfer Protocol*
IDE	*Integrated Development Environment*
IEC	*International Electrotechnical Commission*
IETF	*Internet Engineering Task Force*
Inc	*Incorporation/ Incorporated*
IP	*Internet Protocol*
IPv4	*Internet Protocol version 4*
IPv6	*Internet Protocol version 6*
ISO	*International Organization for Standardization*
ITU-T	*International Telecommunication Union - Telecommunication Standardization Sector*
kbps	*kilobits per second*
kHz	*kilohertz*
KMS	*Kurento Media Server*
Ltda	Limitada

MCS	*Media Control Server*
MCU	*Multipoint Control Unit*
MKV	*Matroska Video*
MP4	MPEG-4 *Part* 14
MPEG	*Moving Picture Experts Group*
MVC	*Model-View-Control*
NUC	*Next Computing Unit*
P2P	*Peer-to-Peer*
PRAV	Projetos em Áudio e Vídeo
RFC	*Request For Comments*
RNP	Rede Nacional de Ensino e Pesquisa
RTC	*Real-Time Communication*
RTCP	*Real-time Transport Control Protocol*
RTP	*Real-time Transport Protocol*
SB-ADPCM	*Sub-Band Adaptive Differential Pulse Code Modulation*
SDP	*Session Description Protocol*
SFU	*Selective Forwarding Unit*
SG	Servidor de Gravação
SIP	*Session Initiation Protocol*
SM	Servidor de Mídia
SMPTE	*Society of Motion Picture and Television Engineers*
SMTP	*Simple Mail Transfer Protocol*
SSRC	*Synchronization Source*
UDP	*User Datagram Protocol*
UFRGS	Universidade Federal do Rio Grande do Sul
VCEG	*Video Coding Experts Group*
VM	*Virtual Machine*
VoIP	*Voice over Internet Protocol*
W3C	*World Wide Web Consortium*
WAV	*WAVEform Audio Format*
WebRTC	*Web Real-Time Communication*
WMV	*Windows Media Video*

SUMÁRIO

1 INTRODUÇÃO

Videoconferências, sejam elas por *hardware*, ou via *web*, são cada mais frequentes no contexto atual. Seja pela sua praticidade e comodidade ou pelo simples fato de que as pessoas hoje não querem perder tempo e dinheiro se deslocando de um lugar a outro apenas para uma conferência de algumas horas. Esse deslocamento deixa de ser vantajoso, se é que um dia foi, no momento em que se perde mais tempo e disposição no percurso para a reunião do que nela propriamente.

Uma das soluções encontradas para o problema da falta de disponibilidade para reuniões foi a gravação das videoconferências e a disponibilização do seu conteúdo. Entretanto, essa nova função tem seus prós e contras. Torna-se muito útil ter acesso a gravação quando quiser, por outro lado, existe um preço a ser pago por isso. Em alguns casos se faz necessário um tempo de pós-processamento do arquivo ao final, atrasando sua disponibilização e podendo gerar transtornos. Esse tempo, às vezes, pode ser de horas, inclusive. E, não menos importante, problemas de falhas ou perdas na qualidade durante a gravação. A soma desses transtornos acaba gerando mal-estar porque, dependendo da situação, há a necessidade de assistir ao arquivo de gravação imediatamente.

Pensando nisso, em parceria com o Grupo de Trabalho de Multipoint Control Unit (GT-MCU), que atualmente se chama GT-Videocolaboração, financiado pela Mconf Tecnologia Ltda e RNP (Rede Nacional de Ensino e Pesquisa), e executado na UFRGS, sob o comando do professor Valter Roesler, foi desenvolvido, pelo autor desta monografia, um sistema de gravação em tempo real de videoconferências em alta resolução sem a presença do tal tempo de pós-processamento. Esse sistema, também chamado de Gerenciador de Gravação, é o foco desta monografia e, como forma de avaliação, foi submetido a testes, comparando-o a *hardwares* de alto desempenho que possuem características de gravação.

1.1 Objetivos do trabalho

Este estudo tem por objetivos:

1. Implementar um sistema de gravação sem tempo de pós-processamento;
2. Testar e comparar o desempenho do sistema desenvolvido em diferentes *hardwares.*

Dessa forma será possível assimilar os pontos positivos existentes e ainda sugerir possíveis melhorias, para trabalhos futuros, dependendo da necessidade do usuário do sistema.

1.2 Organização do texto

Os primeiros capítulos apresentam fundamentos e justificativas para a realização deste trabalho. O capítulo 2 introduz todos os conceitos importantes envolvendo videoconferência e gravação fundamentais para o entendimento do trabalho. Além de uma análise de realidade sobre o tema, apresentando um contexto envolvendo sistemas (e produtos) responsáveis pela gravação de uma videoconferência.

A partir do capítulo 3, todo o processo de desenvolvimento é documentado. O capítulo 4 abrange toda a parte de avaliação e testes do projeto. Apresentando desde metodologia até a discussão dos resultados obtidos. Por fim, no capítulo 5, são ressaltados os principais desafios e contribuições deste trabalho, além de melhorias que podem ser feitas.

2 CONCEITOS ENVOLVENDO VIDEOCONFERÊNCIA E GRAVAÇÃO

Alguns termos vão ser recorrentes neste texto, por conta disso, neste capítulo, os mesmos serão mais detalhados. Tratam-se dos modelos de distribuição de mídia descritos pela literatura, dos protocolos de transporte, sinalização, negociação e compartilhamento de mídia, dos *codecs* (acrônimo para codificadores/decodificadores) e contêineres multimídia utilizados neste trabalho, além dos *frameworks* de mídia e tecnologias de código aberto que auxiliaram na realização deste projeto.

2.1 Modelos de distribuição de mídia

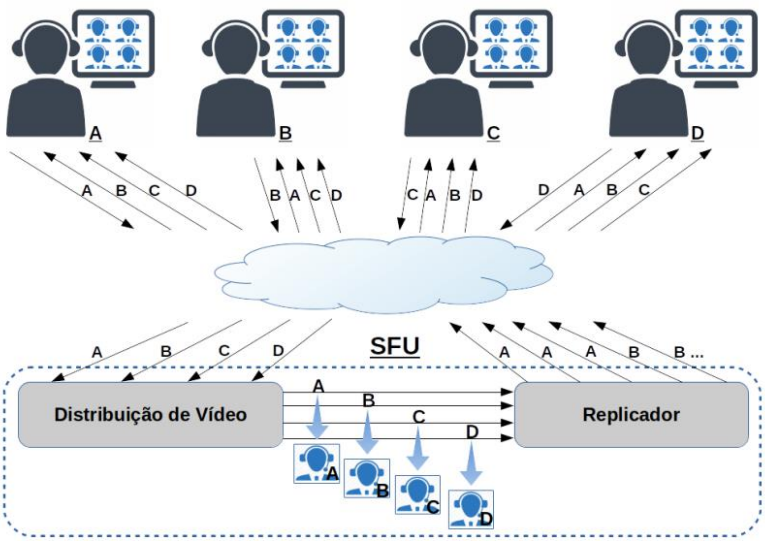
O conceito principal dos sistemas de videoconferência é permitir a troca de fluxo de mídia entre os pares conectados. Segundo GROZEV et al. (2015), topologias comuns para a distribuição de mídia em ambientes de videoconferência são a SFU (*Selective Forwarding Unit*), MCU (*Multipoint Control Unit*) e *MESH* (ou *Peer-to-Peer*). Por este trabalho ter sido feito baseado apenas nos modelos SFU e MCU, o modelo *MESH* não será abordado.

2.1.1 *Selective Forwarding Unit* (SFU)

A Figura 2.1 apresenta a dinâmica do SFU que por característica não possui nenhum elemento central. Ela apenas faz a distribuição (ou encaminhamento) dos fluxos de mídia até os equipamentos conectados (*endpoints)* sem nenhum tipo de processamento adicional (codificação ou decodificação) o que, geralmente, gera mais eficiência de CPU.

Considerando a eficiência de largura de banda, uma SFU não encaminha todos os pacotes, na verdade, ela usa um algoritmo de seleção que decide qual pacote encaminhar e para qual *endpoint* (GROZEV et al., 2015). Resumidamente, o cenário padrão é: usuários enviam seus fluxos de mídia para o SFU que, internamente, vai distribui-los separadamente, replica-los (usando o módulo replicador visto na Figura 2.1) e enviar para cada *endpoint* uma cópia de cada fluxo de cada um dos outros usuários.

Figura 2.1 - Dinâmica de funcionamento SFU

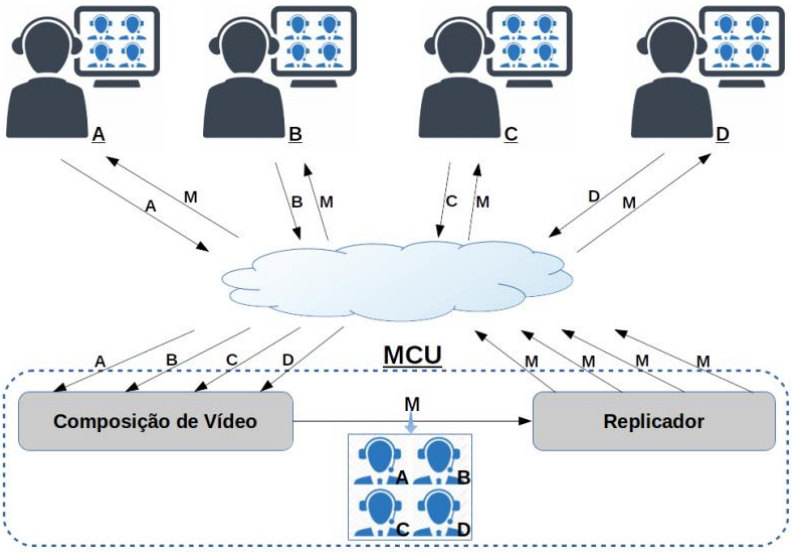


Fonte: (ROESLER et al., 2018)

2.1.2 *Multipoint Control Unit* (MCU)

A Figura 2.2 detalha a ação de um MCU que é um equipamento ou serviço usado em videoconferências que permite comunicação multiponto. Ele funciona como um elemento central que recebe os fluxos de mídia de cada um dos usuários conectados, passa por um processo de composição de vídeo, gerando um único fluxo (ilustrado na Figura 2.2 como fluxo M). Esse novo fluxo é replicado e enviado para todos os *endpoints*, possibilitando aos usuários visualizarem tal composição.

Figura 2.2 - Dinâmica de funcionamento MCU



Fonte: (ROESLER et al., 2018)

2.2 Protocolos de rede

Todas as atividades na Internet que envolvem duas ou mais entidades remotas comunicantes são governadas por um protocolo. Nas próximas subseções são apresentados os protocolos importantes para o contexto de videoconferência e gravação de vídeo (KUROSE; ROSS, 2013).

2.2.1 *Real-time Transport Protocol* (RTP)

O protocolo de transporte em tempo real determina um formato de pacote padrão para o envio de áudio e vídeo pela Internet. Foi definido inicialmente pela RFC 1889 e depois pela RFC 3550. O protocolo fornece serviços de entrega fim-a-fim como identificação do tipo de carga, numeração de sequência, registro de data e hora, e monitoramento de entrega para dados em tempo real como, por exemplo, áudio e vídeo em videoconferências. Normalmente, as aplicações executam o RTP sobre um protocolo de transporte, no caso, o UDP. Além disso, o RTP permite a transferência de dados usando distribuição *multicast* caso seja necessário.

Tratando-se de um cenário de videoconferências, as mídias de áudio e vídeo usadas serão transmitidas em sessões RTP separadas, ou seja, usando dois pares de portas UDP diferentes e/ou endereços *multicast*. Essa opção existe para ceder ao participante a capacidade de receber, ou compartilhar, apenas um meio conforme preferir (SCHULZRINNE, 2003).

2.2.2 *Real-time Transport Control Protocol* (RTCP)

O RTCP, assim como o próprio RTP, foi definido inicialmente pela RFC 1889 e depois pela RFC 3550. O protocolo atua juntamente com o RTP e tem como funções: (1) permitir o monitoramento da entrega de dados de maneira escalável para grandes redes *multicast,* e; (2) fornecer funcionalidade mínima de controle e identificação, ou seja, monitorando a qualidade do serviço e transmitindo informações sobre os participantes de uma sessão. Além de suportar o uso de *mixadores* e tradutores RTP.

Voltando ao cenário de videoconferências, como não existe acoplamento direto (em nível RTP) entre sessões de áudio e vídeo o usuário deve possuir o mesmo nome canônico (CNAME – *canonical name*) nos pacotes RTCP (áudio e vídeo) para que as sessões possam ser associadas (SCHULZRINNE, 2003). O nome canônico funciona como um identificador persistente no nível de transporte para uma origem RTP.

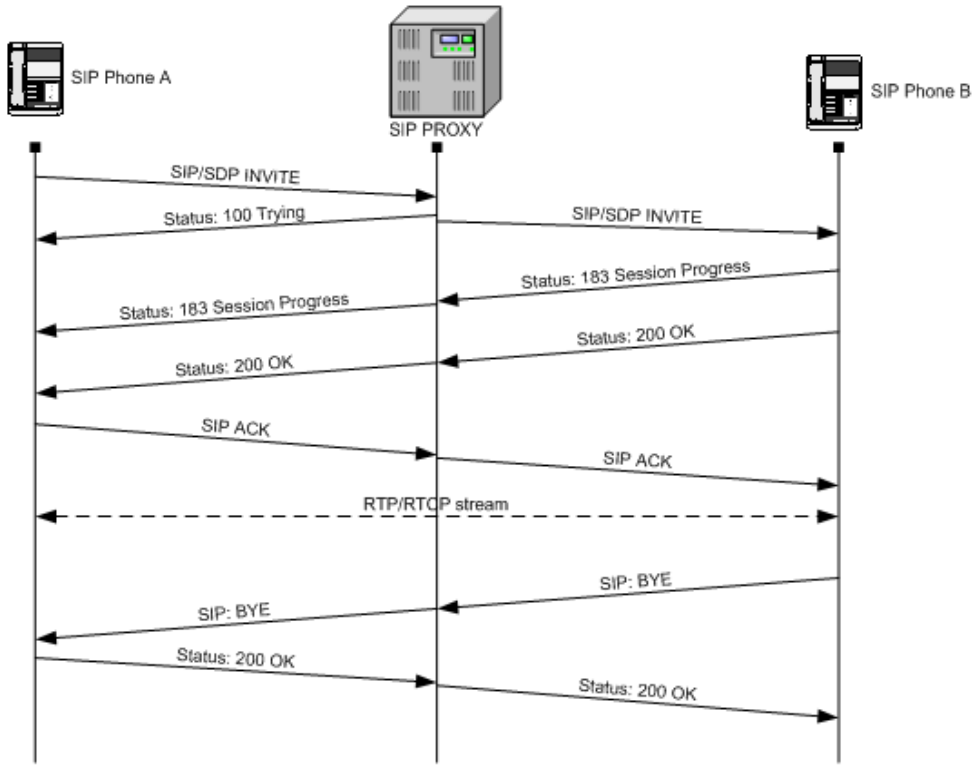
2.2.3 *Session Initiation Protocol* (SIP)

O SIP é um protocolo de sinalização da camada de aplicação que pode estabelecer, modificar e encerrar sessões multimídias (videoconferências, ensino a distância, etc.) ou chamadas (ROSENBERG et al., 2002). O SIP é o padrão para comunicação de voz e vídeo em tempo real. Ele foi criado pela IETF e publicado como RFC 3261.

Além disso, o SIP funciona tanto em sessões *unicast* como nas *multicast* podendo ser usado tanto com IPv4 como IPv6. O desenvolvimento do SIP se deu visando o suporte a pontos importantes do estabelecimento de sessões multimídia na rede, como, localização, disponibilidade e capacidades do usuário, estabelecimento, configuração e gerenciamento de sessões (ROSENBERG et al., 2002).

Na Figura 2.3 pode-se observar uma troca de mensagens SIP desde o seu início até a sua conclusão. Como pode ser visto, uma chamada de um usuário SIP A para um usuário SIP B passa pelas seguintes etapas: estabelecimento do contato com a troca dos pacotes SIP/SDP *INVITE*, OK e SIP ACK. Com um canal de conexão estabelecido a troca de dados pelos pacotes RTP/RTCP *stream* em via dupla estão liberados. Ao final da comunicação é feita a finalização da chamada por meio dos pacotes SIP *BYE* e OK.

Figura 2.3 - Exemplo de comunicação SIP



Fonte: (MASTALIR, 2005)

2.2.4 *Session Description Protocol* (SDP)

O SDP é um conjunto de regras que define como sessões multimídia serão configuradas para permitir aos *endpoints* participar da sessão[1]. Foi definido pela RFC 2327, em 1998, e mais tarde atualizado pelas RFC 3266 (2002) e RFC 4566, em 2006. Em geral, o SDP deve cumprir dois objetivos principais: transmitir informações suficientes para permitir que aplicações ingressem em uma sessão e anunciar os recursos que um provável futuro participante possa precisar saber (HANDLEY; JACOBSON; PERKINS, 2006).

As próximas figuras apresentam um exemplo de arquivo SDP fragmentado pela descrição de sessão e de tipos de mídia. Na Figura 2.4, constam descritas informações referentes a versão do protocolo (v), criador da sessão (o), nome da sessão (s), tempo de atividade da sessão (t), informações de conexão (c) e atributos de sessão (a). Esses dois últimos sendo opcionais.

Figura 2.4 - Trecho de arquivo SDP com descrição da sessão

```
v=0
o=- 3778853736 3778853736 IN IP4 143.54.10.106
s=Kurento Media Server
c=IN IP4 143.54.10.106
t=0 0
a=sendrecv
a=vnd.polycom.MBA.p2p:v=1.0.1
```

Fonte: O Autor

Nas Figura 2.5, Figura 2.6 e Figura 2.7, respectivamente, estão representadas as mídias de áudio, vídeo e conteúdo. Nelas constam o nome da mídia e endereço de transporte (m), além das informações opcionais: atributos da mídia (a), largura de banda (b) e conexão (c). Essa última permite que uma mídia se conecte a um endereço IP diferente da sessão se for necessário.

Figura 2.5 - Trecho de arquivo SDP com descrição da mídia de áudio

```
m=audio 16426 RTP/AVP 9 119
a=rtpmap:9 G722/8000
a=rtpmap:119 telephone-event/8000
a=fmtp:119 0-16
a=rtcp:16427 IN IP4 143.54.10.32
a=ptime:20
```

Fonte: O Autor

[1] https://searchunifiedcommunications.techtarget.com/definition/SDP - acesso em setembro 2019

Figura 2.6 - Trecho de arquivo SDP com descrição da mídia de vídeo

```
m=video 25714 RTP/AVP 111 109 110
b=TIAS:300000
b=AS:300
a=rtpmap:111 H264/90000
a=rtpmap:109 H264/90000
a=rtpmap:110 H264/90000
a=fmtp:111 profile-level-id=42e01f; packetization-mode=0; level-asymmetry-allowed=0
a=fmtp:109 profile-level-id=42e01f; packetization-mode=0; level-asymmetry-allowed=0
a=fmtp:110 profile-level-id=42e01f; packetization-mode=0; level-asymmetry-allowed=0
a=rtcp-fb:* nack pli
a=rtcp-fb:* ccm fir
a=rtcp-fb:* ccm tmmbr
a=sendrecv
a=ssrc:2483496669 cname:user3754850593@host-56e4fd62
```

Fonte: O Autor

Figura 2.7 - Trecho de arquivo SDP com descrição da mídia de conteúdo

```
m=application 50000 UDP/BFCP *
c=IN IP4 143.54.10.32
a=sendrecv
a=floorctrl:s-only
a=setup:passive
a=confid:1
a=userid:100
a=floorid:1 m-stream:3
a=connection:new
```

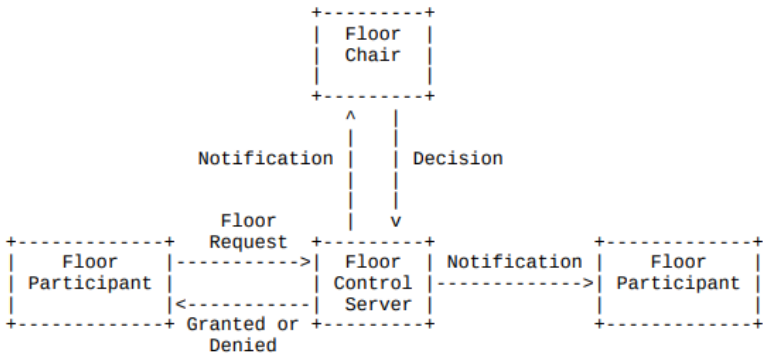
Fonte: O Autor

2.2.5 *Binary Floor Control Protocol* (BFCP)

O BFCP foi criado pela IETF e definido na RFC 4582 em 2006. O *floor* é definido como a permissão temporária para gerenciar o acesso ou manipular um determinado recurso compartilhado ou conjunto de recursos (CAMARILLO; OTT; DRAGE, 2006). Ele complementa funções como configurar conferência e sessão de mídia, manipular política de conferência e controle de mídia (realizados por outros protocolos).

