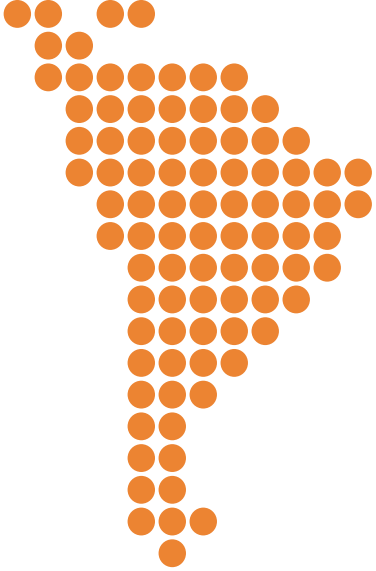

TELECORPO

mesa de corte de vídeo para redes de computadores

Pedro Sousa Lacerda
Grupo de Pesquisa Poéticas Tecnológicas

Resumo

TeleCorpo é uma ferramenta para transmissão de vídeo por redes de computadores que resultou de uma pesquisa voltada para a relação entre estudos do corpo e cultura digital na Arte em Rede. Foi testado e utilizado ao vivo em um espetáculo internacional, síncrono e distribuído de Dança Telemática, mostrando-se uma ferramenta de poucos requisitos, simples instalação, fáceis arquitetura e implementação, além de relevante à excelência deste tipo de evento. Esse artigo discute o TeleCorpo considerando sua motivação, tecnologias subjacentes, codificação, limitações e orientações para o uso por outros pesquisadores e técnicos da Arte em Rede.

Palavras-chave:

Dança telemática, Arte em rede, Transmissão de vídeo, Programação de computadores, TeleCorpo.

Lacerda, Pedro. TeleCorpo: mesa de corte de vídeo para redes de computadores. Revista Eletrônica MAPA D2 - Mapa e Programa de Artes em Dança (e performance) Digital, Ivani Santana (Org) Salvador: PPGAC, Nov. 2015; 2(2): 272-285.

Esse texto possui hiperligações para facilitar o acesso ao leitor às referências e literatura relevante. Uma versão simplificada deste artigo encontra-se na página do projeto.

Introdução

TeleCorpo é uma ferramenta para transmissão de vídeo pela internet ou rede local, foi desenvolvida a partir de uma pesquisa de iniciação científica no Grupo de Pesquisa Poéticas Tecnológicas: corpoaudiovisual (GP Poética), coordenado pela Profa. Ivani Santana, que utiliza tecnologias digitais na Arte, voltando-se para a relação entre estudos do corpo e cultura digital em articulação com várias áreas do conhecimento.

O TeleCorpo foi criado com o principal objetivo de melhorar a qualidade dos espetáculos telemáticos do GP Poética e foi testado e utilizado ao vivo em eventos síncronos internacionais de Dança Telemática, através de redes de computadores acadêmicas de alto desempenho. Mostrou-se uma ferramenta rápida, de poucos requisitos, simples instalação, fáceis arquitetura e implementação, e importante à excelência da sincronia das transmissões.

Este artigo comunica a motivação por trás do desenvolvimento da ferramenta, descreve detalhes de sua arquitetura e implementação, possibilidades de utilização, suas vantagens, limitações e erros, além de alguns fatores que influenciam a qualidade na transmissão de vídeo.

TeleCorpo

TeleCorpo distingue-se pela boa tolerância à perda de pacotes, compatibilidade com programas artísticos como Pure Data e Max, e por transmitir eventos multicâmeras ao vivo pelo Youtube. Pode ser entendida com uma mesa de corte de vídeo, na qual cada ponto de exibição pode alternar entre câmeras espalhadas pela rede. É produto de uma pesquisa realizada pouco após o espetáculo de dança telemática *Embodied in Varios Darmstadt 58*, no Grupo de Pesquisa Poéticas Tecnológicas da Universidade Federal da Bahia, e foi desenvolvido para o *Personare*, semelhante espetáculo apresentado em simultâneo e ao vivo entre Brasil, Chile e Portugal em 27 e 28 de setembro de 2014. A ferramenta representou uma iniciativa com intenções de contribuir à excelência dos espetáculos telemáticos do Grupo.

Outras ferramentas para transmissão de vídeo são: Arthron (Vieira *et al.*, 2012), LoLa (Drioli, 2013), Open Broadcaster Software, Scenic, Snowmix, UltraGrid (Gharai *et al.*, 2006), etc. Exceto para o Youtube, TeleCorpo é incapaz de transmitir áudio, para essa função podem ser utilizados o JackTrip, NetJack, etc., ou outras das ferramentas acima.

Transmissão de vídeo

> *Representing video material in a digital form requires a large number of bits. The volume of data generated by digitising a video signal is too large for most storage and transmission systems (despite the continual increase in capacity and transmission 'bandwith'). This means that compression is essential for most digital video applications.* (Richardson, 2002: 27)

Para transmitir vídeo através de uma rede de computadores, assim como qualquer outro tipo de informação, é necessário codificá-lo em bits. Esta transformação é feita por um complexo algoritmo que, usualmente, além de codificar os dados, também os comprime de modo a reduzir a informação a ser trafegada. Por sua vez, o receptor utiliza um outro algoritmo complementar capaz de decodificar e descomprimir os bits em imagens. Este par de algoritmos codificador/decodificador é chamado *codec*.

A escolha do *codec* e dos parâmetros que o configura influenciam fortemente tanto o atraso da transmissão (sendo mais ou menos eficiente), quanto a qualidade da imagem (comprimindo ao ponto de perder muitas informações), ou a quantidade de informação a ser trafegada (maior ou menor capacidade de compressão). Desde Richardson (2002), a 'largura de banda' cresceu significativamente, fazendo sentido existirem aplicações como LoLa e UltraGrid, que não utilizam (ou têm essa possibilidade) *codecs* para reduzir o custo computacional, diminuindo a latência entre o capturar, transmitir e exibir (Drioli *et al.*, 2013; Gharai *et al.*, 2006).

A potência dos computadores causa impacto no atraso (latência, *delay*) da transmissão porque custa um tempo capturar, codificar, decodificar e exibir. Computadores mais potentes podem realizar estas tarefas mais rapidamente. Comprimir os quadros/*frames* (codificar) custa mais processamento do que descomprimí-los (decodificar), e por isso exibir é mais "leve" do que capturar. Segundo Drioli *et al.* (2013), a escolha de outros equipamentos, como a câmera e placa de captura, também impacta fortemente a latência entre o capturar, transmitir e exibir.

Um terceiro fator influente na latência é a qualidade de implementação da ferramenta e dos algoritmos usados por ela, sendo indissociável o conceitual (arquitetura, algoritmos, etc.) do concreto (implementação, ferramentas, etc.).

Naturalmente, também se espera que algumas características da própria rede influenciem a qualidade da transmissão, como descritas na tabela a seguir:

Característica da rede	Impacto na transmissão	
Delay	Atraso na transmissão	Para levar um pacote de dados de um local até outro, ele precisa percorrer uma distância e atravessar equipamentos de rede.
Perda de pacotes	Degradação da imagem	Perdas superiores a 1% podem inviabilizar a transmissão, dificultando a reconstrução correta da imagem pelo decodificador, então o *codec* é parametrizado de modo a reduzir a eficácia da compressão, aumentando a redundância e tornando as perdas menos relevantes.
Jitter	Pequeno atraso na transmissão	Devido à natureza das redes de computadores, pacotes podem chegar fora de ordem, uns mais atrasados do que outros. Portanto um cache/*buffer* aguarda por pacotes atrasados para evitar a reconstrução errônea da imagem, mas descarta os muito atrasados, que são considerados perdidos.

Essas características estão usualmente fora do controle do artista e dependem do estado e infraestrutura de rede, como deterioração dos equipamentos e se está ou não sobrecarregada, por exemplo. Entretanto algumas ferramentas podem auxiliar no diagnóstico, como: o ping (medição superficial da latência e perdas); iperf (medição das perdas, *jitter* e quantidade de banda disponível); traceroute (identificação da rota, ou seja, dos equipamentos de rede entre dois locais); e mtr (identificação da rota e medição superficial da latência e perdas para cada equipamento na rota).

Arquitetura, implementação e decisões técnicas

O desenho arquitetural da versão atual (v0.92) do TeleCorpo consiste em três módulos essenciais para o funcionamento do programa, e um quarto, utilizado na transmissão para o grande público fora dos palcos, como observado na tabela a seguir.

Módulo	Descrição
tc.producer	Captura uma ou mais câmeras e as disponibilizas como fluxos RTSP.
tc.viewer	Mesa de corte de vídeo que alterna entre os fluxos disponibilizados.
tc.server	Gerencia fluxos ativos.
tc.youtube	Transmite vídeo para o grande público.