No contexto de videoconferências, o BFCP é usado entre participantes, moderadores e servidores de controle. Na Figura 2.8, retirada da RFC 4582, é exibido o fluxo de quando um participante tenta tomar o poder do *floor.* Na Figura 2.8, o participante requisita o *floor* através da mensagem *Floor Request*, o servidor então recebe a mensagem e notifica o moderador, que irá decidir se o participante pode ou não ser o *floor* naquele momento. Sabendo da decisão do moderador (*floor chair*), o servidor informa ao participante que requisitou o pedido se a resposta foi positiva ou não e, em caso afirmativos, notifica os demais participantes da mudança.

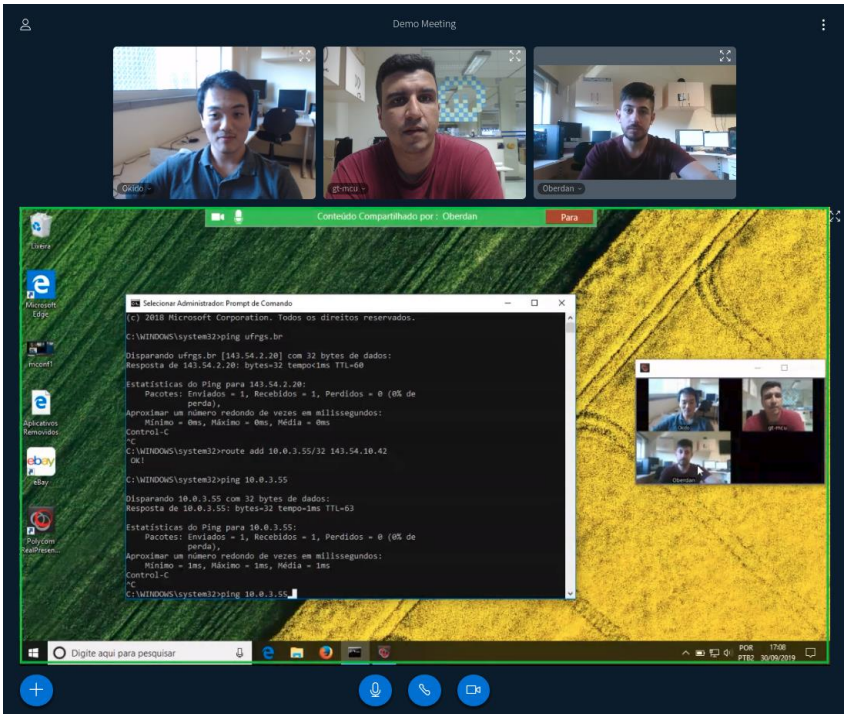
Figura 2.8 - Fluxo de um *Floor Request* no BFCP



Fonte: (CAMARILLO; OTT; DRAGE, 2006)

Na Figura 2.9 exemplifica-se o uso do protocolo que permite enviar e receber conteúdo em uma tela adicional da videoconferência. Dessa maneira, pode ser visto em uma tela os participantes da sessão e em outra o compartilhamento de tela do *floor*. O contexto da Figura 2.9 mostra um ambiente *web* com usuários SIP através do software *RealPresence Desktop*[1] sendo um deles quem está compartilhando o conteúdo.

Figura 2.9 - Chamada com compartilhamento de conteúdo



Fonte: O Autor

[1] https://support.polycom.com/content/support/north-america/usa/en/support/video/realpresence-desktop/realpresence-desktop.html - acesso em setembro 2019

2.3 Transcodificação e *codecs*

Em videoconferência a transcodificação é um processo que envolve essencialmente duas etapas: decodificação e codificação de mídia (áudio ou vídeo). Geralmente, é utilizada quando se quer alterar características da mídia em questão, como por exemplo, taxa compressão do áudio ou tamanho do vídeo.

O termo *codec* vem de codificação e decodificação. Dessa forma, pode-se dizer que um *codec* comprime um arquivo originalmente muito grande em um conteúdo reduzido para que possa ser compartilhado na rede. Após isso ele é "reconstruído" para ser executado.[1]

O que diferencia os *codecs* é a forma de codificação ou decodificação. Por exemplo, existem aqueles que comprimem ignorando totalmente possíveis perdas de qualidade, aqueles que comprimem primando evitar baixas na qualidade da transmissão, e também outros que comprimem ao máximo em momentos que a perda da qualidade é aceitável.

No contexto de videoconferências, tanto para áudio quanto vídeo, existe a necessidade de que o par que esteja se comunicando possua ao menos um *codec* em comum, não importando o modelo de distribuição usado (SFU, MCU, etc.). Existem muitos *codecs* conhecidos, tais como, H.264, VP8, MPEG-2, H.263, WMV, WAV, AC3, G.722, OPUS, etc.

2.3.1 *Codec* de vídeo – Padrão H.264

O padrão H.264, ou MPEG-4 *Part* 10, *Advanced Video Coding* (MPEG-4 AVC), é um padrão de compactação de vídeo baseado em compensação de movimento orientado a blocos. Desde 2014, é um dos formatos mais usados para a gravação, compactação e distribuição de conteúdo de vídeo (OZER, 2016). O nome H.264 segue a convenção de nomenclatura ITU-T, onde o padrão é um membro da linha H.26x dos padrões de codificação de vídeo VCEG; o nome MPEG-4 AVC refere-se à convenção de nomenclatura na ISO / IEC MPEG, onde o padrão faz parte da ISO / IEC 14496, que é o conjunto de padrões conhecido como MPEG-4.

A motivação foi criar um padrão capaz de fornecer boa qualidade de vídeo a taxas de bits mais baixas do que os padrões anteriores (MPEG-2, H.263, etc.) sem aumentar a complexidade do *design* a ponto de torná-lo impraticável ou muito custoso de implementar. O

[1] https://canaltech.com.br/software/o-que-sao-codecs-quais-sao-os-tipos-pra-que-servem-saiba-mais-sobre-esse-tema/ - acesso em setembro 2019

H.264 é normalmente usado para compactação com perdas embora suporte casos de uso raros nos quais toda a codificação é sem perdas.

O H.264 foi padronizado pelo ITCE-T *Video Coding Experts Group* (VCEG) em conjunto com o ISO / IEC JTC1 *Moving Picture Experts Group* (MPEG). A primeira versão do padrão foi redigida em maio de 2003 e várias extensões foram adicionadas nas edições subsequentes. A MPEG LA, e outros proprietários de padrões, têm direito a *royalties* pelo uso comercial das tecnologias H.264.[1]

2.3.2 *Codec* de áudio – G.722

O G.722 é um *codec* de áudio que segue o padrão da ITU-T aprovado em 1988 e que pode operar em 48, 56 ou 64 Kbps. Ele coleta dados de áudio a uma taxa de 16 KHz; esse valor equivale a duas vezes a velocidade das interfaces de telefonia tradicionais.[2] Dessa maneira, o *codec* fornece maior clareza e qualidade de áudio.

Ele usa modulação de código de pulso diferencial adaptativo de sub-banda (SB-ADPCM), isso acaba exigindo uma taxa de bits de 64 Kbps. Esse foi o primeiro *codec* de banda larga emitido pelo ITU-T que suporta uma largura de banda de até 7 KHz. Também existem derivações como G.722.1 e G.722.2, porém não suportam taxa de bits 64 Kbps (DAENGSI et al., 2012). Em vídeo-chamadas, o G.722 é transportado no *payload* do protocolo RTP (SCHULZRINNE; CASNER, 2003).

2.4 Contêineres multimídia

Muitos confundem a definição de contêiner com *codec,* mas, na verdade, um contêiner é um conjunto de arquivos de áudio e vídeo que utilizam *codecs*. Entre os mais conhecidos estão o WebM, AVI, FLV, MP4 e MKV.

2.4.1 MP4

O formato MPEG-4 (ou MP4) faz parte do grupo MPEG (*motion Pictures Expert Group*). Ele é o formato de contêiner multimídia digital mais usado para armazenar vídeo e

[1] https://www.mpegla.com/programs/avc-h-264/faq/ - acesso em setembro 2019

[2] https://www.techopedia.com/definition/4465/g-722 - acesso em setembro 2019

áudio, podendo ser usado para armazenar outros tipos de dados, como legendas e imagens estáticas. O MP4 pode ser usado tanto com o objetivo de criar vídeos para *download*, como para *streaming*.

2.4.2 MKV

O MKV (*Matroska Video*) recebe esse nome em referências as bonecas russas que possuem outras dentro delas. Essa analogia é feita na medida em que os vídeos em MKV podem conter imagem, áudio e legendas num só arquivo, em diferentes formatos. É muito usado também para compressão de vídeos de alta resolução (1080p – FULL HD).

2.5 *Frameworks* de mídia

Por definição *frameworks* solucionam problemas recorrentes com abordagens genéricas. Sendo assim, eles funcionam como um conjunto de bibliotecas, ou componentes, para a construção da aplicação desejada. Nas subseções que seguem será apresentado os *frameworks* de mídia utilizados tanto na etapa de desenvolvimento quanto na etapa de validação.

2.5.1 FFmpeg

O FFmpeg é a principal estrutura multimídia que possui projeto gratuito e de código aberto. Iniciado em 2000, por Fabrice Bellard (*a.k.a*. Gérard Lantau), e continuado por Michael Niedermayer de 2004 a 2015, o projeto oferece alta portabilidade, baixas dependências de outras bibliotecas e máximo compartilhamento de código como principais objetivos.[1]

A principal aplicação é o homônimo do *framework,* FFmpeg. Essa aplicação processa com base em linhas de comando mídias de áudio e vídeo podendo também ser usado para transcodificação, dimensionamento, pós-processamento rápido e conformidade com padrões como o SMPTE e o ITU.

[1] http://ffmpeg.org/about.html - acesso em setembro 2019

2.5.2 GStreamer

O GStreamer, assim como o FFmpeg, é um *framework* multiplataforma para a construção de aplicações de mídia. Foi desenvolvido em 1999 por Erik Walthinsen, entre outros. Mas, o projeto só expandiu após a entrada de Wim Taymans, atual líder. Foi escrito em C e baseado em GObject. O GStreamer utiliza uma arquitetura de *plugins* e *bindings* que aumenta sua diversidade em bibliotecas de mídia e linguagens de programação, sendo, ainda, um software livre, licenciado sob GNU GPL.

Um fato relevante, e que torna o GStreamer ainda mais confiável, é sua estabilidade desde 2012.[1] Uma das ferramentas desse *framework,* que será abordada neste estudo, é o *gst-launch*. Ela é uma ferramenta de linha de comando para prototipagem e testes.

Na Figura 2.10 pode-se observar a grande expressividade do uso do GStreamer para aplicações multimídia. Em poucas linhas foi gerado a captura do vídeo do usuário (através do comando da linha 9), um *frame* de vídeo simulando o compartilhamento de conteúdo (através do comando da linha 11) e uma logomarca com uma imagem estática (usando o comando da linha 2). O resultado final é ilustrado na Figura 2.11.

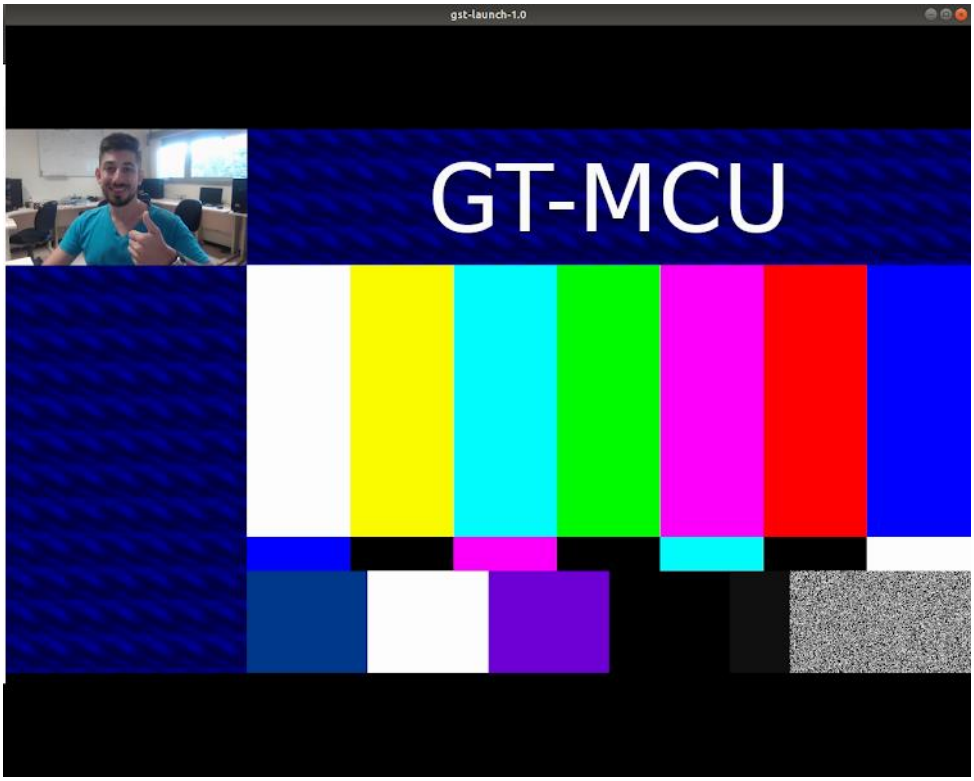
Figura 2.10 - *Pipeline* usando ferramenta *gst-launch*

```
1   gst-launch-1.0 videotestsrc pattern=black !
2                  gdkpixbufoverlay location=gtmcu.png !
3                  video/x-raw, framerate=\(fraction)1/1, width=1280, height=720 !
4                  compositor name=comp sink_0::xpos=0 sink_0::ypos=0
5                                       sink_1::xpos=0 sink_1::ypos=0
6                                       sink_2::xpos=320 sink_2::ypos=180 !
7                  video/x-raw, width=1280, height=720 !
8                  videoconvert !
9                  ximagesink v4l2src device=/dev/video0 !
10                 video/x-raw, width=320, height=180 !
11                 comp. videotestsrc !
12                 video/x-raw, width=960, height=540 !
13                 comp.
```

Fonte: O Autor

[1] https://en.wikipedia.org/wiki/GStreamer - acesso em setembro 2019

Figura 2.11 - Captura gerada pelo *pipeline* da Figura 2.10



Fonte: O Autor

2.6 Outras tecnologias

2.6.1 Plataforma de videoconferência MCU

O GT-MCU (Grupo de Trabalho em MCU) é um projeto do grupo de pesquisa PRAV (Projetos em Áudio e Vídeo) da UFRGS liderado pelo professor Valter Roesler. Foi desenvolvido em *software* livre, financiado pela RNP (Rede Nacional de Ensino e Pesquisa) e pela empresa Mconf Tecnologia.

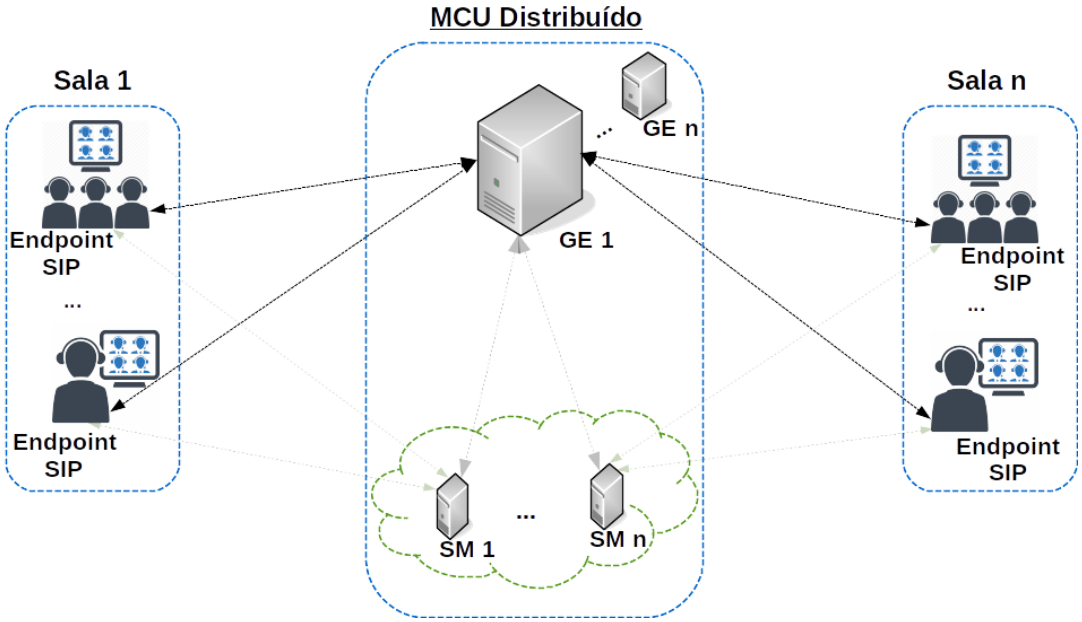
Esse projeto propõe um serviço de videoconferência de baixo custo totalmente virtualizado, funcionando em nuvem de forma escalável. A plataforma funciona praticamente como um *framework* universal de encaminhamento de mídias e não apenas como um MCU, o que acaba permitindo que outros serviços se integrem, como webconferência e VoIP.[1]

[1] http://www.inf.ufrgs.br/prav/mcu.htm - acesso em setembro 2019

O autor desta monografia foi o responsável por desenvolver, por completo, o sistema de gravação do GT-MCU, de maneira quase que genérica, a ponto de possibilitar que este sistema de gravação possa, futuramente, ser usado em outras plataformas, com poucas adaptações.

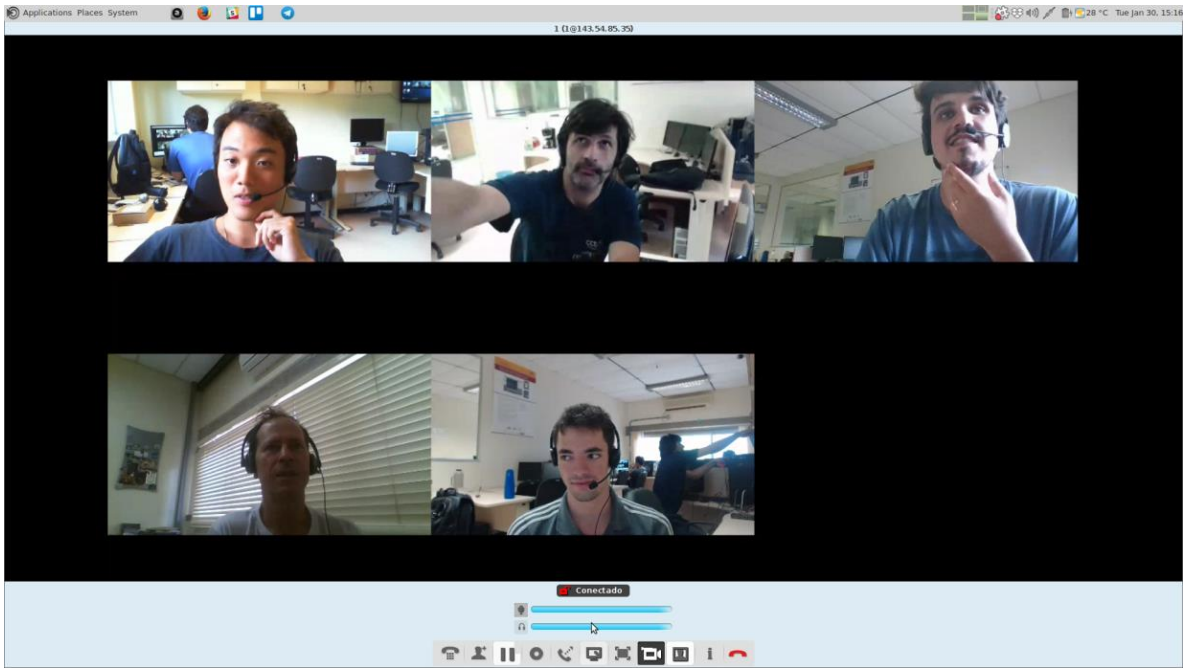
A Figura 2.12 mostra a visão geral do GT-MCU que é composto por um Gerenciador de Escalabilidade (GE) redundante e uma rede de Servidores de Mídia (SM) de MCUs em *software*, distribuídos em nuvem. A redundância do GE é necessária para que o sistema possua alta disponibilidade, visto que o GE gerencia toda a sinalização e possui a inteligência de todas as conferências em andamento, tornando-se um elemento crítico no sistema. A redundância dos Servidores de Mídia é necessária para, além de fornecer alta disponibilidade, aumentar o poder de processamento do sistema, suportando mais *endpoints*. Como visto na Figura 2.12, os *endpoints* conectam-se com o GE (linhas pretas), que é o ponto de entrada do MCU para os usuários. O GE escolhe um Servidor de Mídia disponível e direciona esse usuário (*endpoint*) para entrar na sala escolhida, fazendo com que toda a transferência de mídia (vídeo, áudio ou ambos) seja feita então diretamente entre SM e *endpoint* (ROESLER, V. et al., 2018). Um exemplo de resultado que será recebido pelos *endpoints* pode ser visto na **Erro! Fonte de referência não encontrada.**.

Figura 2.12 - Visão geral de um MCU distribuído



Fonte: (ROESLER, V. et al., 2018)

Figura 2.13 - Reunião da equipe do GT-MCU via videoconferência MCU



Fonte: (ROESLER et al., 2018)

2.6.2 Polycom

A Polycom é a empresa líder global em soluções de comunicações unificadas baseadas em padrão aberto para telepresença, vídeo e voz, impulsionada pela plataforma Polycom RealPresence (PORTAL TIBAHIA, 2012). Essa plataforma transita suas operações em diversos dispositivos e aplicativos sociais, móveis e de negócios.

Em 2018, a Polycom foi comprada pela Plantronics, Inc. (empresa californiana especializada em *headsets* e sistemas para atendimento telefônico) em uma transação valorada em dois bilhões de dólares aproximadamente (BUSINESS TECH, 2018). O resultado dessa união foi a empresa chamada de Poly (*do grego, "muito"*), uma empresa de tecnologia focada na experiência humana de comunicação e colaboração, com o objetivo de tornar a comunicação tão rica e natural quanto um encontro presencial (VOIT, 2019).

Algumas das soluções que possibilitam a gravação são o Polycom RealPresence Video Content Management, o Polycom RSS 4000 Recording and Streaming Server, o RealPresence Capture Station Pro, entre outras soluções que fazem da Polycom um referencial nessa área.

O primeiro, garante um acesso seguro a *streamings web* e gravações de vídeo sob demanda por meio de um portal na web. Já o RSS 4000, além de possuir uma interface baseada

em *web*, permite o uso de *streamings* e gravações de videoconferências em alta definição (HD) (RSS 4000, 2009).

Enquanto isso, o Station Pro é dispositivo de fácil utilização que grava e faz *streaming* de apresentações de qualquer ambiente. Ele sincroniza áudio, vídeo e slides de apresentação em tempo real (STATION PRO, 2014).

2.6.3 *Web Real-Time Communication* (WebRTC)

O WebRTC, escrito em C++ e JavaScript, é um projeto aberto e de *software* livre (sobre termos de licença BSD) que permite aos navegadores e aplicativos móveis comunicação em tempo real (RTC) por meio de *APIs* simples.[1] Foi padronizado pela W3C e pela IETF. A principal vantagem do WebRTC é a eliminação da necessidade de uso de *plugins* ou *downloads* de *softwares* nativos, diferente de outras soluções *web*.

2.6.4 Kurento *Media Server* (KMS)

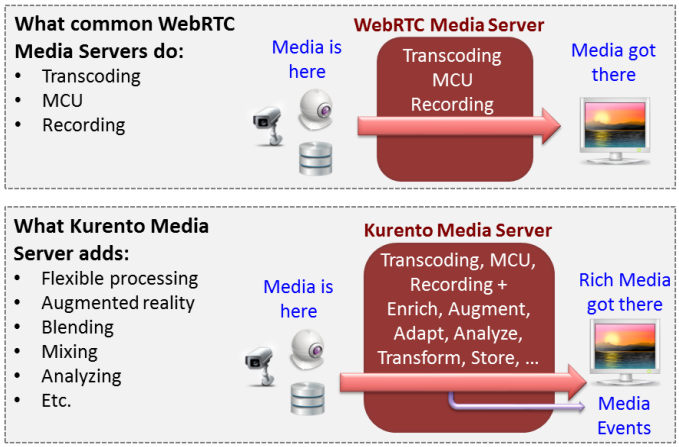
O KMS é um servidor de mídia WebRTC e um conjunto de *APIs* clientes que simplificam o desenvolvimento de aplicativos de vídeo avançados para as plataformas *web* e *mobile.* Além disso, possui código aberto que foi lançado sob os termos da *Apache License Version 2.0* e disponível no *GitHub*.

O KMS, desenvolvido em cima do GStreamer, é o elemento principal do Kurento por ser responsável pela transmissão, processamento, carregamento e gravação de mídia. Na Figura 2.14, retirada da página oficial do Kurento, pode ser visto as funcionalidades que o KMS suporta em relação a um servidor WebRTC comum.

Como mostra o retângulo superior da Figura 2.14, um WebRTC *Media Server* genérico apresenta três funcionalidades básicas: transcodificação, MCU e gravação. Já o KMS, visto no retângulo inferior, além dessas funções básicas, possui outras funcionalidades, tais como: mixagem, realidade aumentada, armazenamento, entre outras. Essas características acabam tornando o KMS um servidor mais atrativo.

[1] http://www.webrtc.org/ - acesso em setembro de 2019

Figura 2.14 - Comparação de funcionalidades entre um KMS e um WebRTC *Media Server*



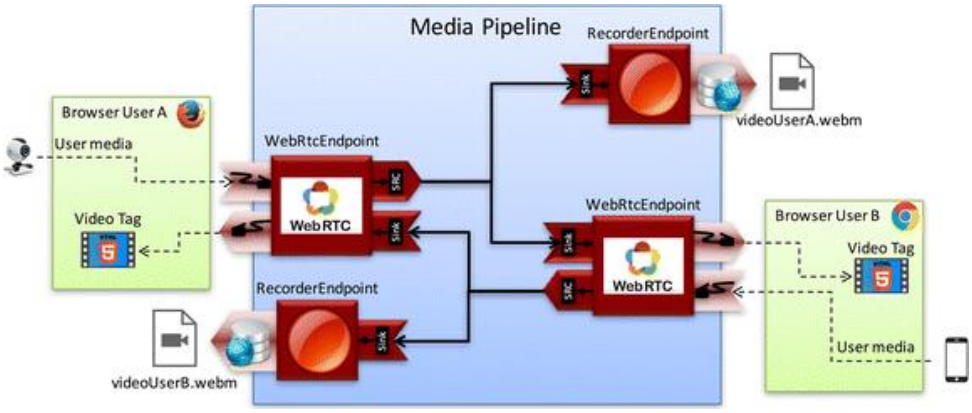
Fonte: (KURENTO, 2018)

Na sua página de documentação (KURENTO), o Kurento exibe uma demonstração de como funciona seu fluxo de operação em relação ao processo de gravação. Lá é detalhado, passo a passo, todo o desenvolvimento do código. Vale ressaltar que, diferente dos *hardwares* da *Polycom* citados anteriormente, essa solução é desenvolvida visando a plataforma *web*.

A ideia base é que o fluxo de mídias será enviado ao KMS que irá gravar o fluxo (usando o *RecorderEndpoint*) ao mesmo tempo que envia o fluxo de volta ao *endpoint* através do *WebRtcEndpoint*. Esses dois elementos fazem parte do *pipeline* de mídia criado para que todo esse processo funcione.

Na Figura 2.15 esse processo é ilustrado, onde pode ser visto um usuário de *browser* A, com uma *webcam*, se comunicando com um usuário *browser* B, com um *smartphone*, através do *WebRtcEndpoint*. E, ao mesmo tempo, cada um está gravando a comunicação através do *RecorderEndpoint*.

Figura 2.15 - Funcionamento de uma gravação usando KMS



Fonte: (LÓPEZ-FERNÁNDEZ et al., 2017)

3 DESENVOLVIMENTO DO PROTÓTIPO

Este capítulo trás todo o desenvolvimento do foco desta monografia: a construção de um sistema de gravação usado em videoconferências. Nele serão apresentadas as características do sistema proposto, o planejamento feito, a arquitetura do projeto, as características de cada bloco, os detalhes da implementação dos códigos e, por fim, os estudos de caso adotados. Vale ressaltar também que o desenvolvimento deste sistema de gravação genérico se faz necessário, inicialmente, para o uso no GT-MCU do grupo PRAV da UFRGS.

3.1 Características do sistema

O sistema de gravação desenvolvido apresentará as seguintes especificações:

- Ferramenta de gravação: FFmpeg;
- Formato de gravação: MKV;
- Qualidade da gravação: 720p (HD);
- Tempo de pós-processamento: Não há;
- Disponibilidade de gravação de múltiplas salas simultaneamente.

A escolha da ferramenta se deu por ela ser a mesma utilizada no código *mcs-rtp-client* (do projeto GT-MCU, onde *bots* são gerados para simular usuários nas vídeo-chamadas) no qual o gravador deste estudo foi baseado. A definição pela qualidade de vídeo, em HD, foi feita analisando o cenário mais comum de videoconferências, onde a qualidade padrão das chamadas é em alta definição. O formato MKV foi, na verdade, a segunda opção no desenvolvimento do sistema. Inicialmente, o escolhido havia sido o MP4, porém, graças à capacidade do MKV de "recuperar-se" de uma falha, continuando a gravação do momento onde ela parou, essa troca foi realizada.

E, as principais características do sistema proposto, que diferenciam ele dos gravadores comuns, são a ausência de pós-processamento do arquivo de gravação e a possibilidade de gravação simultânea de múltiplas salas. Sendo assim, é possível ter acesso aos arquivos de gravação no instante que a mesma for concluída. Ou, inclusive, graças ao suporte do MKV, acompanhar o conteúdo da videoconferência enquanto ela está sendo gravada.

3.2 Planejamento

O passo inicial foi definir a arquitetura do sistema de gravação. Para isso foi necessário decidir qual seria o comportamento do gravador e como ele se comunicaria com as demais estruturas através de estudos de caso.

Feito isso, construiu-se a ideia de uma página *web*, chamada como "portal", que pode funcionar, entre outras funções, como controle de início ou parada das gravações. Por fim, foi feito um estudo da biblioteca *mcs-rtp-client*.

A biblioteca *mcs-rtp-client* era usada para simular usuários RTP nos testes de vídeo-chamadas do GT-MCU. Ou seja, essa biblioteca possibilita a simulação de usuários (*bots*) RTP com fluxo de mídia (vídeo, áudio ou ambos) em qualquer sentido (apenas enviando, apenas recebendo ou enviando e recebendo os fluxos de mídia existentes). Com base nisso, o gravador construído funciona como um usuário "robô" de videoconferência (*bot*) que só recebe os fluxos de mídia da chamada e grava esses fluxos em um destino pré-determinado.

3.3 Arquitetura

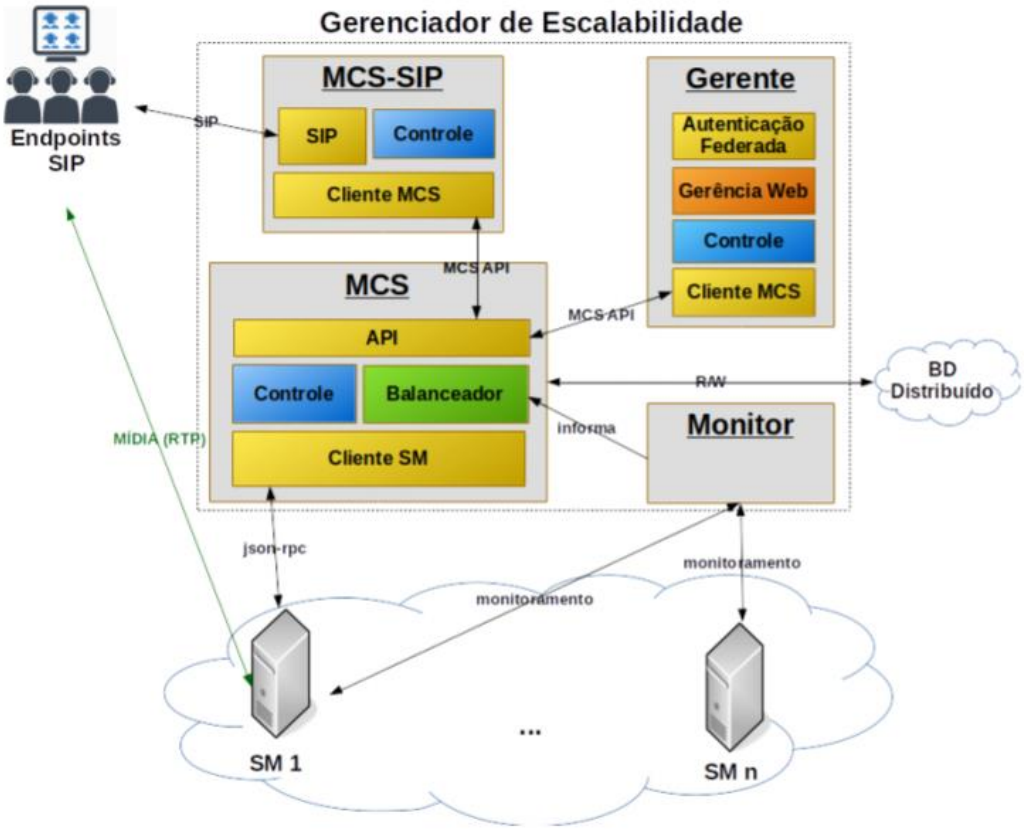
Como já mencionado, a ideia deste estudo foi desenvolvida inicialmente para a plataforma do GT-MCU. Sendo assim, a arquitetura do gravador está contida na arquitetura do GT-MCU do PRAV. A Figura 3.1 apresenta a arquitetura do MCU antes da adição do Gerenciador de Gravação e do Portal. Em relação a arquitetura do MCU, Roesler et al. (2018) resume os principais blocos do Gerenciador de Escalabidade da seguinte maneira:

- **MCS-SIP (*Media Control Server – Session Initiation Protocol*):** atua como recebedor das requisições SIP, traduzindo-as em chamadas de API para o servidor MCS. Além de traduzir as mensagens SIP para a API própria, esse bloco envia mensagens SIP *Redirect* para o *endpoint*, a fim dele trocar a mídia diretamente com o SM designado.
- **MCS (*Media Control Server*):** corresponde a uma aplicação e um *framework* para desenvolvimento de aplicações multimídia. Uma de suas principais funções é fazer o balanceamento de carga dos servidores de mídia disponíveis. Além disso, o MCS define um modelo de abstração que é independente do tipo de comunicação de mídia realizado (RTP, WebRTC, etc.) permitindo a gerência de salas, usuários e mídias. Dessa forma, um usuário de uma webconferência (acessando uma sala de MCU através do navegador WEB de um PC) é capaz de

se comunicar com um usuário de videoconferência (acessando a mesma sala de MCU através de um *endpoint* de videoconferência).

- **Gerente:** através de uma interface *web*, o Gerente permite o acesso para controle do sistema. Esse módulo permite a criação de salas, desde que o usuário tenha permissão para isso. Além disso, cada sala tem um gerente local (moderador), responsável por mudanças de *layout*, início de gravação, remover um usuário, colocar em mudo, entre outras. Esse módulo também permite a gerência local da sala. A comunicação com o MCS é através da mesma API mencionada anteriormente
- **BD Distribuído:** todos os dados instantâneos de todas as conferências em andamento (nome da sala, quem está na sala, *pipelines*, entre outros) ficam armazenados em um banco de dados distribuído a fim de aumentar a disponibilidade do sistema. Caso uma máquina virtual (*virtual machine* – VM) de GE fique indisponível, outra VM de GE assume, e utiliza os dados instantâneos que estão armazenados no BD distribuído, que é um elemento crítico do sistema para manter a alta disponibilidade.
- **Monitor:** atua como um servidor que tem conhecimento do estado atual da nuvem de servidores de mídia. Seu papel é manter uma conexão com todos os agentes (ou clientes) monitores que estão presentes em cada uma das máquinas servidores de mídia (SMs) da nuvem, bem como receber, armazenar e informar o MCS sobre novos eventos detectados por esses agentes. Essa informação é fundamental para que o balanceador de carga consiga tomar uma decisão sobre qual servidor de mídia uma nova sala de videoconferência deve utilizar.

Figura 3.1 - Arquitetura do GT-MCU antes do sistema de gravação e do Portal



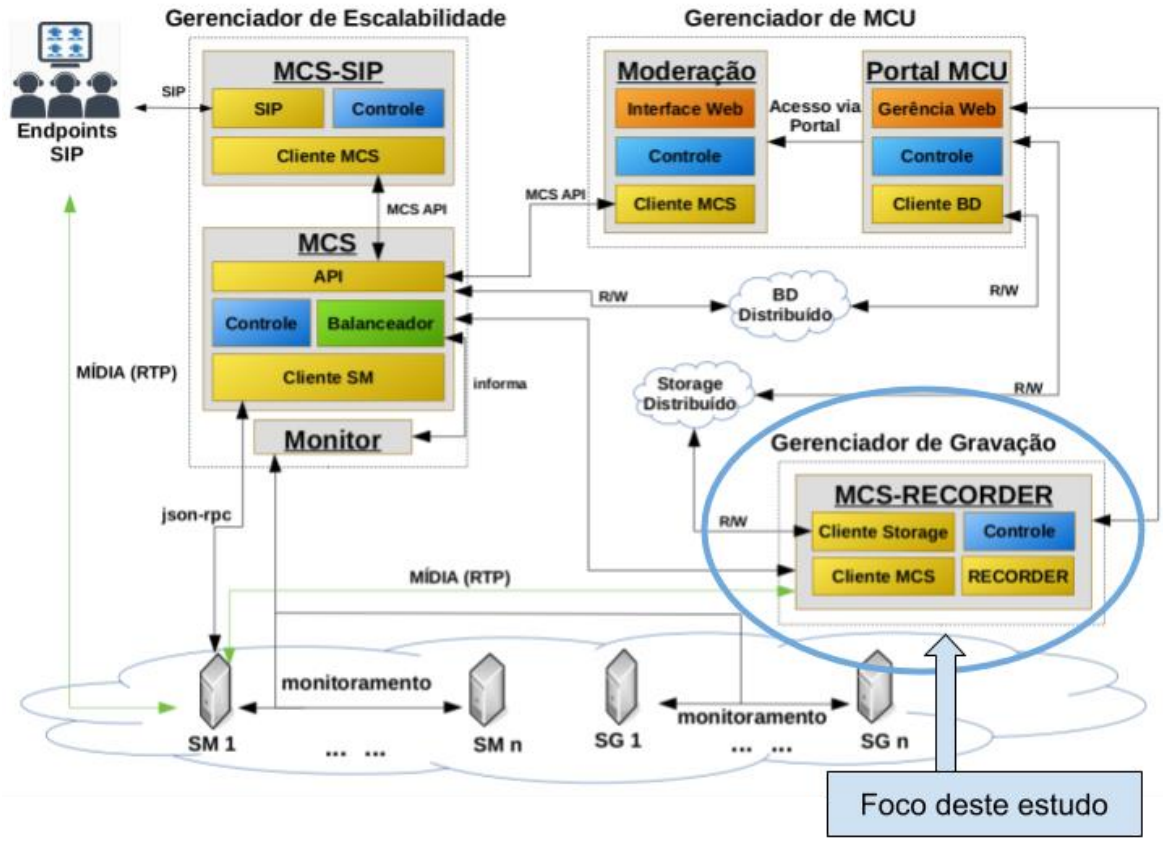
Fonte: (ROESLER et al., 2018)

Na Figura 3.2 é apresentada a arquitetura final do MCU após as mudanças feitas para este estudo. Explicando a Figura 3.2, pode ser observado que o bloco Gerente foi removido do GE e transformado em um Gerenciador de MCU com dois sub-blocos (moderação e Portal MCU) esses blocos dividem as funções que antes eram do Gerente, além das funções extras geradas pelo Gerenciador de Gravação.

O Monitor e os SMs também foram afetados nessa mudança. Agora existem também SGs (Servidores de Gravação) em nuvem que permitem a gravação de múltiplas salas ao mesmo tempo. Com isso, o Monitor passa a ter conhecimento do estado atual da nuvem não só dos servidores de mídia como também dos servidores de gravação. E finalmente, os SMs passam a enviar mídias RTP também para o gravador já que ele é visto pelos SMs como um usuário final.

O Gerenciador de Gravação, foco deste estudo, se comunica com o Gerenciador de Escalabilidade através do Cliente MCS e com o Gerenciador de MCU, mais especificamente, com o Portal, através do bloco Controle.

Figura 3.2 - Arquitetura final do MCU após adição do Gerenciador de Gravação e do Portal



Fonte: O Autor

3.4 Linguagens de programação

No desenvolvimento dessa ferramenta foram utilizadas duas linguagens de programação. Essas linguagens foram usadas em etapas distintas do processo. Para todo o bloco MCS-Recorder (que compreende o sistema de gravação proposto) que inclui, essencialmente, o Cliente MCS, o Controle e o Recorder foi utilizada a linguagem NodeJS com características de alto desempenho de CPU durante a execução de aplicações, facilidade de uso em *loop* de eventos, assincronicidade, entre outros fatores oferecidos pela linguagem.