O protocolo escolhido foi o RTSP, semelhante ao HTTP, mas que transmite conteúdo audiovisual em vez de hipertexto, usualmente através dos protocolos subjacentes RTP e UDP (Schulzrinne, 1998). Entretanto a vantagem da escolha foi o fato do RTSP disponibilizar os conteúdos (fluxos, *streams*) por uma URL e a sua grande compatibilidade com diversos tocadores multimídias, tornando a ferramenta mais acessível para usuários não técnicos.

Os *frameworks* multimídias escolhidos foram o GStreamer1.0 e o gst-rtsp-server, ambos escritos na linguagem de programação C. Devido à morosidade em se desenvolverem aplicativos nesta linguagem, TeleCorpo foi desenvolvido em Python3, utilizando tais *frameworks* através de *bindings* gerados automaticamente pelo *middleware* GObject Introspection.

O *codec* escolhido foi o H.264 por nenhum motivo especial além da sua ampla popularidade, compatibilidade e relativa modernidade. *Codecs* mais modernos são capazes de comprimir melhor as imagens, reduzindo o consumo de banda, tipicamente ao custo de mais processamento. Quando o codificador envia um *key-frame* (*IDR picture*), isto é, um quadro completo equivalente a uma fotografia, os quadros subsequentes serão incompletos, mas complementares, de modo a atualizar a imagem com os movimentos que seguem. (ITU, 2014) Quando acontecem perdas de pacotes, a imagem de referência é perdida, sendo necessário receber outra *IDR picture* para reconstruir a imagem corretamente.

A forma utilizada para aumentar a resiliência do H.264 às perdas foi com a diminuição da maior distância (temporal) entre dois *key-frames*. Logo, quando ocorrem perdas, rapidamente o decodificador poderá recuperar-se recebendo novamente uma imagem completa. No GStreamer este comportamento é controlado pela opção key-int-max. No x264, biblioteca utilizada pelo GStreamer, é controlado pela opção -keyint.

De acordo com Hunt & Thomas (1999), o código do TeleCorpo não poderia ser considerado espaguete[1], pois mudanças num módulo não interferem em outros já que não há dependências explícitas entre eles. Acontece portanto um baixíssimo acoplamento. Mas há dependências em tempo de execução que ocorrem na dinâmica do sistema, exigindo uma ordem específica para inicialização dos componentes.