Num segundo momento, para o desenvolvimento do bloco referente ao Portal, foi escolhida a linguagem *Ruby On Rails*, que trouxe uma certa facilidade para lidar tanto com a parte de banco de dados quanto com o *front-end* da página além da arquitetura com formato MVC (*Model-View-Client)*. A IDE escolhida para tal uso foi o *RubyMine* da *Jetbrains.*

3.5 Detalhes de implementação

A ideia central do sistema de gravação deste estudo é funcionar como um usuário "invisível" na chamada que vai gravar tudo o que passar por ele, seja áudio, vídeo e/ou conteúdo. Para isso foi implementado um *bot* (usuário "robô" de videoconferência) com característica *receive-only* (que recebe os fluxos de mídia, mas não envia nenhum).

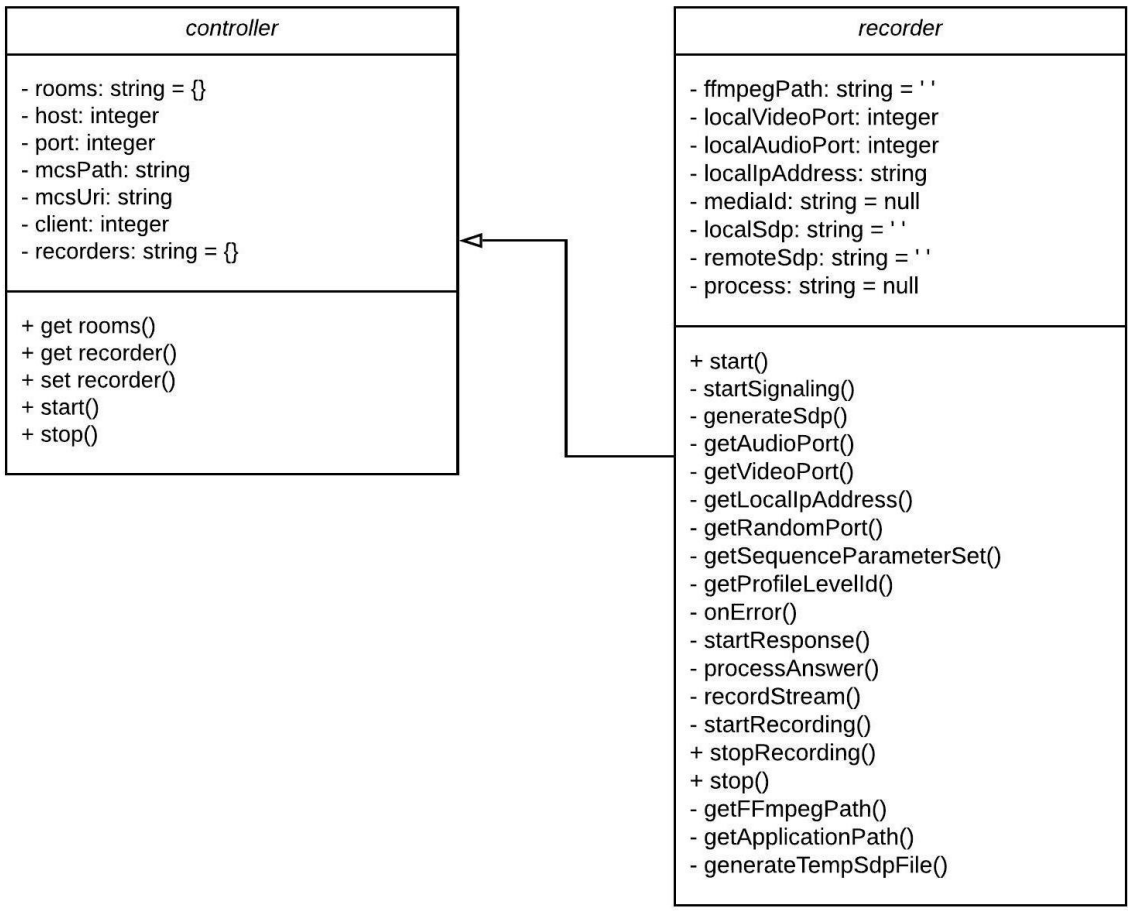
3.5.1 Gerenciador de gravação (MCS-Recorder)

O bloco desenvolvido pelo autor desta monografia e que compõe a sistema de gravação proposto pode ser dividido e explicado da seguinte forma:

- **Cliente MCS:** Responsável pela comunicação do gravador com o MCS. No caso da arquitetura do MCU sem essa comunicação seria inviável a utilização do MCS-Recorder.
- **Controle:** Responsável por interpretar os eventos que estiverem acontecendo e determinar que ação deve ser tomada. Sendo a principal delas, sinalizar para o MCS-Recorder o momento de iniciar (ou parar) uma gravação.
- ***Recorder*:** Responsável pelo processo de preparação e de gravação, propriamente dito. Isso inclui, a montagem do pacote SDP a ser enviado; o preparo e execução do comando de gravação; e também pelo seu armazenamento.
- **Cliente *Storage***: Responsável por armazenar e monitorar em nuvem os arquivos das gravações finalizadas.

A Figura 3.3 apresenta um diagrama de classes com a representação do sistema como um todo.

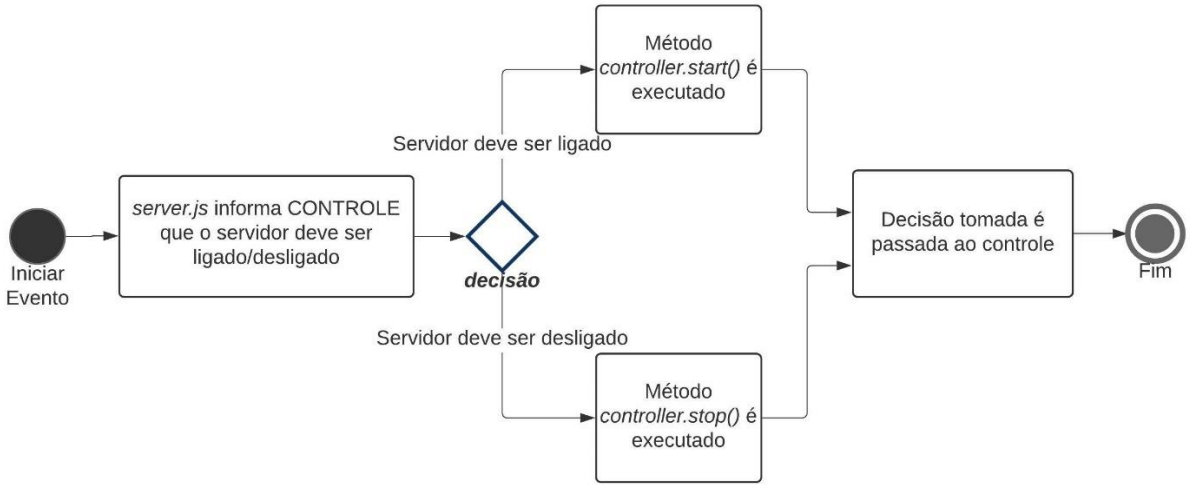
Figura 3.3 - Diagrama de classes do sistema



Fonte: O Autor

O bloco de controle foi desenvolvido sendo constituído por duas bibliotecas: uma para o servidor (*server.js*) e outra para o controle propriamente dito (*controller.js*). O servidor desenvolvido, tendo como dependência a biblioteca *controller.js*, é o responsável por avisar o controle que o servidor de gravação deve ser ligado ou desligado (esboçado na Figura 3.4). Para isso, ele utiliza os métodos *controller.start()* e *controller.stop(),* respectivamente.

Figura 3.4 - Fluxograma detalhando ações tomadas pela biblioteca *server.js*



Fonte: O Autor

O bloco de controle foi desenvolvido como uma classe orientada a eventos tendo como dependência a biblioteca *recorder.js* e o arquivo de configuração *config.js.* Após ser avisada pelo servidor (*server.js*), a classe *controller* determina suas ações baseadas em eventos e subeventos. São eles:

- ***open:*** esse evento indica que o sistema foi iniciado e, a partir disso, através do método *getRooms()* faz uma busca por todas as possíveis salas criadas.
- ***roomsList*:** na hierarquia, esse pode ser considerado um sub-evento, já que ocorre dentro de outro evento (*open*). Após o método *getRooms()* agir, todas as salas encontradas serão listadas com seu número de identificação correspondente (*roomId)* e para cada uma que satisfizer a condição de pelo menos uma mídia ativa será criado um objeto da classe *Recorder* referente ao *roomId* dessa sala e então o método *recorder.start()* dará início a gravação da mesma. Esse evento foi desenvolvido pensando no estudo de caso 2 (subseção 3.6.2), que será detalhado mais à frente, onde o *recorder* é iniciado após existirem salas criadas e ativas.
- ***mediaConnected:*** esse evento ocorre quando um usuário (uma mídia) conecta em alguma sala. Primeiro será testado se a sala existe, senão, será criada. O segundo teste é, se na sala informada, a gravação está ativa, o controle apenas informa que uma mídia entrou na sala, porém, se a sala estiver vazia, ou apenas com a mídia do gravador, será criado um objeto da classe *Recorder* referente ao

roomId dessa sala e então o método *recorder.start()* dará início a gravação da mesma.

- ***mediaDisconnected:*** quando uma mídia deixar uma sala, esse será o evento que será interpretado pelo controle. Ele irá remover a mídia da sala indicada pelo *roomId* e informar que uma mídia deixou a sala. E, no caso de ser a última mídia (além da mídia do gravador), além de removida essa mídia, uma possível mídia de gravação (se existir) também será removida através do método *recorder.stop()*.
- ***close:*** esse evento serve principalmente para trazer robustez ao sistema. Em caso de falha do MCS, uma mensagem de falha será mostrada e o gravador será fechado parando toda e qualquer gravação ativa graças ao método *recorder.stopRecording()*.
- ***unpublishedAndUnsubscribed:*** esse evento é muito semelhante ao evento *close,* porém, ele é usado em caso de falhas do KMS.
- ***error*****:** A função do *error* é informar quando ocorrer uma falha, notificando qual erro aconteceu.

A classe *Recorder* foi desenvolvida, como já mencionado, levando em conta a lógica da biblioteca *mcs-rtp-client* (do GT-MCU) que simula usuários *bots* para as chamadas. Dessa maneira, toda a parte de comunicação foi reaproveitada e a parte do tratamento da gravação foi adicionada. Tendo em vista isso, será detalhado apenas o tratamento da gravação, já que este é o foco do trabalho. Detalhando os métodos que, de fato, efetuam a gravação temos:

- ***recordstream():*** Nessa etapa, após toda a comunicação já estabelecida, é testado se o SDP da mídia do usuário existe e é válido. Se sim, um arquivo temporário deste tipo é criado. Após isso, as informações necessárias (argumentos) são coletadas e passadas para o método *startRecording.*
- ***startRecording():*** Esse método, através do módulo *child_process*, reproduz o comando passado como parâmetro, no caso o *appPath + args*, o que dará início a gravação.
- ***stopRecording():*** como visto no método anterior, a gravação ativa, nada mais é que um processo em execução. Sendo assim, esse método apenas para o processo ativo, encerrando a gravação.

- ***stop():*** Diferente do método anterior, o método *stop()* não só para as gravações como encerra o servidor de gravação, removendo todas as mídias dos usuários do mesmo.

3.5.2 Portal

O portal funciona como uma página *web* para controle das salas de gravação de usuário (normalmente, administrador ou moderador das mesmas). A ideia por trás dessa página é permitir dinamicidade nas salas, alterando a opção que habilita (ou desabilita) a capacidade de gravá-las sempre que preciso. Além de facilitar o acesso às gravações feitas, tanto por *download* como a opção de assisti-la na própria página. O portal já existia para a aplicação do GT-MCU, a contribuição do autor desta monografia nele foi a adição das funções referentes ao processo de gravação.

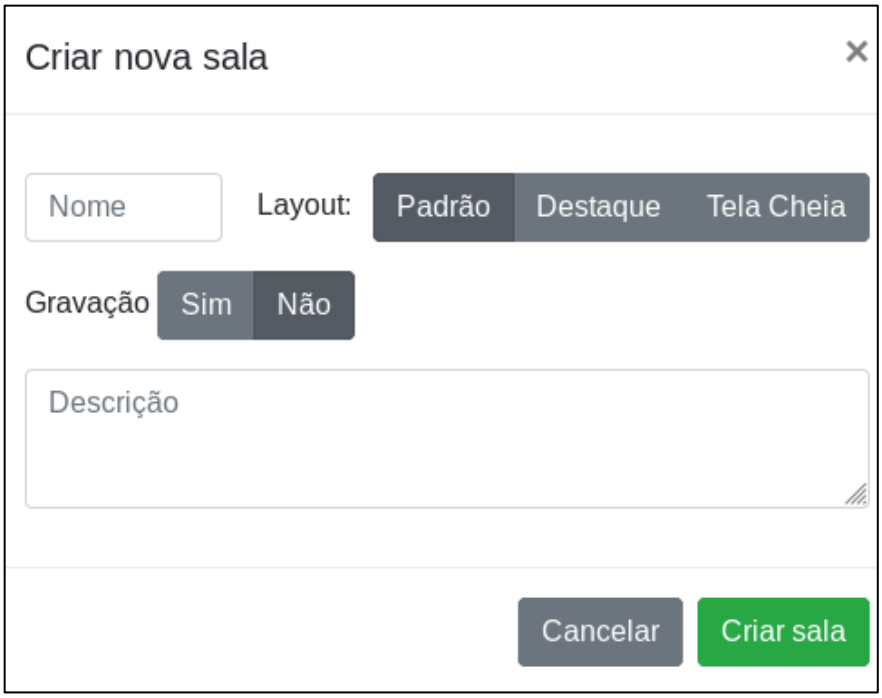
Na FIGURA é apresentado o que o usuário vai encontrar após acessar a página com seu *login* e senha. A página principal conta com a lista de salas pertencente àquele usuário. Nela constam informações da sala como nome, padrão do *layout* e *status* da gravação (número "2" na figura). Pode ser observado também uma aba "Gravações" (número "1" na figura) que levará para a página de gravações do usuário. Além disso, há também dois botões referentes à criação de novas salas (número "4" na figura) e edição de uma sala existente (número "3" na figura).

Figura 3.5 - Primeira visão do portal

Fonte: O Autor

A Figura 3.6 faz referência a interação com o botão “Nova Sala” da Figura 3.5. Ao clicar no botão mencionado uma janela surgirá com as opções para criação de uma nova sala. Essas opções são: Nome, *Layout*, Gravação e Descrição. O *Layout* por convenção sugere o tipo “Padrão” enquanto, na mesma lógica, a Gravação é sugerida como “Não”. Ambas opções podem ser editadas no futuro. Após criar a sala, ela aparecerá na listagem de salas da Figura 3.5.

Figura 3.6 - Janela para criação de nova sala



Fonte: O Autor

A Figura 3.7 faz referência a interação com botão “Editar” da Figura 3.5. Ao clicar no botão mencionado uma janela surgirá com as informações atuais da sala escolhida. Essas opções são: Nome, *Layout*, Gravação e Descrição. Após alteradas e salvas, as novas opções passam a ter efeito. Vale ressaltar que uma sala estando ativa, as alterações feitas nela passam a valer a partir daquele momento. Ou seja, alterações de *layout*, ou estado de gravação, tem efeito instantâneo. Sendo assim, se uma sala estava sendo gravada e o moderador editá-la e mudar a opção de gravação de “SIM” para “NÃO” um evento será emitido e o Gerenciador de Gravação irá parar de gravar a sala. O oposto também é verdadeiro.

Figura 3.7 - Janela de edição de uma sala que pode ser alterada com a sala em andamento

Fonte: O Autor

3.6 Casos de uso

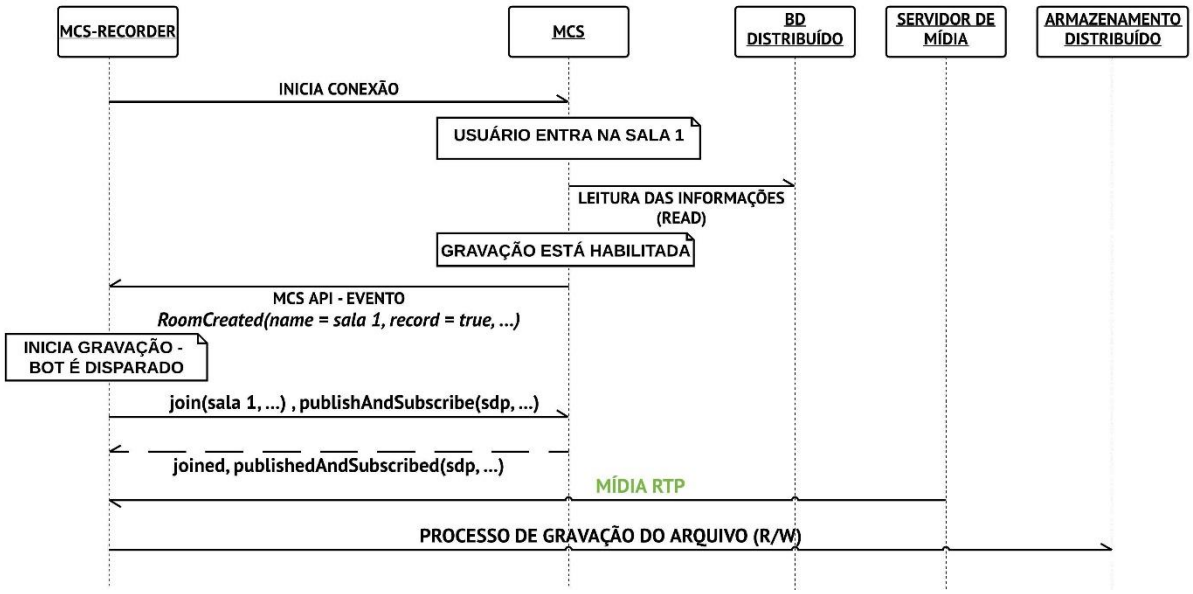
Foram avaliados dois possíveis casos de uso para o sistema de gravação. No primeiro deles, a gravação é ativada no início dos tempos, junto com a criação da sala. Já no segundo, foi feito o diagrama de sequência ativando a gravação da sala após ela ter sido iniciada e já conter usuários.

3.6.1 Caso de uso 1

Nessa primeira situação, o MCS-Recorder é iniciado juntamente com todos outros servidores (MCS, MCS-SIP e KMS). O MCS-Recorder se conecta ao MCS e faz uma primeira busca por salas com usuários. Após isso, aguarda a entrada de algum *endpoint*.

Quando um usuário se conecta na videoconferência, O MCS faz uma leitura no banco de dados para saber se a sala do usuário possui gravação ativa ou não. Em caso afirmativo, ele envia um evento *RoomCreated* para o MCS-Recorder com todos os parâmetros dessa nova sala. Então, em posse dessas informações, o MCS-Recorder inicia o *bot* para essa sala, que terá a missão de gravar todo o fluxo de mídia recebido pelo servidor de mídia até que não haja mais nenhum usuário na sala informada. Todo esse cenário é mostrado no diagrama da Figura 3.8.

Figura 3.8 - Diagrama 1: Usuário entra na sala com gravação habilitada

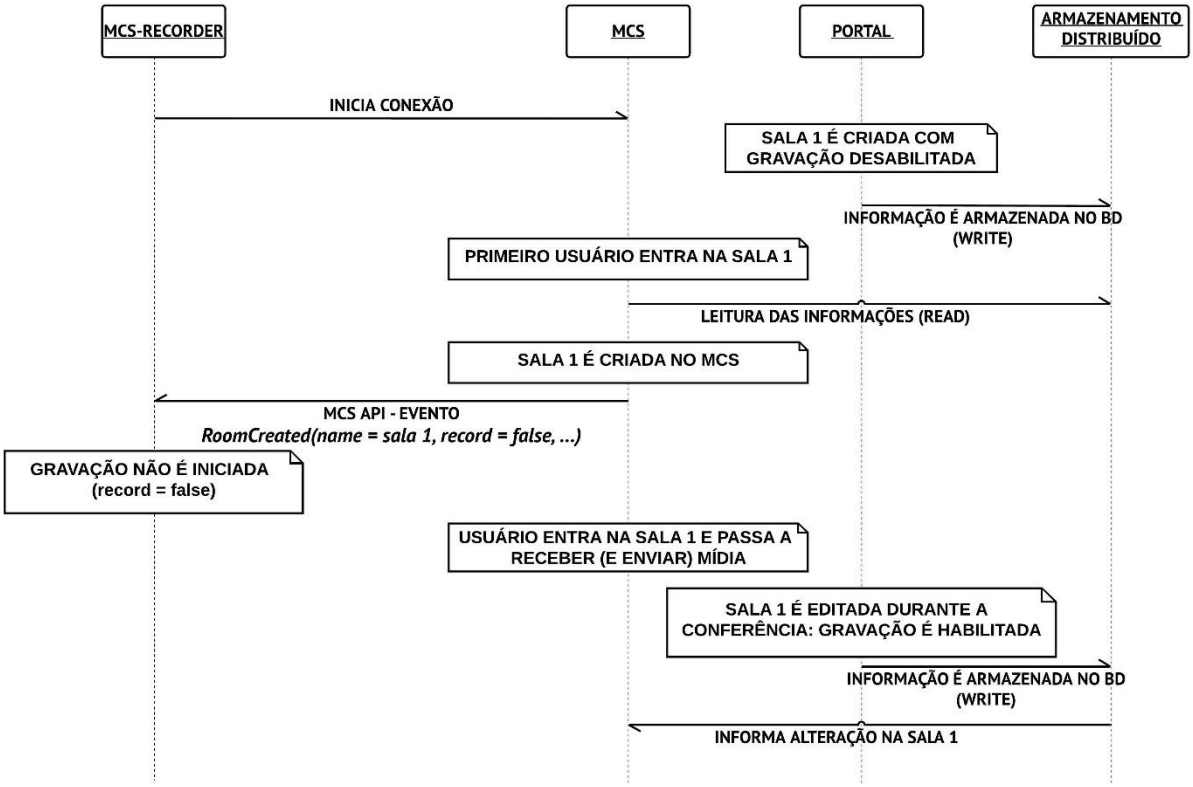


Fonte: O Autor

3.6.2 Caso de uso 2

Nesse caso, inicialmente, um moderador acessa o portal e cria uma sala com a opção de gravação desativada que é armazenada no BD distribuído. No momento que o primeiro usuário entrar na sala, o MCS vai procurar as informações da sala no BD e criar ela no MCS. Após isso, ele informa ao MCS-Recorder que a opção gravação está desativada. Sendo assim, o MCS-Recorder não executa nenhuma ação até que essa *flag* seja alterada. Isso é mostrado no diagrama da Figura 3.9.

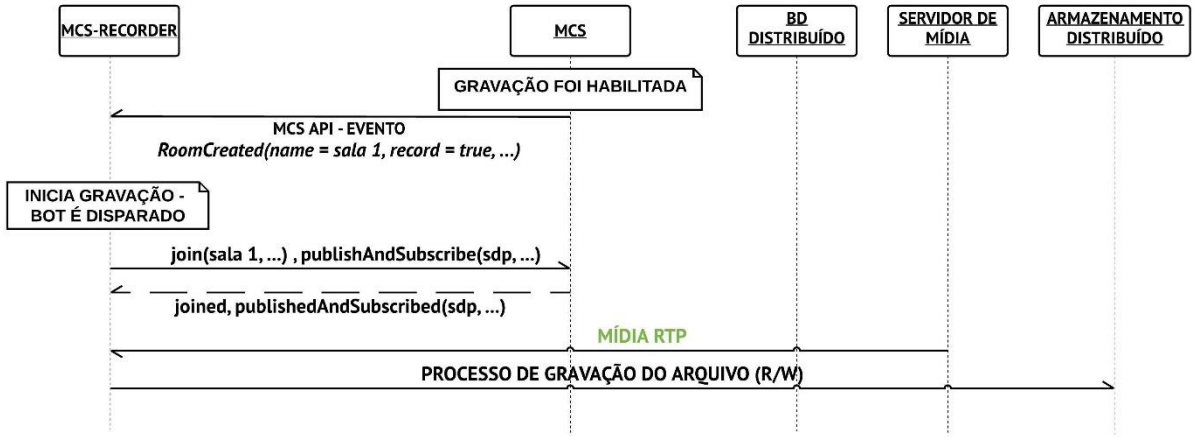
Figura 3.9 - Diagrama 2: Conferência iniciando sem gravação



Fonte: O Autor

Todavia, no momento em que a sala for editada no Portal, alterando essa variável e atualizando o BD, o MCS será avisado da mudança e repassa a informação da sala para o MCS-*Recorder* que então iniciará o processo de gravação até que seja finalizado, como exemplificado no diagrama da Figura 3.10.

Figura 3.10 - Diagrama 3: Evento da gravação após ativação via portal



Fonte: O Autor

4 AVALIAÇÃO DO PROTÓTIPO

Neste capítulo será detalhado o estudo dos testes feitos entre o sistema desenvolvido neste trabalho e outros *hardwares* e ferramentas conhecidos. Será apresentado o objetivo, a metodologia usada, os ambientes e ferramentas testados, as métricas avaliadas e, ao final, os resultados obtidos.

4.1 Objetivo

O objetivo dessa avaliação é verificar o desempenho de alguns tipos de *hardwares* como equipamentos para gravação de videoconferências em quatro cenários. Dois desses cenários utilizam o *framework* GStreamer, que funcionará como o *baseline* dos testes. Os outros dois utilizarão o sistema de gravação desenvolvido para esta monografia. Foram escolhidos estes quatro cenários para que se pudesse comparar o sistema desenvolvido com uma ferramenta que também é muito utilizada para fazer gravações (o *gst-launch* do GStreamer).

4.2 Metodologia e métricas avaliadas

A ferramenta desenvolvida foi executada com, e sem, transcodificação em três ambientes: uma máquina local, um minicomputador Intel NUC e num *hardware* com uma GPU. Ela foi comparada com outra ferramenta: o *gst-launch* do GStreamer. Essa outra ferramenta é capaz de, através linhas de comando, gerar *pipelines* que também executam a gravação de vídeo. Esses *pipelines* serão detalhados na seção 4.3.

Para verificar o desempenho de cada equipamento foi escolhida a métrica de número de gravações simultâneas. No total, cada um dos cenários foi repetido seis vezes para cada ambiente, variando o número de gravações simultâneas em 2, 4, 8, 16, 32 e 64 através da utilização de *scripts* criados para facilitar o processo. Foram comparadas e consideradas métricas dos testes: o consumo de CPU pelo número de gravações simultâneas, o tamanho de arquivos gerados e a qualidade das gravações, em cada ambiente, variando os cenários.

4.3 Ambientes

Aprofundando um pouco mais a metodologia, nesta seção, serão descritos os ambientes (*hardwares*) e ferramentas utilizadas, além das métricas avaliadas. Começando pelos ambientes, destacam-se as seguintes características do primeiro *hardware*:

- Processador i7-7700HQ - 8 Core(s)
- Memória RAM de 16GB
- Sistema de arquivos: ext4
- Sistema Operacional Ubuntu 18.04.3 LTS - 64 bits
- Motivação da escolha: Essa máquina local foi escolhida para os testes por se tratar da máquina onde o MCS-Recorder foi inicialmente desenvolvido.

O segundo ambiente é representado por um minicomputador Intel NUC com as seguintes características:

- Processador i7-5557U - 8 Core(s)
- Memória RAM de 8GB
- Sistema de arquivos: ext4
- Sistema Operacional Ubuntu 18.04 LTS - 64 bits
- Motivação da escolha: Esse *hardware* foi escolhido por ser uma máquina com configurações semelhantes as da máquina local do ambiente 1 e por ser um produto comercial. Isso dará uma resposta aproximada de como o sistema de gravação desenvolvido responderá em máquinas já consolidadas no mercado.

O terceiro, e último ambiente testado, trata-se de um *hardware* NVIDIA Jetson TX2 com as seguintes características:

- Processador ARMv8 (64-bit) HMP CPU Complex – 6 Core(s)
  - NVIDIA Denver 2 Dual-core CPU
  - ARM Cortex-A57 Quad-core CPU
- Memória RAM de 8GB na CPU
- Sistema de arquivos: ext4
- Sistema Operacional Ubuntu 16.04 LTS - 64 bits
- Motivação da escolha: o MCS-Recorder utiliza somente recursos de CPU para a execução de suas aplicações, de modo que pode ser possível a alocação de parte do processamento na GPU, aumentando (ou diminuindo) assim a capacidade final da videoconferência, em número de usuários. Sendo assim, os testes irão comparar o desempenho dos cenários afim de confirmar essa hipótese.

4.4 Ferramentas

Nesta seção são apresentadas as ferramentas, tanto de *hardware* quanto de *software* que auxiliaram no processo de validação.

4.4.1 Aplicações

- ***gst-launch*** - Aplicação capaz de criar os cenários descritos na seção 4.5, através da criação de um *pipeline* de processamento de vídeo.
- ***Top*** - Aplicação de Monitoramento da CPU usando os filtros *"| grep gst"* e *"| grep ffmpeg"* para GStreamer e para o *Recorder*, respectivamente.
- ***ls*** – Comando usado para listar o tamanho dos arquivos através do parâmetro "*-sh"* que exibirá os valores em MB.

4.4.2 Hardwares

1. ***Webcam* Logitech C920** – Escolhida por fornecer vídeo já compactado no formato H.264 e pela alta qualidade de vídeo. A Figura 4.1 ilustra a ferramenta escolhida.

Figura 4.1 - *Webcam* Logitech c920

Fonte: Site oficial da Logitech

4.5 Cenários testados e resultados obtidos

Apresentados os objetivos, e as ferramentas utilizadas para a validação do sistema de gravação, nesta seção, serão detalhados os cenários desenhados para cada um dos testes e os resultados obtidos em cada um deles. Basicamente, serão quatro cenários onde, em cada um, um ambiente (seção 4.3) passará pelas quatro situações mencionadas na seção 4.1. Nos casos

onde o *Recorder* desenvolvido para essa monografia for testado, serão utilizados *bots* a partir de outra máquina, o que não afetará no cálculo do uso de CPU do mesmo.

4.5.1 Cenário 1 – GStreamer COM transcodificação

Como mencionado na seção 4.2, foram desenvolvidos *scripts* para agilizar o processo de testes. Na Figura 4.2 e na Figura 4.3 pode ser observado as linhas de comando utilizadas tanto para o ambiente 1 (máquina local) e ambiente 2 (NUC), quanto para a TX2 (ambiente 3). Vale ressaltar que para o ambiente 3 foram utilizados um *encoder* e um *decoder* diferentes do que nos outros dois ambientes, por conta da arquitetura da NVIDIA. Foram utilizados o "omxh264enc" e o "omx264dec" para substituir, respectivamente, o "x264enc" e o "avdec_h264". Essa troca se faz necessária para que a Jetson TX2 tenha um desempenho condizente com o dos outros *hardwares.*

Figura 4.2 - *Script* usando GStreamer COM transcodificação

```
1   gst-launch-1.0 -e v4l2src device=/dev/video0 !\
2   video/x-h264,width=1280,height=720,framerate=30 !\
3   tee name=tee1 ! queue ! avdec_h264 !\
4   videoscale ! video/x-raw,width=640,height=360,framerate=30/1 ! x264enc !
5   matruskamux ! filesink location=/tmp/tcc/gst-trans/2/0.mkv\
6   tee1.! queue ! avdec_h264 ! videoscale ! video/x-raw,width=640,height=360,
    framerate=30/1 ! x264enc ! matruskamux !
7   filesink location =/tmp/tcc/gst-trans/2/1.mkv\
```

Fonte: O Autor

Explicando a linha de comando pode-se observar que foi usado como fonte de vídeo a *webcam* do ambiente em questão (linha "1" da figura). Foram definidas as propriedades do vídeo (linha "2" da figura) e as informações de como será salvo o vídeo, passando pelos *encoders e decoders*, até o destino *(*linhas "3", "4" e "5"). As linhas subsequentes referem-se ao segundo *pipeline* do comando (idêntico ao anterior) que executará a segunda gravação.

Figura 4.3 - *Script* específico para a transcodificação do TX2

```
1   gst-launch-1.0 -e v4l2src device=/dev/video0 !\
2   video/x-h264,width=1280,height=720,framerate=30 !\
3   tee name=tee1 ! queue ! omx264dec !\
4   videoscale ! video/x-raw,width=640,height=360,framerate=30/1 ! omxh264enc !
5   matruskamux ! filesink location=/tmp/tcc/gst-trans/2/0.mkv\
6   tee1.! queue ! omx264dec ! videoscale ! video/x-raw,width=640,height=360,
    framerate=30/1 ! omxh264enc ! matruskamux !
7   filesink location =/tmp/tcc/gst-trans/2/1.mkv\
```

Fonte: O Autor

A mesma explicação vale para a Figura 4.3, a única e principal diferença são os *encoders* e *decoders* diferentes. Nesse caso, foram utilizados o "omxh264enc" e o "omx264dec" para substituir, respectivamente, o "x264enc" e o "avdec_h264", como já mencionado.

Tabela 4.1 - Resultados obtidos para a máquina local no Cenário 1

MÁQUINA LOCAL						
Nº GRAVAÇÕES	**2**	**4**	**8**	**16**	**32**	**64**
% CPU	3,70%	44,60%	70,15%	96,01%	97,32%	100,00%
TAM. ARQUIVO	12 MB	13 MB	12 MB	8,1 MB	4,3 MB	200 KB

Fonte: O Autor

Tabela 4.2 - Resultados obtidos para o NUC no Cenário 1

NUC						
Nº GRAVAÇÕES	**2**	**4**	**8**	**16**	**32**	**64**
% CPU	40,45%	95,41%	99,58%	99,70%	99,70%	99,72%
TAM. ARQUIVO	13 MB	13 MB	7,2 MB	3,7 MB	1,9 MB	1,2 MB

Fonte: O Autor

Tabela 4.3 - Resultados obtidos para a Jetson TX2 no Cenário 1

JETSON TX2						
Nº GRAVAÇÕES	**2**	**4**	**8**	**16**	**32**	**64**
% CPU	14,32%	31,56%	40,41%	85,27%	90,43%	87,56%
TAM. ARQUIVO	25 MB	24 MB	23 MB	25 MB	16 MB	9,1 MB

Fonte: O Autor

Tabela 4.4 – Comparação entre os resultados obtidos em cada ambiente

CENÁRIO 1 – GSTREAMER COM TRANSCODIFICAÇÃO						
Nº GRAVAÇÕES	**2**	**4**	**8**	**16**	**32**	**64**
% CPU	A1	A3	A3	A3	A3	A3
TAM. ARQUIVO	TODOS	TODOS	A1 - A3	A3	NENHUM	NENHUM

Fonte: O Autor

Levando em conta as configurações dos ambientes, já apresentadas na seção 4.3, e os resultados encontrados nas execuções e apresentados nas Tabela 4.1, Tabela 4.2, Tabela 4.3 e Tabela 4.4 pode-se dizer que tudo saiu como esperado. Sendo mais claro, analisando a Tabela 4.4 que traz o resumo dos testes do cenário 1, mostrando qual ambiente foi melhor em cada situação testada, observa-se que, em relação ao desempenho (consumo de CPU), os melhores resultados foram mesmo da Jetson TX2, devido ao auxílio da sua GPU, reduzindo o gasto energético da CPU.

Um fato curioso é que apenas na primeira situação testada o ambiente 3 (A3) foi menos eficiente que outro ambiente, no caso, o ambiente 1 (A1 – máquina local). Isso pode ser observado já que com apenas duas gravações simultâneas o consumo de CPU ainda é baixo, o que fez com que na Jetson TX2, o auxílio da GPU não fosse utilizado nesse caso.

Observando os tamanhos de arquivos gerados, nota-se que conforme o consumo de CPU aumenta muito, mais pacotes vão sendo perdidos, o que vai piorando a qualidade da gravação e, ao mesmo tempo, gerando arquivos cada vezes menores até o momento em que não chegam pacotes suficientes nem para abrir o arquivo. Isso fica evidente em gravações com vídeos travados ou com a imagem muito "*pixelada*".

Levando em conta as tabelas, na Tabela 4.4 é visto que para as duas primeiras situações (2 e 4 gravações simultâneas) todos os ambientes responderam com êxito. Já no caso de 8 gravações simultâneas, o A2 (ambiente 2 – NUC) apresentou problemas na gravação. No caso seguinte (16 gravações simultâneas), foi a vez do A1 falhar. E, por fim, a partir de 32 gravações simultâneas, no cenário 1, todos os ambientes apresentaram problemas referentes a perda de pacotes. Lembrando que o ambiente 3 apresenta arquivos com tamanhos diferentes dos outros dois por usar *encoders* e *decoders* específicos para sua arquitetura (explicado anteriormente nesta seção).

4.5.2 Cenário 2 – Gstreamer SEM transcodificação

A opção por esse cenário se dá justamente para tornar a comparação entre as ferramentas ainda mais justa. Dessa forma, podemos comparar o GStreamer com ele mesmo, alterando apenas a opção da transcodificação, e também compará-lo com outro sistema, no caso o MCS-Recorder que também tem a opção de utilizar, ou não, a transcodificação. Na Figura 4.4, é apresentado o *script* utilizado para os três ambientes.

Figura 4.4 - *Script* para o GStreamer SEM transcodificação

```
1   gst-launch-1.0 -e v4l2src device=/dev/video0 !\
2   video/x-h264,width=1280,height=720,framerate=30,clock-rate=90000 !\
3   tee name=tee1 ! \
4   queue max-size-buffers=0 max-size-bytes=0 max-size-time=1000000000 !\
5   h264parse ! matruskamux ! filesink location=/tmp/tcc/gst/2/0.mkv \
6   tee1.!
7   queue max-size-buffers=0 max-size-bytes=0 max-size-time=1000000000 !\
8   h264parse ! matruskamux ! filesink location=/tmp/tcc/gst/2/1.mkv \
```

Fonte: O Autor

A principal diferença entre esse *script* e o *script* anterior (com transcodificação) está na presença de dois elementos: o "*h264parse*" e uma fila "*queue*". Na linha "4" da Figura 4.4 aparecem os parâmetros da fila que limitam o tamanho máximo dela, quando a fila atingir o mais restritivo desses valores ela: bloqueia ou descarta os pacotes que ainda estão nela. No caso da Figura 4.4 apenas o tempo é um limitante.

Tabela 4.5 - Resultados obtidos para a máquina local no Cenário 2

MÁQUINA LOCAL						
Nº GRAVAÇÕES	**2**	**4**	**8**	**16**	**32**	**64**
% CPU	0,55%	1,06%	2,28%	4,28%	6,97%	11,47%
TAM. ARQUIVO	22 MB	22 MB	22 MB	22 MB	22 MB	22 MB

Fonte: O Autor

Tabela 4.6 - Resultados obtidos para o NUC no Cenário 2

NUC						
Nº GRAVAÇÕES	**2**	**4**	**8**	**16**	**32**	**64**
% CPU	1,20%	2,32%	3,77%	6,71%	9,28%	14,40%
TAM. ARQUIVO	22 MB	22 MB	22 MB	22 MB	22 MB	22 MB

Fonte: O Autor

Tabela 4.7 - Resultados obtidos para a Jetson TX2 no Cenário 2

JETSON TX2						
Nº GRAVAÇÕES	**2**	**4**	**8**	**16**	**32**	**64**
% CPU	0,40%	0,80%	1,45%	3,10%	6,47%	13,26%
TAM. ARQUIVO	22 MB	22 MB	22 MB	22 MB	22 MB	22 MB

Fonte: O Autor

Tabela 4.8 – Comparação entre os resultados obtidos em cada ambiente

CENÁRIO 2 – GSTREAMER SEM TRANSCODIFICAÇÃO						
Nº GRAVAÇÕES	**2**	**4**	**8**	**16**	**32**	**64**
% CPU	TODOS	TODOS	TODOS	TODOS	TODOS	TODOS
TAM. ARQUIVO	TODOS	TODOS	TODOS	TODOS	TODOS	TODOS

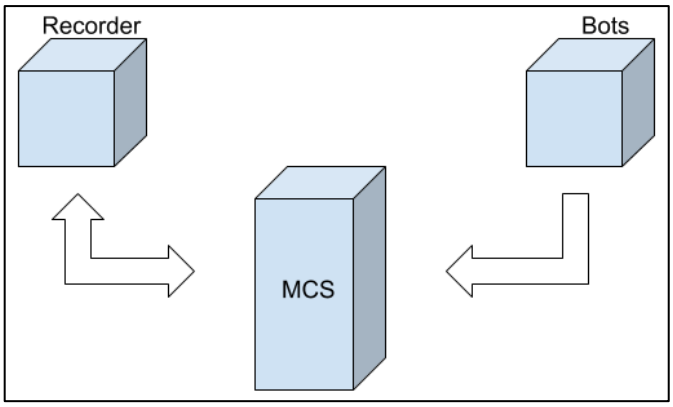
Fonte: O Autor

A Tabela 4.8 resume o uso do GStreamer sem transcodificação. Analisando ela e as tabelas anteriores (Tabela 4.5, Tabela 4.6 e Tabela 4.7) observa-se que o ausência de transcodificação, nestes testes, permitiu com que todas as situações testadas obtivessem êxito em todos os *hardwares* e com uma semelhança muito grande. O consumo de CPU em todos os *hardwares* foi muito baixo, podendo ser desconsiderado. E, o tamanho dos arquivos gerados se manteve constante em todos os casos, o que demonstra que em nenhum momento houve perda de pacotes ou degradação na qualidade do vídeo, tanto com poucas gravações simultâneas quanto com muitas.