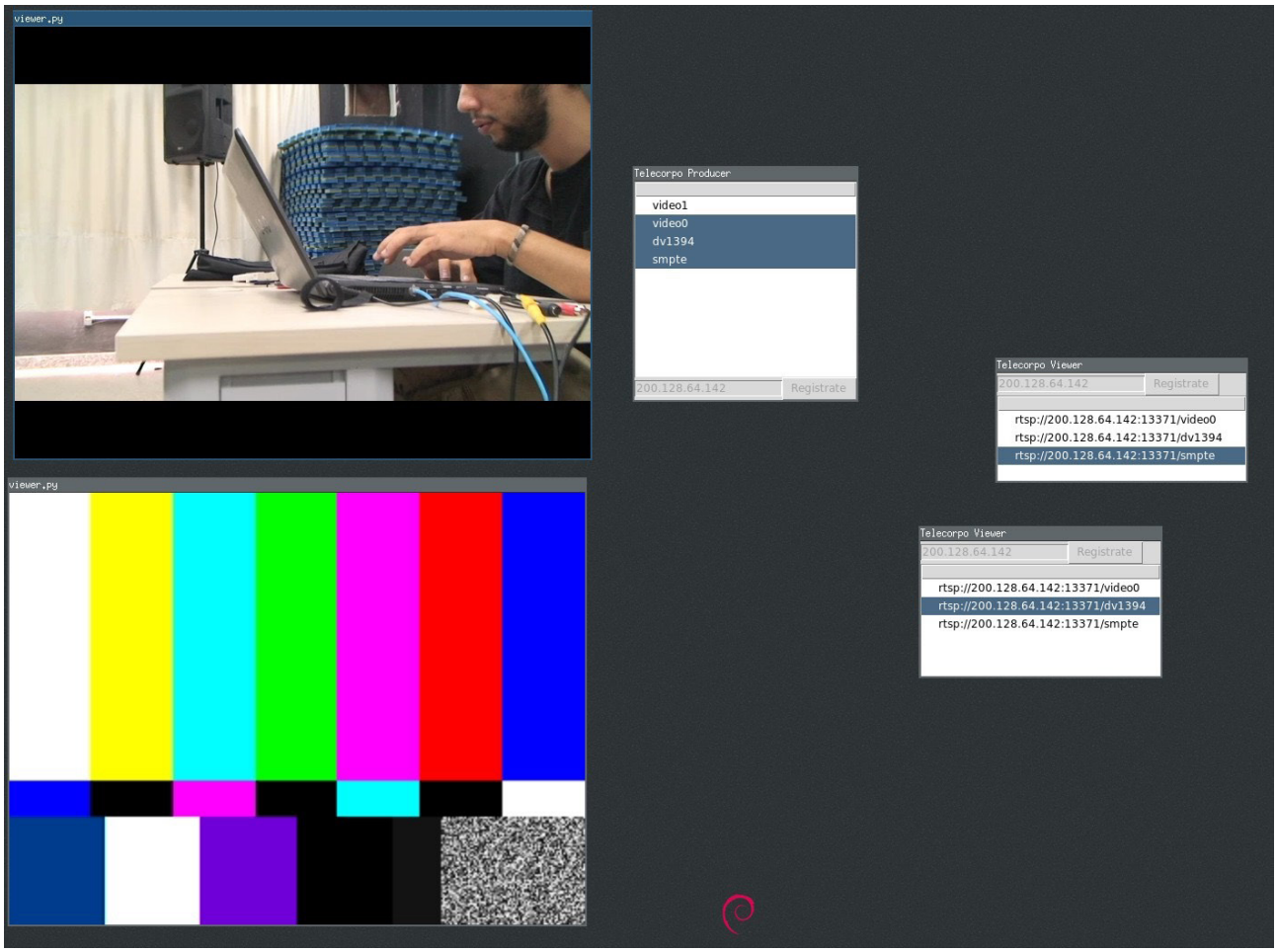

Figura 1. TeleCorpo em ação com um producer e dois viewers.

Com o *server* em execução, o *producer* registra nele os fluxos produzidos (câmeras capturadas). Então o *server* passa a consultar periodicamente cada URL de fluxo para verificar se ainda está ativa, e desregistrá-la caso a consulta falhe. Já o *viewer* é mais simples, apenas inquire periodicamente as URLs ainda ativas no server, isto é, ainda não encerradas. Na **Fig. 2** e **Fig. 3**, vê-se exemplos de dois sistemas distintos demonstrando como ocorre a troca de mensagens.

Já o módulo youtube, utilizado para transmitir vídeos ao vivo para o Youtube Live, é completamente independente dos demais, podendo ser utilizado para transmitir conteúdo que não foi produzido pelo próprio TeleCorpo. O módulo foi baseado em scripts feitos por MAERSK-MOLLER (2013) para o projeto Snowmix. A escolha de transmitir via uma distribuidora de conteúdos ao invés de utilizar recursos próprios foi para evitar a saturação da rede no caso de muitos espectadores.

Instruções de uso

Basta apenas uma instância do *server* para o funcionamento do sistema. Os *producer*s e *viewers* podem ser inicializados, tantos quantos forem necessários. Se há duas câmeras em dois países diferentes, por exemplo, precisará inicializar dois *producers*, um em cada país. Similarmente, se há três projetores em três países diferentes, precisará inicializar três *viewers*, um em cada país.

Devido a limitações da implementação e pouca investigação não é possível o uso do TeleCorpo (especificamente o *producer* e *server*) em máquinas protegidas por *firewalls*.

Além de ser exigida uma ordem específica para inicialização dos módulos devido à própria natureza dos sistemas distribuídos, existem defeitos de implementação que reafirmam a necessidade de cuidados na inicialização do sistema. O módulo *server* é sempre o primeiro a ser executado, já que *producers* e *viewers* se registram nele. Então, teoricamente, quantos forem demandados, *producers* e *viewers* poderiam ser inicializados em qualquer ordem, mas devido a discrepâncias entre a arquitetura e implementação, deverá ser executado primeiro os *producers*, para depois os *viewers*.

Na Fig. 4 tem-se como exemplo três nós (ou nichos), cada um possuindo uma quantidade diferente de câmeras e projetores. Nesta configuração seriam necessários quatro *viewers*, um para cada projetor. Mas *producers*, como cada um é capaz de capturar múltiplas câmeras, bastariam apenas três, um por nicho.