4.5.3 Cenário 3 – MCS-Recorder COM transcodificação

O terceiro cenário ilustra os testes com o gravador desenvolvido como mostra a Figura 4.5. Nela é explicitado o que já foi dito anteriormente em relação ao MCS-Recorder estar num ambiente totalmente isolado tanto da máquina que rodará os *bots* quanto do servidor MCS. Sendo mais específico, em uma máquina virtual estará rodando o GE (servidor do MCS do GT-MCU), em outra máquina estarão sendo executados os *scripts* para criação/execução dos *bots* e, finalmente, no ambiente 1 (máquina local) estará sendo executado o gravador a ser testado. Como dito, a Figura 4.5 ilustra essa dinâmica.

Figura 4.5 - Arquitetura simplificada do Cenário 3



Fonte: O Autor

Tabela 4.9 - Resultados obtidos para a máquina local no Cenário 3

MÁQUINA LOCAL						
Nº GRAVAÇÕES	**2**	**4**	**8**	**16**	**32**	**64**
% CPU	17,00%	28,00%	43,53%	52%	-	-
TAM. ARQUIVO	5,1 MB	5,7 MB	5,7 MB	5,7 MB	-	-

Fonte: O Autor

Tabela 4.10 - Resultados obtidos para o NUC no Cenário 3

NUC						
Nº GRAVAÇÕES	**2**	**4**	**8**	**16**	**32**	**64**
% CPU	26,70%	59,86%	83,40%	-	-	-
TAM. ARQUIVO	5,1 MB	5,1 MB	5,7 MB	-	-	-

Fonte: O Autor

Tabela 4.11 - Resultados obtidos para a Jetson TX2 no Cenário 3

JETSON TX2						
Nº GRAVAÇÕES	**2**	**4**	**8**	**16**	**32**	**64**
% CPU	54,60%	98,63%	100%	-	-	-
TAM. ARQUIVO	5,1 MB	5,1 MB	5,7 MB	-	-	-

Fonte: O Autor

Tabela 4.12 – Comparação entre os resultados obtidos em cada ambiente

CENÁRIO 3 – MCS-RECORDER COM TRANSCODIFICAÇÃO						
Nº GRAVAÇÕES	**2**	**4**	**8**	**16**	**32**	**64**
% CPU	A1	A1	A1	A1	-	-
TAM. ARQUIVO	TODOS	TODOS	TODOS	A1	-	-

Fonte: O Autor

Diferente dos outros dois cenários com GStreamer, no Cenário 3, leva-se em conta também a capacidade do servidor MCS suportar uma alta quantidade de gravações de salas simultâneas, iniciadas, praticamente, ao mesmo tempo. Esse foi um fator que afetou os testes do *Recorder*.

Como visto na Tabela 4.9, apesar da máquina local suportar com tranquilidade as gravações com até 8 salas simultâneas, do caso 16 em diante, o servidor MCS apresentou instabilidades que geraram problemas como: queda do servidor, usuários *bots* sendo removidos e gravações não sendo iniciadas devido à falta de comunicação com o KMS. Como dito anteriormente, esse “gargalo” foi gerado pelo próprio servidor MCS. O que prejudicou os testes do sistema de gravação que, reforçando, respondia bem aos testes até o servidor do MCU parar de funcionar.

A Tabela 4.10 e a Tabela 4.11 mostram que os ambientes 2 e 3 (NUC e Jetson, respectivamente) também apresentaram problemas com a ferramenta. No caso desses dois ambientes, a transcodificação gerada pelo FFmpeg foi maior do que se comparada com a transcodificação feita pelo GStreamer levando-se em conta o consumo de CPU. Por isso, foi possível apenas a realização de oito salas simultâneas em ambos ambientes causado por um uso excessivo de CPU. Lembrando que, no caso do MCS-Recorder, ele utiliza somente recursos de CPU para a execução de suas aplicações.

Por outro lado, nos casos que tiveram sucesso nos três ambientes, os arquivos gerados possuem um tamanho inferior se comparados aos do Cenário 1 (GStreamer com

transcodificação). Os arquivos com GStreamer possuíam em torno de 13 MB (ou 25 MB no caso da Jetson TX2) com transcodificação e aproximadamente 22MB sem transcodificação. Enquanto os arquivos gerados neste terceiro cenário apresentam pouco mais de 5 MB sem perda na qualidade do vídeo.

4.5.4 Cenário 4 – Recorder SEM transcodificação

O quarto, e último cenário, ilustra os testes com o gravador desenvolvido assim como o do Cenário 3 (subseção 4.5.3). A principal diferença está na ausência de transcodificação graças à adição dos dois argumentos destacados na linha 10 da Figura 4.5, o *"-vcodec" e o "copy"*. São eles que garantem a ausência de transcodificação no FFmpeg.

Figura 4.6 – Trecho de código mostrando o comando de gravação no cenário 4

```
1   recordStream(localSdp) {
2
3     //...
4
5       let destination = Config.getConfig('download.destination');
6       let now = new Date();
7       let recordArgs = [
8         '-protocol_whitelist', 'file,rtp,udp',
9         '-i',path,
10        '-vcodec', 'copy',
11        destination + 'Gravacao_' +
12        this._roomId + '_' +
13        now.getDate().toString() + '-' +
14        (now.getMonth() + 1).toString() + '-' +
15        now.getFullYear() + '_' +
16        now.getHours() + '-' +
17        now.getMinutes() + '-' +
18        now.getSeconds() +
19        '.mkv', '-y',
20        '-loglevel','quiet'
21      ];
22      this.startRecording(this._ffmpegPath, recordArgs);
23    });
24  }
25
26  startRecording(appPath, recordArgs) {
27    this._process = childProcess.spawn(appPath, recordArgs);
28  }
```

Fonte: O Autor

Tabela 4.13 - Resultados obtidos para a máquina local no Cenário 4

MÁQUINA LOCAL						
Nº GRAVAÇÕES	**2**	**4**	**8**	**16**	**32**	**64**
% CPU	1%	1,5%	2,7%	4,97%	-	-
TAM. ARQUIVO	7,5 MB	7,5 MB	7,4 MB	7,4 MB	-	-

Fonte: O Autor

Tabela 4.14 - Resultados obtidos para o NUC no Cenário 4

NUC						
Nº GRAVAÇÕES	**2**	**4**	**8**	**16**	**32**	**64**
% CPU	1,3%	2,62%	6,35%	16,41%	-	-
TAM. ARQUIVO	7,5 MB	7,5 MB	7,4 MB	7,4 MB	-	-

Fonte: O Autor

Tabela 4.15 - Resultados obtidos para a Jetson TX2 no Cenário 4

JETSON TX2						
Nº GRAVAÇÕES	**2**	**4**	**8**	**16**	**32**	**64**
% CPU	2,47%	4,8%	8,42%	13%	-	-
TAM. ARQUIVO	7,5 MB	7,5 MB	7,4 MB	7,4 MB	-	-

Fonte: O Autor

Tabela 4.16 – Comparação entre os resultados obtidos em cada ambiente

CENÁRIO 4 – MCS-RECORDER SEM TRANSCODIFICAÇÃO						
Nº GRAVAÇÕES	**2**	**4**	**8**	**16**	**32**	**64**
% CPU	TODOS	TODOS	TODOS	TODOS	-	-
TAM. ARQUIVO	TODOS	TODOS	TODOS	TODOS	-	-

Fonte: O Autor

O Cenário 4, assim como no anterior, apresentou um problema de “gargalo” no servidor MCS. Como os arquivos usados para gerar os *bots* eram muitos grandes, ao passar do número de 16 gravações simultâneas, o MCS alcançava o máximo de uso de CPU e elevados índices, também, de uso de memória até chegar no momento que o servidor caia e mais nenhuma gravação era criada.

Explicado isso, analisando os resultados que foram alcançados na Tabela 4.13, na Tabela 4.14 e na Tabela 4.15 torna-se evidente que o uso (ou nesse caso, o não uso) de transcodificação foi novamente determinante para alcançar baixos percentuais de CPU nos três ambientes. Em relação ao tamanho dos arquivos, o Cenário 4 também pode ser considerado satisfatório já que mesmo sendo um pouco maior que o arquivo transcodificado (subseção 4.5.3), possuindo em torno de 7,5 MB, que ainda é bem menor que os vídeos obtidos nos cenários com GStreamer.

5 CONCLUSÕES

Gravação em videoconferência está longe de ser um assunto que gere discussão ou polêmica. Pelo contrário, ao perguntar para qualquer pessoa, seja ela especialista na área ou não, "*quais os principais pontos que tornam uma gravação de vídeo boa e confiável?*", obteremos as mesmas respostas quase sempre: qualidade de vídeo e acesso rápido.

Com os resultados alcançados foi possível enxergar um gargalo que, em algum momento, o servidor MCS pode causar no Gerenciador de Gravação apresentado até aqui. Mesmo assim, levando-se em conta situações e usos reais do gravador foi possível provar que o sistema funciona com alta qualidade de vídeo, baixo consumo de CPU e gera arquivos com tamanhos aceitáveis para o padrão de mercado. Essas características foram discutidas no capítulo de avaliação. Uma solução para esse problema seria o desacoplamento total de servidor de gravação do GT-MCU (o que pode ser um trabalho futuro).

A relação entre a solução proposta (o sistema de gravação) e o GStreamer (*framework* muito utilizado para a gravação de vídeo) é que o sistema desenvolvido foi capaz de equiparar todas as funcionalidades do GStreamer no quesito gravação de vídeo e, ainda, apresentar inovações como a questão do tempo de pós-processamento nulo e a gravação de múltiplas salas. Isso tudo faz com que o MCS-Recorder possa ser considerado um gravador apto para concorrer com outros gravadores do mercado.

Um ponto controverso de toda essa discussão é a transcodificação. Ao passo que seu uso é importante, em alguns casos, parar a formatação do vídeo, reduzindo o tamanho final dos arquivos, inclusive. Por outro lado, faz com que haja um consumo muito elevado de CPU, que dependendo do *hardware* (ou do cenário da conferência), irá prejudicar a qualidade da gravação, ou até inviabilizá-la. Dessa forma, o ideal é que o usuário determine o que é melhor para ele em cada cenário. Possuindo a alternativa de usar, ou não, a transcodificação.

Visando melhorias no sistema, trabalhos futuros podem abordar a questão de tornar a transcodificação juntamente com o uso de GStreamer opções configuráveis, aumentando, ainda mais, o poder de adaptação do gravador e permitindo ao portador dele definir qual o melhor uso em cada qualquer situação, avaliando seu custo-benefício de uma maneira mais certeira. Além disso, uma outra adição interessante seria a disponibilização das gravações pelo Portal e em nuvem. Essa possibilidade, talvez, possa ser uma solução para um possível problema de pouco espaço em disco. O que é uma limitação de *hardwares.*

REFERÊNCIAS

BUSINESS TECH. 2018. **Plantronics compra Polycom: Um negócio de US$ 2 Bilhões.** Disponível em: < https://www.businesstech.net.br/site/2018/03/plantronics-compra-polycom-um-negocio-de-us-2-bilhoes/ > - acesso em dezembro de 2019.

CAMARILLO, G.; OTT, J.; DRAGE, K. **The Binary Floor Control Protocol (BFCP).** [S.l.]: RFC Editor, 2006. RFC 4582. (Request for Comments, 4582).

DAENGSI, T.; WUTIWIWATCHAI, C.; PREECHAYASOMBOON, A.; SUKPARUNGSEE, S. (2012). **Speech Quality Assessment of VoIP: G.711 VS G.722 Based on Interview Tests with Thai Users.** International Journal of Information Technology and Computer Science. 4. 19-25. 10.5815/ijitcs.2012.02.03.

DATASHEET TEGRA. 2014. **DATA SHEET - NVIDIA Jetson TX2 System-on-Module.pdf.** Disponível em: <https://www.docdroid.net/yGXIxZu/data-sheet-nvidia-jetson-tx2-system-on-module.pdf#page=2 > - acesso em dezembro de 2019.

GROZEV, B. et al. **Last n: Relevance-based selectivity for forwarding video in multimedia conferences.** In: Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video. New York, NY, USA: ACM, 2015. (NOSSDAV '15), p. 19–24. ISBN 978-1-4503-3352-8.

HANDLEY, M.; JACOBSON, V.; PERKINS, C. **SDP: Session Description Protocol.** [S.l.]: RFC Editor, 2006. RFC 4566. (Request for Comments, 4566).

KURENTO. 2018. Disponível em: <https://doc-kurento.readthedocs.io/en/stable/tutorials/js/tutorial-recorder.html> - acesso em outubro de 2019.

KUROSE, J.; ROSS, K. **Redes de Computadores e a Internet: Uma Abordagem Top-Down.** 6. Ed. Pearson. 2013.

LÓPEZ-FERNÁNDEZ, L., GARCÍA, B., GALLEGO, M. et al. **Multimed Tools Appl** (2017) 76: 14247. https://doi.org/10.1007/s11042-016-3729-z.

OZER, J. **"Encoding for Multiple Screen Delivery, Section 3, Lecture 7: Introduction to H.264".** Udemy. Disponível em: <https://www.udemy.com/course/encoding-for-multiple-screen-delivery/> - acesso em setembro de 2019.

PORTAL TIBAHIA. **Polycom: gravação de vídeo em dispositivos móveis** (2012). Disponível em: <http://tibahia.com/tecnologia_informacao/conteudo_unico.aspx?c=NTO_FABR&fb=B_FULL&hb=B_CENTRA&bl=LAT1&r=NTO_FABR&nid=16883>. - acesso em setembro de 2019.

ROESLER, V. et al. **Proposta de uma estratégia econômica de videocolaboração em tempo real para as NRENs.** “Transformación Digital en Instituciones de Educación Superior, Ciencia y Cultura”. Octava Conferencia de Directores de Tecnología de Información y Comunicación en Instituciones de Educación Superior, TICAL2018 y II Encuentro Latinoamericano de e-Ciencia.

ROESLER, V. et al. **Desenvolvimento de um Serviço de Videoconferência com MCU (Multipoint Control Unit) na Nuvem.** “Transformación Digital en Instituciones de Educación Superior, Ciencia y Cultura”. Octava Conferencia de Directores de Tecnología de Información y Comunicación en Instituciones de Educación Superior, TICAL2018 y II Encuentro Latinoamericano de e-Ciencia.

ROSENBERG, J. et al. **SIP: Session Initiation Protocol.** [S.l.]: RFC Editor, 2002. RFC 3261. (Request for Comments, 3261).

RSS 4000. Datasheet do RSS 4000, 2009. Disponível em: <https://www.vtkt.ru/upload/iblock/1c8/Polycom_rss-4000.pdf> - acesso em outubro de 2019.

SCHULZRINNE, H. et al. **RTP: A Transport Protocol for Real-Time Applications.** [S.l.]: RFC Editor, 2003. RFC 3550. (Request for Comments, 3550).

STATION PRO. Datasheet do Station Pro, 2014. Disponível em: <https://www.databox-int.com/wp-content/uploads/realpresence-capture-station-pro-ds-enus.pdf> - acesso em outubro de 2019.

VOIT. 2019. Disponível em: <https://www.voit.com.br/poly-empresa-resultado-da-uniao-da-plantronics-polycom-com-foco-na-experiencia-de-comunicacao-e-colaboracao/ > - acesso em dezembro de 2019.

ANEXO A – TRABALHO DE GRADUAÇÃO – I

- Título: Gravação de vídeo em sistemas de videoconferência
- Data: outubro de 2019

Gravação de vídeo em sistemas de videoconferência

Oberdan Costa dos Santos[1], Valter Roesler[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{ocsantos, roesler}@inf.ufrgs.br

Abstract. *Video conferencing systems save time and resources by avoiding travel and travel expenses. Making the recording of these meetings available allows asynchronous access to those who cannot attend or want to review the video. The purpose of this paper is to analyze the operation of video recording in videoconferencing systems. Some existing recording formats will be compared, and a recording system implemented. Its performance will be analyzed on different hardware to better validate the study.*

Resumo. *Sistemas de videoconferência representam economia de tempo e recursos uma vez que evitam deslocamentos e gastos com viagens. A disponibilização da gravação dessas reuniões permite o acesso assíncrono a quem não pode participar ou quer rever o vídeo. A proposta deste trabalho é analisar o funcionamento da gravação de vídeo em sistemas de videoconferência. Serão comparados alguns formatos de gravação existentes, e será efetuada a implementação de um sistema de gravação. Seu desempenho será analisado em diferentes hardwares para melhor validar o estudo.*

1. Introdução

Videoconferências por hardware ou via web são cada mais frequentes no dia-a-dia. Seja pela sua praticidade e comodidade ou pelo simples fato de que as pessoas hoje não querem perder tempo e dinheiro se deslocando de um lugar a outro apenas para uma reunião de algumas horas. Esse *tradeoff* deixou de ser vantajoso, se é que um dia foi, no momento em que perde-se mais tempo e disposição no percurso para a reunião do que nela propriamente.

Um das soluções encontradas para o problema da falta de disponibilidade para reuniões, sejam elas presenciais ou até mesmo *online,* (afinal, pode ser que naquele momento do dia a pessoa possa não estar apta a participar seja por um imprevisto ou qualquer outra razão) foi a gravação das videoconferências e a disponibilização do seu conteúdo ao final delas. Entretanto, essa nova função tem seus prós e contras, ao passo que facilita muito a vida de todos ter a gravação quando quiser, por outro lado um sistema ou aparelho de gravação pode, e normalmente, custa muito caro. Além de, em alguns casos, necessitar de um tempo de pós-processamento do arquivo ao final, atrasado sua liberação, o que pode gerar transtornos. E, ainda, os clássicos problemas de falhas/perdas na qualidade durante a gravação.

Pensando nisso, em parceria com o GT-MCU (atual GT-Videocolaboração) da UFRGS sobre o comando do professor Valter Roesler, foi desenvolvido um sistema de gravação em tempo real de videoconferências em alta resolução, com bom desempenho e tempo ínfimo (quase nulo) de pós-processamento. Neste artigo, serão apresentados conceitos importantes sobre o tema, uma análise da realidade de gravações atualmente e o planejamento do desenvolvimento do sistema proposto. O desenvolvimento juntamente com sua validação através de testes e simulações serão detalhados na segunda etapa do trabalho de graduação.

2. Conceitos importantes envolvendo videoconferência e gravação

Alguns termos vão ser recorrentes neste texto, por conta disso, nesta seção, os mesmos serão mais detalhados. Tratam-se dos modelos de distribuição de mídia descritos pela literatura,

dos protocolos de transporte, sinalização, negociação e compartilhamento de mídia, dos codecs e containers multimídia utilizados neste trabalho, além dos frameworks de mídia e tecnologias de código aberto que auxiliaram na realização desse projeto.

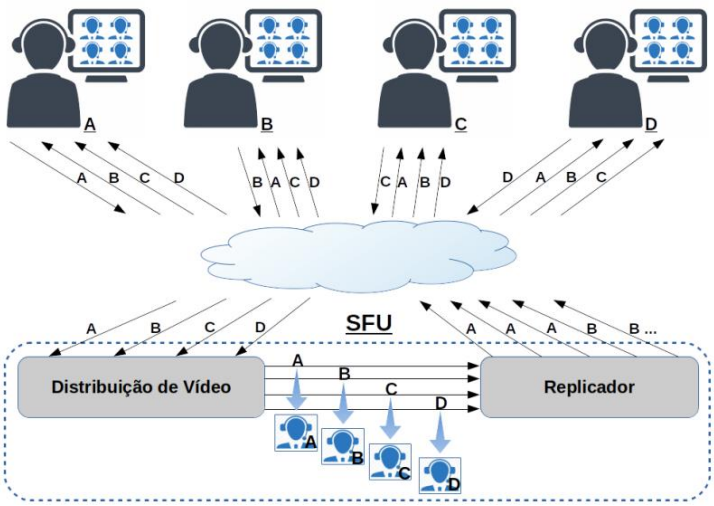
2.1. Modelos de distribuição de mídia

O conceito principal dos sistemas de videoconferência é permitir a troca de fluxo de mídia entre os pares conectados. Segundo GROZEV et al. (2015), topologias comuns para a distribuição de mídia em ambientes de videoconferência são a SFU (*Selective Forwarding Unit*), MCU (*Multipoint Control Unit*) e MESH (ou *Peer-to-Peer*).

Por este trabalho ter sido feito baseado apenas nos modelos SFU e MCU, o modelo MESH não será abordado.

2.1.1. Selective Forwarding Unit

Figura 1 - Dinâmica de funcionamento SFU



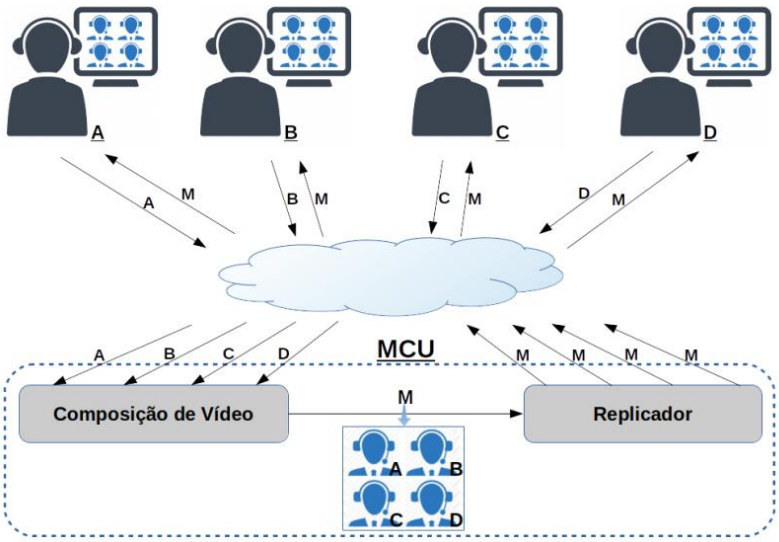
Fonte: (ROESLER et al., 2018)

A Figura 1 apresenta a dinâmica de uma unidade de encaminhamento seletivo de mídia (SFU) que, por característica, não possui nenhum elemento central. Ela apenas faz a distribuição (ou encaminhamento) dos fluxos de mídia até os endpoints sem nenhum tipo de processamento adicional (codificação ou decodificação) o que, geralmente, gera mais eficiência de CPU. Considerando a eficiência de largura de banda, uma SFU não encaminha todos os pacotes, na verdade, ela usa um algoritmo de seleção que decide qual pacote encaminhar para qual *endpoint* (GROZEV et al., 2015).

Resumidamente, usuários enviam seus fluxos de mídia para o SFU que replica esses fluxos e envia para cada *endpoint* uma cópia de cada fluxo de cada um dos outros usuários.

2.1.2. Multipoint Control Unit

Figura 2 - Dinâmica de funcionamento MCU



Fonte: (ROESLER et al., 2018)

A Figura 2 detalha a ação de um MCU (*Multipoint Control Unit*) que é um equipamento ou serviço usado em videoconferências que permite comunicação multiponto. Ele funciona como um elemento central que recebe os fluxos de mídia de cada um dos usuários conectados, passa por um processo de composição de vídeo, gerando um único fluxo. Esse novo fluxo é replicado e enviado para todos os equipamentos conectados (*endpoints*), possibilitando aos usuários visualizarem tal composição.

2.2. Protocolos

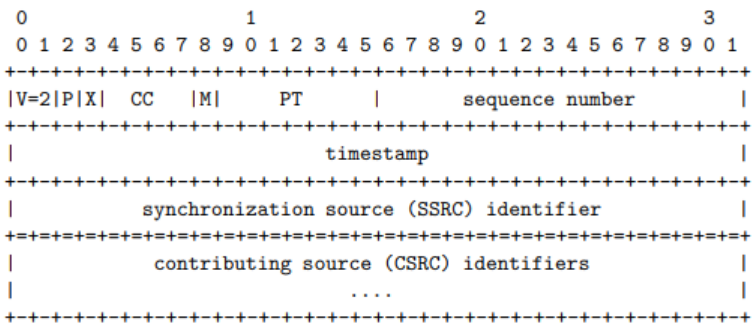
2.2.1. Transporte e controle de mídia

2.2.1.1. Real-time Transport Protocol

O protocolo de transporte em tempo real, RTP, determina um formato de pacote padrão para o envio de áudio e vídeo pela Internet. Foi definido inicialmente pela RFC 1889 e depois pela RFC 3550. O protocolo fornece serviços de entrega fim-a-fim como identificação do tipo de carga, numeração de sequência, registro de data e hora e monitoramento de entrega para dados em tempo real como, por exemplo, áudio e vídeo em videoconferências. Normalmente, as aplicações executam o RTP sobre UDP. Além disso, o RTP permite a transferência de dados usando distribuição *multicast*, se necessário.

Se tratando de um cenário de videoconferências, as mídias de áudio e vídeo usadas serão transmitidas em sessões RTP separadas, ou seja, usando dois pares de portas UDP diferentes e/ou endereços *multicast*. Essa opção existe para ceder ao participante a capacidade de receber ou compartilhar apenas um meio conforme preferir (SCHULZRINNE, 2003).

Figura 3 - Formato do cabeçalho RTP



Fonte: (SCHULZRINNE et al., 2003)

Na Figura 3, pode-se ver o formato de um cabeçalho RTP. Os doze primeiros bytes estão presentes em todos os pacotes RTP, enquanto a lista dos identificadores do CSRC só está quando inserida por um mixer. Os elementos em questão são: versão, *padding*, extensão, contador CSRC, marcador, tipo do *payload*, número de sequência, *timestamp*, SSRC e a lista CSRC.

2.2.1.2. Real-time Transport Control Protocol

O RTCP (ou protocolo de controle RTP), assim como o RTP, foi definido inicialmente pela RFC 1889 e depois pela RFC 3550. O protocolo atua juntamente com RTP e tem como funções permitir o monitoramento da entrega de dados de maneira escalável para grandes redes *multicast* e para fornecer funcionalidade mínima de controle e identificação, ou seja, monitorando a qualidade do serviço e transmitindo informações sobre os participantes de uma sessão. Além de suportar o uso de mixadores e tradutores RTP.

Voltando ao cenário de videoconferências, como não existe acoplamento direto (em nível RTP) entre sessões de áudio e vídeo, o usuário deve possuir o mesmo nome canônico nos pacotes RTCP (áudio e vídeo) para que as sessões possam ser associadas (SCHULZRINNE, 2003).

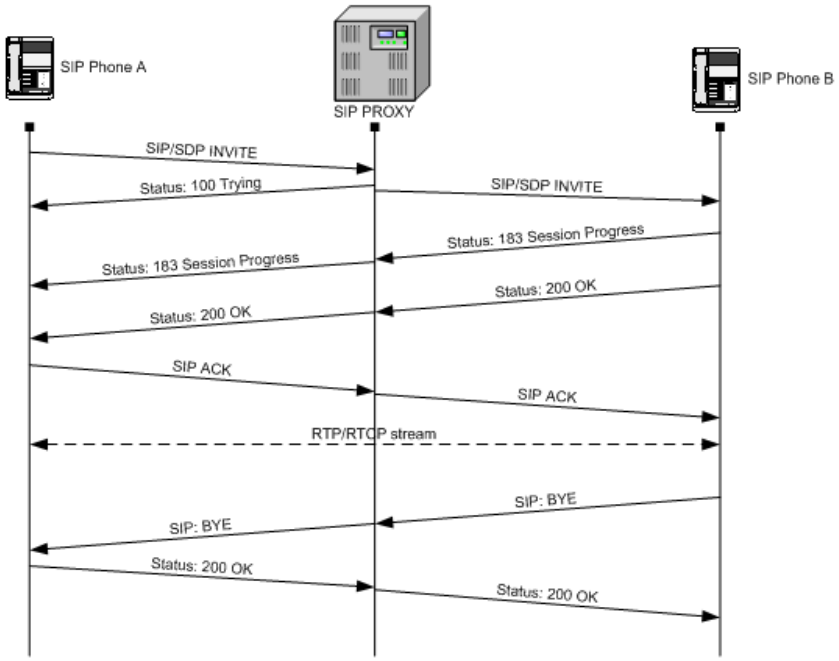
2.2.2. Sinalização e negociação de mídia

2.2.2.1. Session Initiation Protocol

O SIP (ou Session Initiation Protocol) é um protocolo de sinalização da camada de aplicação que pode estabelecer, modificar e encerrar sessões multimídias (videoconferências, ensino a distância, etc) ou chamadas (ROSENBERG et al., 2002). SIP é o padrão para comunicação de voz e vídeo em tempo real. Ele foi criado pela IETF e publicado como RFC 3261. Além disso, o SIP funciona tanto em sessões unicast como nas *multicast* podendo ser usado tanto com IPv4 como com IPv6. O desenvolvimento do SIP se deu visando o suporte a pontos importantes do estabelecimento de sessões multimídia na rede, como, localização, disponibilidade e capacidades do usuário, estabelecimento, configuração e gerenciamento de sessões (ROSENBERG et al., 2002). Em linhas gerais, podemos fazer a seguinte associação: "O SIP está para as comunicações em tempo real assim como o HTTP está para a rede e o SMTP está para os E-mails"[1].

[1]https://www.3cx.com.br/voip-sip/sip-faq/ - acesso em setembro 2019

Figura 4 - Exemplo de comunicação SIP



Fonte: (MASTALIR, 2005)

Na Figura 4, pode-se observar uma troca de mensagens SIP, desde o seu início até a sua conclusão. Como pode ser visto, uma chamada de um usuário SIP A para um usuário SIP B passa pelas seguintes etapas: estabelecimento do contato (SIP/SDP *INVITE*, OK e SIP ACK), troca de dados (RTP/RTCP *stream*) e finalmente a finalização da chamada (SIP *BYE* e OK).

2.2.2.2. Session Description Protocol

O protocolo de descrição de sessão, SDP, é um conjunto de regras que define como sessões multimídia serão configuradas para permitir aos endpoints participar da sessão.[2] Foi definido pela RFC 2327 em 1998 e mais tarde atualizado pelas RFC 3266 (2002) e finalmente pela RFC 4566 em 2006.

Em geral, o SDP deve cumprir dois objetivos principais: transmitir informações suficientes para permitir que aplicações ingressem em uma sessão e anunciar os recursos que um provável futuro participante possa precisar saber (HANDLEY; JACOBSON; PERKINS, 2006).

Figura 5 - Trecho de arquivo SDP com descrição da sessão

```
v=0
o=- 3778853736 3778853736 IN IP4 143.54.10.106
s=Kurento Media Server
c=IN IP4 143.54.10.106
t=0 0
a=sendrecv
a=vnd.polycom.MBA.p2p:v=1.0.1
```

Fonte: O autor

[2]https://searchunifiedcommunications.techtarget.com/definition/SDP - acesso em setembro 2019

Figura 6 - Trecho de arquivo SDP com descrição da mídia de áudio

```
m=audio 16426 RTP/AVP 9 119
a=rtpmap:9 G722/8000
a=rtpmap:119 telephone-event/8000
a=fmtp:119 0-16
a=rtcp:16427 IN IP4 143.54.10.32
a=ptime:20
```

Fonte: O autor

Figura 7 - Trecho de arquivo SDP com descrição da mídia de vídeo

```
m=video 25714 RTP/AVP 111 109 110
b=TIAS:300000
b=AS:300
a=rtpmap:111 H264/90000
a=rtpmap:109 H264/90000
a=rtpmap:110 H264/90000
a=fmtp:111 profile-level-id=42e01f; packetization-mode=0; level-asymmetry-allowed=0
a=fmtp:109 profile-level-id=42e01f; packetization-mode=0; level-asymmetry-allowed=0
a=fmtp:110 profile-level-id=42e01f; packetization-mode=0; level-asymmetry-allowed=0
a=rtcp-fb:* nack pli
a=rtcp-fb:* ccm fir
a=rtcp-fb:* ccm tmmbr
```

Fonte: O autor

Figura 8 - Trecho de arquivo SDP com descrição da mídia BFCP

```
m=application 50000 UDP/BFCP *
c=IN IP4 143.54.10.32
a=sendrecv
a=floorctrl:s-only
a=setup:passive
a=confid:1
a=userid:100
a=floorid:1 m-stream:3
a=connection:new
```

Fonte: O autor

Nas figuras acima, é apresentado um exemplo de arquivo SDP fragmentado pela descrição de sessão e de tipos de mídia. Na Figura 5, constam descritas informações referentes a versão do protocolo (v), criador da sessão (o), nome da sessão (s), tempo de atividade da sessão (t), informações de conexão (c) e atributos de sessão (a). Esses dois últimos sendo opcionais. Nas Figuras 6, 7 e 8, respectivamente, estão representadas as mídias de áudio, vídeo e conteúdo. Nelas constam o nome da mídia e endereço de transporte (m), além das informações opcionais: atributos da mídia (a), largura de banda (b) e conexão (c). Essa última permite que uma mídia se conecte a um endereço IP diferente da sessão se for necessário.

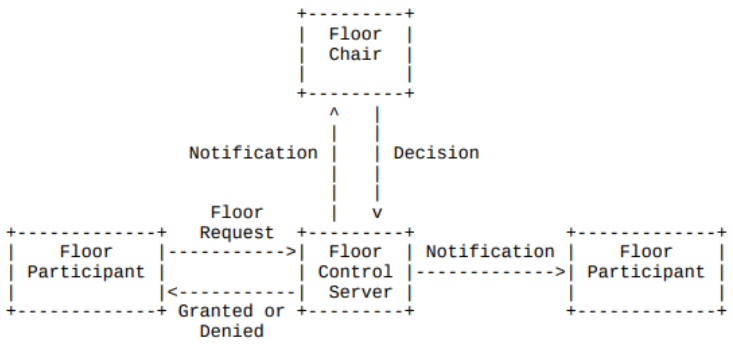
2.2.3. Protocolo de compartilhamento de conteúdo

2.2.3.1. Binary Floor Control Protocol

O BFCP foi criado pela IETF e definido na RFC 4582 em 2006. O controle do *floor* é a permissão temporária para gerenciar o acesso ou manipular um determinado recurso compartilhado ou conjunto de recursos (CAMARILLO; OTT; DRAGE, 2006). Ele complementa funções como configurar conferência e sessão de mídia, manipular política de conferência e controle de mídia (realizados por outros protocolos).

No contexto de videoconferências, o BFCP é usado entre participantes, moderadores e servidores de controle. Na Figura 9, retirada da RFC 4582, são exibidas tarefas que o BFCP pode executar.

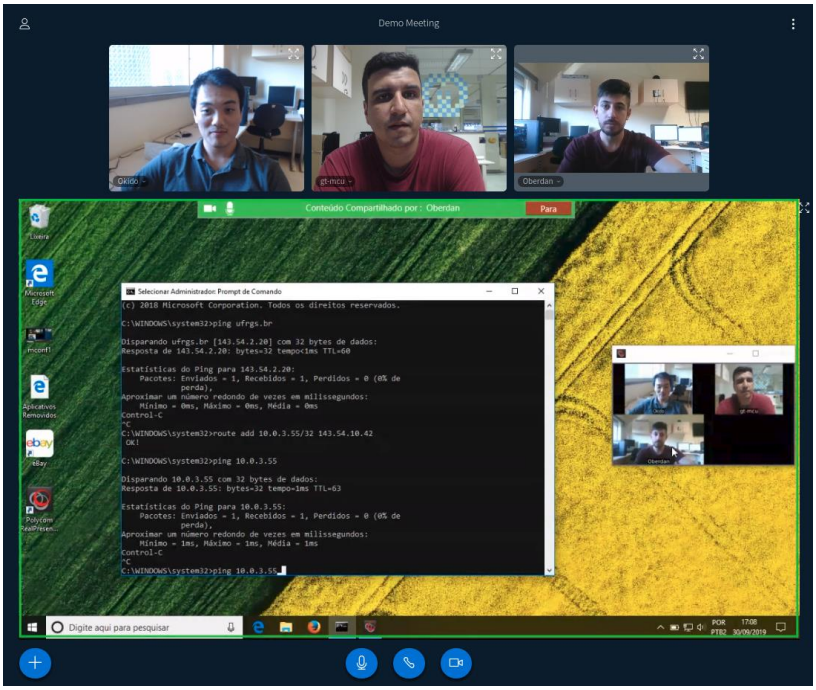
Figura 9 - Tarefas suportadas pelo BFCP



Fonte: (CAMARILLO; OTT; DRAGE, 2006)

Na Figura 10, entende-se bem o uso do protocolo que permite enviar e receber conteúdo em uma tela adicional da videoconferência. Dessa maneira, teremos uma tela com os participantes da sessão e em outra o compartilhamento de tela do *floor*. O contexto da imagem mostra ambiente *web*, com usuários SIP, através do software *RealPresence Desktop*[3], sendo um deles quem está compartilhando o conteúdo.

Figura 10 - Chamada com compartilhamento de conteúdo



Fonte: O autor

[3]https://support.polycom.com/content/support/north-america/usa/en/support/video/realpresence-desktop/realpresence-desktop.html - acesso em setembro 2019

2.3. Codecs

Codec vem de codificação e decodificação. Dessa forma, pode-se dizer que um codec comprime um arquivo originalmente muito grande em um conteúdo bem menos pesado para que possa ser compartilhado na rede. Após isso ele é "reconstruído" para ser executado.[4]

O que diferencia os codecs é a forma de codificação ou decodificação. Por exemplo, uns comprimem ignorando totalmente possíveis perdas de qualidade; alguns comprimem primando evitar baixas na qualidade da transmissão e outros comprimem ao máximo em momentos que a perda da qualidade é aceitável.

No contexto videoconferências, tanto para áudio quanto vídeo, existe a necessidade de que o par que esteja se comunicando possua ao menos um codec em comum, não importando o modelo de distribuição usado (SFU, MCU, etc). Existem muitos codecs conhecidos, tais como, H.264, VP8, MPEG-2, H.263, WMV, WAV, AC3, G.722, OPUS, etc. Para este estudo, serão abordados o H.264 e G.722. Esses dois foram os codecs usados como padrão pelo gravador desenvolvido.

2.3.1. Codec de vídeo

2.3.1.1. H.264

O padrão H.264, ou MPEG-4 Part 10, Advanced Video Coding (MPEG-4 AVC), é um padrão de compactação de vídeo baseado em compensação de movimento orientado a blocos. Desde 2014, é um dos formatos mais usados para a gravação, compactação e distribuição de conteúdo de vídeo (OZER, 2016). O nome H.264 segue a convenção de nomenclatura ITU-T, onde o padrão é um membro da linha H.26x dos padrões de codificação de vídeo VCEG; o nome MPEG-4 AVC refere-se à convenção de nomenclatura na ISO / IEC MPEG, onde o padrão faz parte 10 da ISO / IEC 14496, que é o conjunto de padrões conhecido como MPEG-4.

A motivação foi criar um padrão capaz de fornecer boa qualidade de vídeo a taxas de bits mais baixas do que os padrões anteriores MPEG-2, H.263, etc), sem aumentar a complexidade do design a ponto de torná-lo impraticável ou muito custoso de implementar. O H.264 é normalmente usado para compactação com perdas, podendo também serem criadas regiões realmente sem código em imagens com código ou com suporte a casos de uso raros nos quais toda a codificação é sem perdas.

O H.264 foi padronizado pelo ITCE-T *Video Coding Experts Group* (VCEG) em conjunto com o ISO / IEC JTC1 *Moving Picture Experts Group* (MPEG). A primeira versão do padrão foi redigida em maio de 2003, e várias extensões foram adicionadas nas edições subsequentes. A MPEG LA e outros proprietários têm direito a royalties pelo uso comercial das tecnologias H.264. Foi o uso gratuito dessas tecnologias para transmissão de vídeos da Internet gratuitos para os usuários finais, e a *Cisco Systems* paga *royalties* à MPEG LA em nome dos usuários dos binários pelo seu codificador H.264 de código aberto.[5]

---

[4] https://canaltech.com.br/software/o-que-sao-codecs-quais-sao-os-tipos-pra-que-servem-saiba-mais-sobre-esse-tema/ - acesso em setembro 2019

[5] https://www.mpegla.com/programs/avc-h-264/faq/ - acesso em setembro 2019

2.3.2. Codec de áudio

2.3.2.1. G.722

É um codec de áudio que segue o padrão da ITU-T, aprovado em 1988, que pode operar em 48, 56 ou 64 kbps. Ele coleta dados de áudio a uma taxa de 16 kHz; esse valor equivale a duas vezes a velocidade das interfaces de telefonia tradicionais.[6] Dessa maneira, o codec fornece maior clareza e qualidade de áudio.

Ele usa modulação de código de pulso diferencial adaptativo de sub-banda (SB-ADPCM), isso acaba exigindo uma taxa de bits de 64 kbps. Este foi o primeiro codec de banda larga emitido pelo ITU-T que suporta uma largura de banda de até 7 kHz. Também existem derivações como G.722.1 e G.722.2, porém não suportam taxa de bits 64 kbps (DAENGSI et al., 2012). Em videochamadas, o G.722 é transportado no payload do protocolo RTP (SCHULZRINNE; CASNER, 2003).

2.4. Contêineres multimídia

Muitos confundem a definição de contêiner com codec, mas na verdade um contêiner é um conjunto arquivos de áudio e vídeo que utilizam codecs. Entre os mais conhecidos estão o WebM, AVI, FLV, MP4 e MKV. Os últimos dois citados foram testados nesse estudo, sendo detalhados na sequência.

2.4.1. MP4

O formato MPEG-4 (ou MP4) faz parte do grupo MPEG (*motion Pictures Expert Group*). Ele é o formato de contêiner multimídia digital mais usado para armazenar vídeo e áudio, podendo ser usado para armazenar outros tipos de dados, como legendas e imagens estáticas. O MP4 pode ser usado tanto com o objetivo de criar vídeos para download, como para streaming. No caso deste estudo, ele foi um dos formatos escolhidos para este estudo.