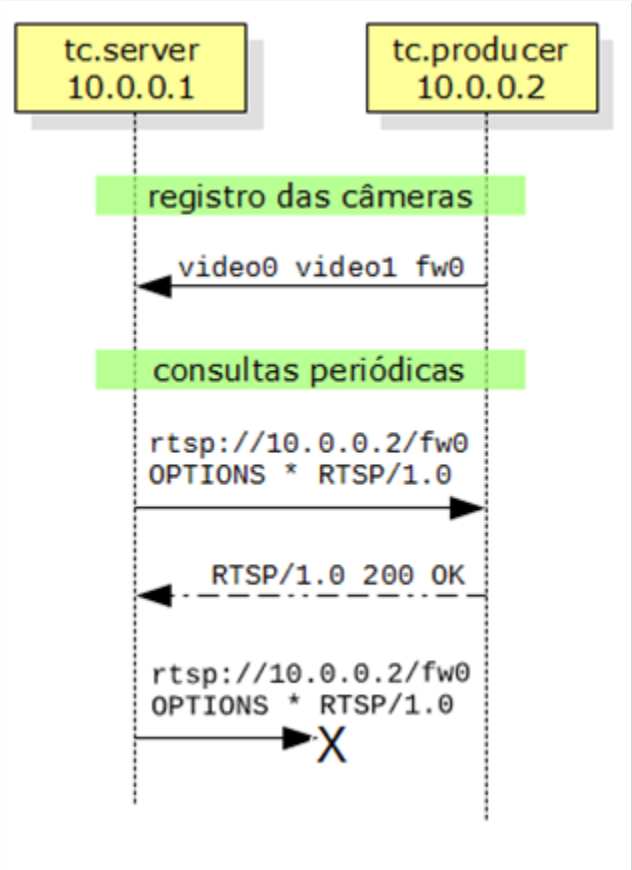

Figura 2. registro de um producer com três câmeras, depois troca de mensagens relativas a uma destas câmeras até o producer ser encerrado, quando a consulta falha e o producer é desregistrado.

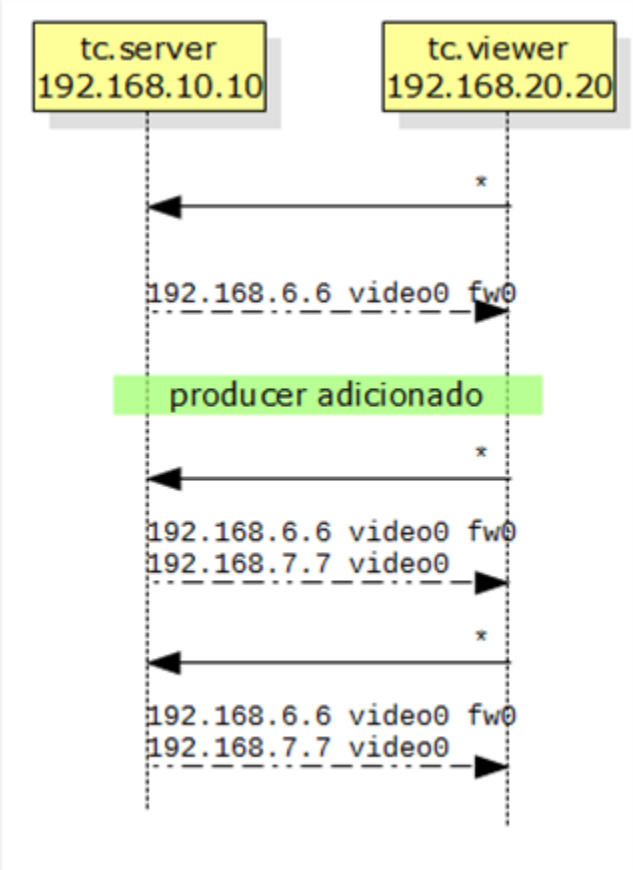

Figura 3. um viewer inquerindo pelos producers ativos, quando então um segundo producer é registrado no server.

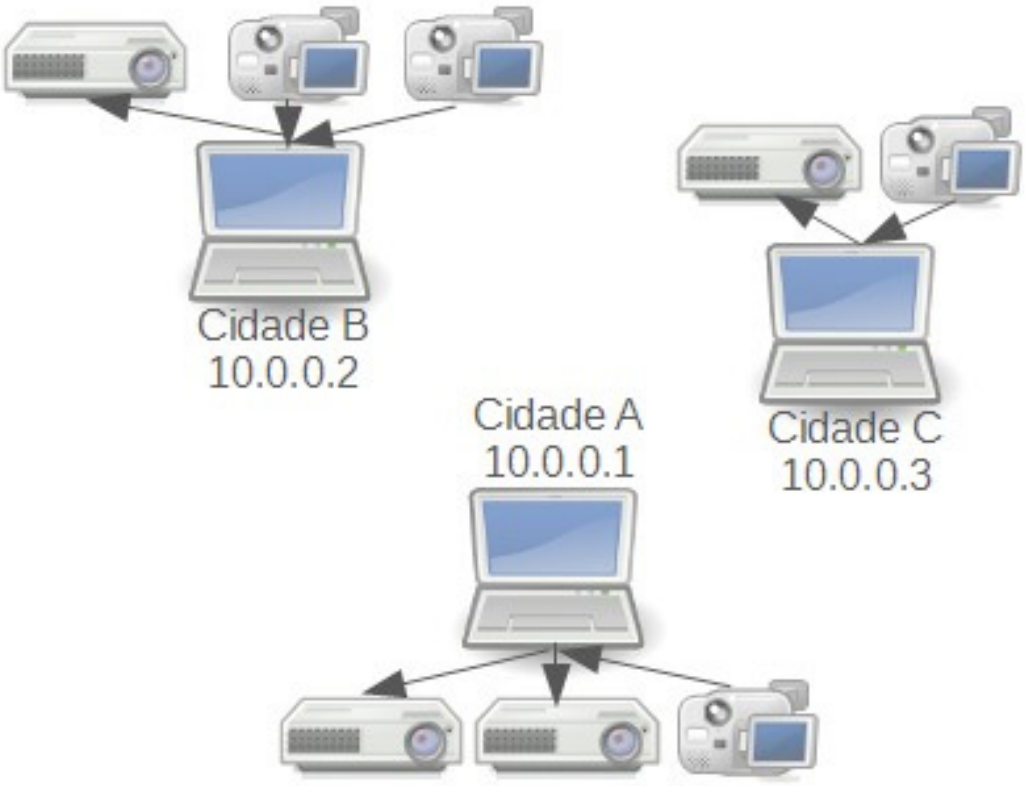

Figura 4. Exemplo de configuração possível.

O uso do módulo *tc.youtube* exige a criação de um evento ao vivo no Youtube Live, e, em sequência, sua execução configurando-o com alguns valores obtidos na página de configuração do evento:

```
$ python3 -m tc.youtube -r 360p -t my_name.
a1b2-c3d4-e5f6-g7h8 \

        rtsp://10.0.0.1:13371/video0
```

Se o parâmetro -b for utilizado, o evento será também transmitido para o servidor de *backup*, além de para o servidor principal do Youtube. Este módulo exige ter uma instância do servidor de áudio JACK Audio Connection Kit em execução na mesma máquina. É também possível transmitir eventos independente do TeleCorpo ajustando a URL; a primeira webcam de um computador, por exemplo, é usualmente disponível pela URL v4l2:///dev/video0.

Recepção de vídeo por aplicativos externos

O principal motivo da escolha do protocolo RTSP foi sua ampla compatibilidade com diversos programas multimídias. Aplicações baseadas em QuickTime, por exemplo, herdam a capacidade de digerir tais tipos de fluxos, como o Max, no qual é possível configurar o objeto *[jit.qt.movie]* ou *[jit.movie]* com uma mensagem semelhante a [read rtsp://10.0.0.1:13371/fw0(.

O Pure Data não é naturalmente capaz de digerir tais fluxos, mas, dentro de uma mesma máquina, é possível utilizar o programa v4l2loopback como ponte entre o Pure Data e outros aplicativos de vídeo. A solução encontrada foi criar dispositivos virtuais de vídeo (dev, devices) com o v4l2loopback, recepcionar o fluxo RTSP por um programa qualquer, redirecioná-lo para o dispositivo criado, e consumi-lo no Pure Data pelo objeto [pix_film] ou [pix_video] do Gem com uma mensagem semelhante à [device 10(:

```
$ sudo modprobe v4l2loopback video_nr=10
```

```
$ gst-launch-1.0 rtspsrc
location=rtsp://10.0.0.1:13371/fw0 latency=30 !
decodebin \

      ! queue ! v4l2sink sync=false device=/
dev/video10
```

Desta maneira é inclusive possível recriar com facilidade a funcionalidade de mesa de corte de vídeo, tanto no Max, quanto no Pure Data, descartando o módulo *tc.viewer*. A propriedade *latency* do comando prévio determina o

tamanho do cache/*buffer* em milisegundos utilizado na recepção, qual deverá ser ligeiramente maior que o *jitter* da rede para suportar turbulências.