2.4.2. MKV

O já citado MKV (*Matroska Video*) recebe esse nome em referências as bonecas russas que possuem outras dentro delas. Essa relação é feita por conta de os vídeos em MKV poderem conter imagem, áudio e legendas num só arquivo, em diferentes formatos. É muito usado também para compressão de vídeos de alta resolução (1080p). Por esse último motivo e, principalmente, por não perder o conteúdo em caso de falhas, foi escolhido neste estudo para substituir o MP4 para gravações que se fizesse necessário uma resolução FULL HD (1080p). Embora os arquivos finais ficassem maiores que os com formato MP4.

2.5. Frameworks de mídia

2.5.1. FFmpeg

O FFmpeg é a principal estrutura multimídia que possui projeto gratuito e de código aberto. Iniciado em 2000 por Fabrice Bellard (a.k.a. Gérard Lantau) e continuado por Michael Niedermayer de 2004 à 2015. Alta portabilidade, baixas dependências de outras bibliotecas e máximo compartilhamento de código são objetivos vendidos pelo projeto.[7]

---

[6] https://www.techopedia.com/definition/4465/g-722 - acesso em setembro 2019

[7] http://ffmpeg.org/about.html - acesso em setembro 2019

O principal programa é o homônimo do *framework*, FFmpeg. Ele processa com base em linhas de comando mídias de áudio e vídeo podendo, também, ser usado para transcodificação, dimensionamento, pós-processamento rápido e conformidade com padrões como o SMPTE e o ITU. Esses foram alguns dos motivos para a escolha de usar esse *framework* inicialmente. Outra razão foi por esse formato ser o mesmo usado no código *mcs-rtp-client* que gera o *bot* no qual o gravador deste estudo foi baseado.

2.5.2. GStreamer

GStreamer é um framework multiplataforma para a construção de aplicações de mídia. Foi desenvolvido em 1999 por Erik Walthinsen, entre outros. Mas o projeto expandiu após a entrada de Wim Taymans, atual líder do projeto. Foi escrito em C e baseado em GObject. Utiliza uma arquitetura de *plugins* e *bindings* que aumenta sua diversidade em bibliotecas de mídia e linguagens de programação. GStreamer é um software livre, licenciado sob GNU GPL. Um fato relevante e que torna o GStreamer ainda mais confiável é sua estabilidade desde 2012.[8]

Uma das ferramentas desse framework que será abordada neste estudo é o *gst-launch*, que nada mais é do que uma ferramenta de linha de comando para prototipagem e testes que foi usada para gerar o comando de gravação quando foi optado usar GStreamer no lugar do FFmpeg.[9]

Figura 11 - Pipeline usando ferramenta *gst-launch*

```
gst-launch-1.0 videotestsrc pattern=black ! gdkpixbufoverlay location=gtmcu.png
! video/x-raw, framerate=\(fraction\)1/1, width=1280, height=720 !
compositor name=comp sink_0::xpos=0 sink_0::ypos=0 sink_1::xpos=0 sink_1::ypos=0
sink_2::xpos=320 sink_2::ypos=180 ! video/x-raw, width=1280, height=720 !
videoconvert ! ximagesink v4l2src device=/dev/video0 !
video/x-raw, width=320, height=180 ! comp. videotestsrc !
video/x-raw, width=960, height=540 ! comp.
```

Fonte: O autor

Figura 12 - Captura gerada através do pipeline da Figura 11

Fonte: O autor

[8]https://en.wikipedia.org/wiki/GStreamer - acesso em setembro 2019

[9]https://gstreamer.freedesktop.org/features/index.html - acesso em setembro 2019

Nas Figuras 11 e 12, podemos ver a força do uso do GStreamer para aplicações multimídia. Em poucas linhas, foi gerado a captura do vídeo do usuário, um frame de vídeo simulando o compartilhamento de conteúdo e uma logomarca com uma imagem estática.

2.6. Outras tecnologias

2.6.1. *Plataforma de videoconferência MCU*

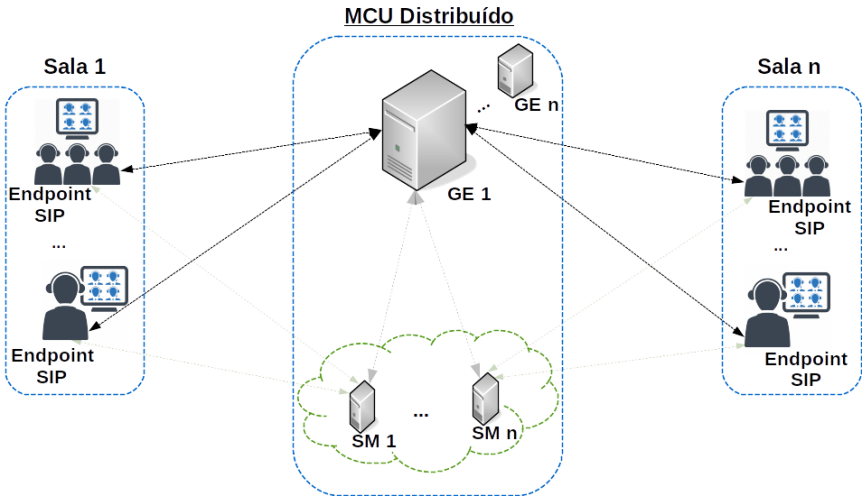
O GT-MCU (Grupo de Trabalho em MCU) é um projeto do grupo de pesquisa PRAV da UFRGS liderado pelo professor Valter Roesler. Foi desenvolvido em software livre, financiado pela RNP (Rede Nacional de Ensino e Pesquisa do Brasil) e pela empresa Mconf Tecnologia.

Esse projeto propõe um serviço de videoconferência de baixo custo totalmente virtualizado, funcionando na nuvem de forma escalável. A plataforma funciona praticamente como um framework universal de encaminhamento de mídias e não apenas como um MCU, o que acaba permitindo que outros serviços se integrem, como webconferência e VoIP.[10]

O autor deste estudo foi o responsável por desenvolver o sistema de gravação do GT-MCU de maneira quase que genérica a ponto de possibilitar este sistema de gravação possa, futuramente, ser usado em outras plataformas, com poucas adaptações.

As Figuras 13 e 14[10] apresentam a visão geral do GT-MCU e uma videoconferência do grupo, respectivamente.

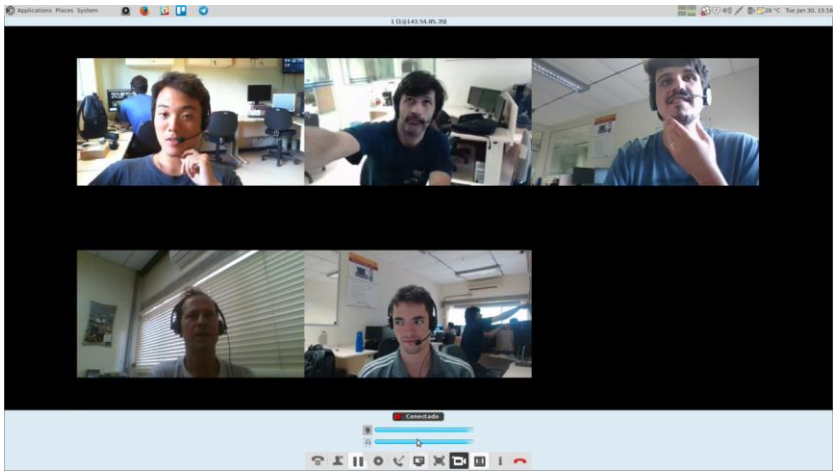
Figura 13 - Visão geral de um MCU distribuído



Fonte: (ROESLER et al., 2018)

[10] http://www.inf.ufrgs.br/prav/mcu.htm - acesso em setembro 2019

Figura 14 - Reunião da equipe do GT-MCU via videoconferência MCU



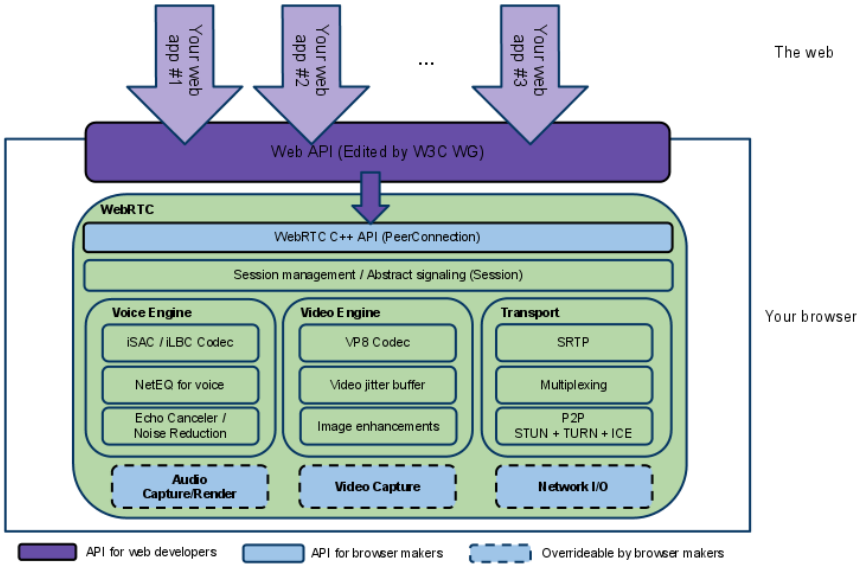
Fonte: (ROESLER et al., 2018)

2.6.2. Web Real-Time Communication

O WebRTC, escrito em C++ e *JavaScript*, é um projeto aberto e de software livre (sobre termos de licença BSD) que permite aos navegadores e aplicativos móveis comunicação em tempo real (RTC) por meio de APIs simples.[11] Foi padronizado pela W3C e pela IETF. A principal vantagem do WebRTC é a eliminação da necessidade de uso de *plugins* ou downloads de softwares nativos.

A Figura 15, retirada da página oficial[11] do WebRTC, apresenta a arquitetura por trás do projeto.

Figura 15 - Arquitetura do WebRTC



Fonte: Página oficial do WebRTC[11]

[11] https://webrtc.org/ - acesso em setembro 2019

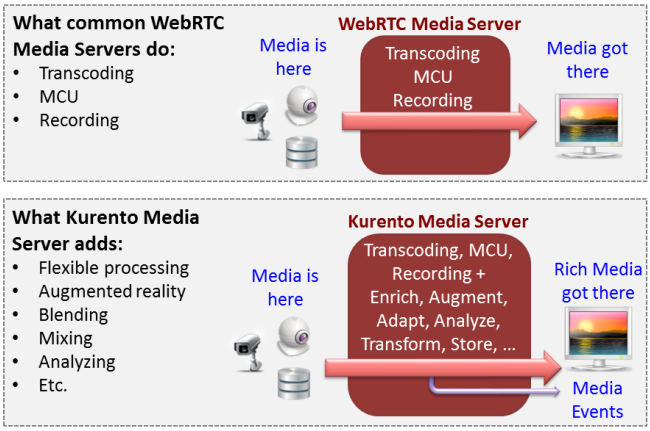
2.6.3. Servidores de mídia

2.6.3.1. Kurento Media Server

O Kurento é um WebRTC *Media Server* e um conjunto de APIs clientes que simplificam o desenvolvimento de aplicativos de vídeo avançados para plataformas da *web* e de *smartphones*.[12] Possui código aberto, lançado sob os termos da *Apache License Version* 2.0[13] e disponível no GitHub[14].

O Kurento Media Server (KMS) é o elemento principal do Kurento, por ser responsável pela transmissão, processamento, carregamento e gravação de mídia.[15] É desenvolvido em cima do GStreamer. Na Figura 16, retirada da página oficial do Kurento[12], pode ser visto as funcionalidades que o KMS suporta em relação a um servidor WebRTC comum.

Figura 16 - Comparação de funcionalidades entre um KMS e um servidor de mídia WebRTC



Fonte: Página oficial do Kurento[12]

3. Análise de realidade

Para embasar o desenvolvimento do gravador deste trabalho foram comparados outros sistemas (e produtos) responsáveis pela gravação de uma videoconferência. Serão apresentados as soluções da *Polycom* e o sistema de gravação do Kurento. A ideia principal dessa seção é apresentar o contexto de gravadores atualmente.

3.1.Polycom

A *Polycom* é a empresa líder global em soluções de comunicações unificadas baseadas em padrão aberto para telepresença, vídeo e voz, impulsionada pela plataforma *Polycom RealPresence* (PORTAL TIBAHIA, 2012). Essa plataforma transita suas operações em diversos dispositivos e aplicativos sociais, móveis e de negócios.

---

[12]https://doc-kurento.readthedocs.io/en/6.11.0/user/about.html - acesso em outubro 2019

[13]https://www.apache.org/licenses/LICENSE-2.0 - acesso e m outubro 2019

[14]https://github.com/Kurento - acesso em outubro 2019

[15]https://doc-kurento.readthedocs.io/en/6.11.0/glossary.html#term-kurento-media-server - acesso em outubro 2019

Algumas das soluções que possibilitam a gravação são o *Polycom RealPresence Video Content Management*, o *Polycom* RSS 4000 *Recording and Streaming Server*, o *RealPresence Capture Station Pro*, entre outras soluções que fazem da Polycom um referencial nessa área.

O primeiro, garante um acesso seguro a *streamings web* e gravações de vídeo sob demanda por meio de um portal na *web*. Já o RSS 4000, além de possuir uma interface baseada em *web*, permite o uso de *streamings* e gravações de videoconferências em alta definição (HD) (RSS 4000, 2009).

Figura 17 - Polycom RSS 4000 Recording and Streaming Server

Fonte: Loja online vuports[16]

Um exemplar desse *hardware* pode ser visto na Figura 17 que, depois de uma rápida pesquisa, foi encontrado por um valor de 22.500 dólares.[16] Enquanto isso, o *Station Pro* é dispositivo de fácil utilização que grava e faz *streaming* de apresentações de qualquer ambiente. Ele sincroniza áudio, vídeo e slides de apresentação em tempo real (STATION PRO, 2014). Um modelo desse *hardware* pode ser visto na Figura 18, sendo encontrado num valor aproximado de 9.000 dólares.[17]

Figura 18 - Polycom RealPresence® Capture Station Pro

Fonte: Loja online *ipphone-warehouse*[17]

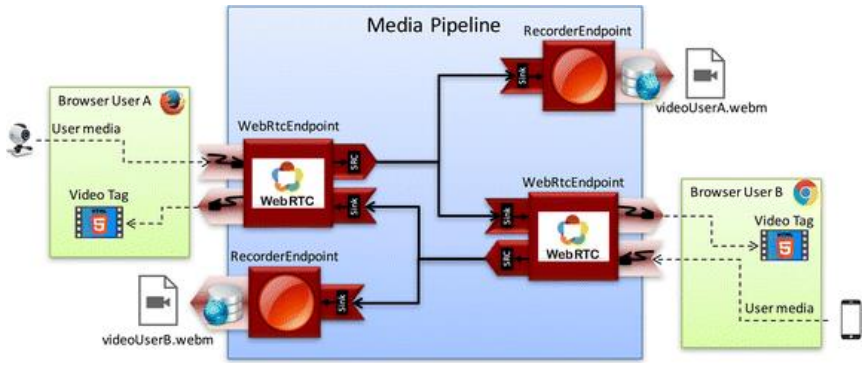
3.2. Kurento (KMS)

Na sua página de documentação (KURENTO), o Kurento exibe uma demonstração de como funciona seu modus operandis quando o assunto é gravação. Lá é detalhado, passo a passo, todo o desenvolvimento do código. Vale ressaltar que, diferente dos *hardwares* da *Polycom* citados anteriormente, essa solução é baseada em *web*.

[16]https://www.vuports.com/polycom/rss-4000/ - acesso em outubro de 2019

[17]https://www.ipphone-warehouse.com/Polycom-Capture-Station-Pro-p/2200-75500-000.htm - acesso em outubro de 2019

Figura 19 - Funcionamento de uma gravação usando KMS



Fonte: (LÓPEZ-FERNÁNDEZ et al., 2017)

A ideia base é que o fluxo de mídias será enviado ao KMS (*Kurento Media Server*, seção 2.6.3.1) que irá, ao mesmo tempo que envia de volta ao endpoint através do *WebRtcEndpoint*, gravar o fluxo (usando o *RecorderEndpoint*). Esses dois elementos fazem parte do pipeline de mídia criado para que todo esse processo funcione. Na Figura 19, fica mais claro esse processo, onde pode ser visto um usuário de *browser* A, com uma uma *webcam*, se comunicando com um usuário *browser* B, com um *smartphone*, através do *WebRtcEndpoint* ao mesmo tempo que cada um está gravando a comunicação usando o *RecorderEndpoint*.

4. Planejamento

O passo inicial será definir a arquitetura do sistema de gravação. Para isso será necessário decidir qual seria o comportamento do gravador e como ele se comunicará com as demais estruturas através de estudos de caso. Feito isso, será construída a ideia da página *web*, definida como portal, que funcionará, entre outras funções, como controle de início e parada das gravações. Haverá, em seguida, uma etapa de validação do sistema através de testes e simulações a serem definidas.

A Tabela 1 apresenta uma estimativa (em semanas) do período que cada etapa levará. Cada célula contém o número de semanas estimadas para cada etapa.

Tabela 1 - Cronograma estimado do projeto

	SETEMBRO	OUTUBRO	NOVEMBRO	DEZEMBRO
IMPLEMENTAÇÃO	4	2	-	-
VALIDAÇÃO	-	2	3	-
ESCRITA	-	-	1	-
APRESENTAÇÃO	-	-	-	2

Fonte: O autor

Referências

CAMARILLO, G.; OTT, J.; DRAGE, K. **The Binary Floor Control Protocol (BFCP).** [S.l.]: RFC Editor, 2006. RFC 4582. (Request for Comments, 4582).

DAENGSI, T.; WUTIWIWATCHAI, C.; PREECHAYASOMBOON, A.; SUKPARUNGSEE, S. (2012). **Speech Quality Assessment of VoIP: G.711 VS G.722 Based on Interview Tests with Thai Users.** International Journal of Information Technology and Computer Science. 4. 19-25. 10.5815/ijitcs.2012.02.03.

GROZEV, B. et al. **Last n: Relevance-based selectivity for forwarding video in multimedia conferences.** In: Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video. New York, NY, USA: ACM, 2015. (NOSSDAV '15), p. 19–24. ISBN 978-1-4503-3352-8.

HANDLEY, M.; JACOBSON, V.; PERKINS, C. **SDP: Session Description Protocol.** [S.l.]: RFC Editor, 2006. RFC 4566. (Request for Comments, 4566).

KURENTO. 2018. <https://doc-kurento.readthedocs.io/en/stable/tutorials/js/tutorial-recorder.html> - acesso em outubro de 2019.

LÓPEZ-FERNÁNDEZ, L., GARCÍA, B., GALLEGO, M. et al. **Multimed Tools Appl** (2017) 76: 14247. https://doi.org/10.1007/s11042-016-3729-z.

OZER, J. **"Encoding for Multiple Screen Delivery, Section 3, Lecture 7: Introduction to H.264".** Udemy. <https://www.udemy.com/course/encoding-for-multiple-screen-delivery/> - acesso em setembro de 2019.

PORTAL TIBAHIA. **Polycom: gravação de vídeo em dispositivos móveis** (2012). <http://tibahia.com/tecnologia_informacao/conteudo_unico.aspx?c=NTO_FABR&fb=B_FULL&hb=B_CENTRA&bl=LAT1&r=NTO_FABR&nid=16883>. - acesso em setembro de 2019.

ROESLER, V. et al. **Proposta de uma estratégia econômica de videocolaboração em tempo real para as NRENs.** "Transformación Digital en Instituciones de Educación Superior, Ciencia y Cultura". Octava Conferencia de Directores de Tecnología de Información y Comunicación en Instituciones de Educación Superior, TICAL2018 y II Encuentro Latinoamericano de e-Ciencia.

ROESLER, V. et al. **Desenvolvimento de um Serviço de Videoconferência com MCU (Multipoint Control Unit) na Nuvem.** "Transformación Digital en Instituciones de Educación Superior, Ciencia y Cultura". Octava Conferencia de Directores de Tecnología de Información y Comunicación en Instituciones de Educación Superior, TICAL2018 y II Encuentro Latinoamericano de e-Ciencia.

ROSENBERG, J. et al. **SIP: Session Initiation Protocol.** [S.l.]: RFC Editor, 2002. RFC 3261. (Request for Comments, 3261).

RSS 4000. Datasheet do RSS 4000, 2009: <https://www.vtkt.ru/upload/iblock/1c8/Polycom_rss-4000.pdf> - acesso em outubro de 2019.

SCHULZRINNE, H. et al. **RTP: A Transport Protocol for Real-Time Applications.** [S.l.]: RFC Editor, 2003. RFC 3550. (Request for Comments, 3550).

STATION PRO. Datasheet do Station Pro, 2014: <https://www.databox-int.com/wp-content/uploads/realpresence-capture-station-pro-ds-enus.pdf> - acesso em outubro de 2019.