Considerações finais

Sendo projetado especificamente para o espetáculo *Personare*, TeleCorpo supre muitas das necessidades de transmissão de vídeo para danças telemáticas desenvolvidas nos moldes do Grupo de Pesquisa Poéticas Tecnológicas: corpoaudiovisual. Graças aos *frameworks* GStreamer e gst-rtsp-*server* e à tecnologia de codificação utilizada, mostrou-se extremamente eficiente, resiliente à perda de pacotes, e com baixíssima latência quando comparado a outros programas de transmissão. Apesar de não ter sido realizado nenhum estudo quantitativo, a velocidade de transmissão rivaliza com outros programas relevantes da área. No entanto, é esperado que outros aplicativos projetados especificamente com o propósito de atingirem baixa latência tenham um melhor desempenho neste quesito, como UltraGrid e LoLa.

Uma vantagem do TeleCorpo sobre outros programas é o seu pequeno tamanho e arquitetura simplificada, permitindo a rápida compreensão por desenvolvedores júniores ou seniores, e acomodando modificações com facilidade.

Perspectivas futuras incluem transmitir vídeo por tecnologia *multicast*, evitando tráfego redundante, isto é, reduzindo o consumo de banda. Esta modificação permitiria simplificações na arquitetura, tornando desnecessário o módulo tc.server. A adoção de *codecs* mais modernos não é desejável devido a baixa compatibilidade com aplicativos externos.

Notas

1 Código espaguete: código emaranhado, com pouca estrutura, o controle pula incorentemente de um trecho para o outro

Referências

DRIOLI, C. ; ALLOCCHIO, C. ; BUSO. *Networked performances and natural interaction via LOLA: Low latency high quality A/V streaming system. Information Technologies for Performing Arts, Media Access, and Entertainment*. Springer Berlin Heidelberg, p. 240-250. Disponível em <http://www.internetsociety.org/sites/default/files/pdf/accepted/32_LOLA.pdf>Acesso em: 21 ago. 2015.

Gharai, Ladan, et al. "Experiences with high definition interactive video conferencing". *Multimedia and Expo, 2006 IEEE International Conference on. IEEE*, 2006.

HUNT, A. ; THOMAS, D. *O Programador Pragmático: de aprendiz a mestre*. Bookman, 2010.

ITU, Telecommunication Standartization Sector. H.264: *Advanced video coding for generic audiovisual services*. 2014. Disponível em <https://www.itu.int/rec/T-REC-H.264-201402-I> Acesso em: 21 ago. 2015.

MAERSK-MOLLER, P. *Snowmix and CDNs*. 2013. Disponível em: <http://sourceforge.net/p/snowmix/wiki/Snowmix%20and%20CDNs/>. Acesso em: 21 ago. 2015.

RICHARDSON, I. E. G. *Video Codec Design: Developing Image and Video Compression Systems*. Chichester: John Wiley & Sons, 2002.

SCHULZRINNE, H. ; RAO, A. ; LANPHIER, R. "RFC 2326: Real Time Streaming Protocol (RTSP)". *The Internet Society*. 1998. Disponível em <https://tools.ietf.org/html/rfc2326>. Acesso em: 21 ago. 2015.

VIEIRA, E. ; PASSOS, M. ; SANTOS, B. A. ; OLIVEIRA, S. S. ; MELO, E. A. ; MOTTA, G. H. M. B. ; TAVARES, T. A. ; SOUZA FILHO, Guido Lemos de. "Uma Ferramenta para Gerenciamento e Transmissão de Fluxos de Vídeo em Alta Definição para Telemedicina". *In: Salão de Ferramentas do Simpósio Brasileiro de Redes de Computadores 2012*, 2012, Ouro Preto. Anais do Salão de Ferramentas do Simpósio Brasileiro de Redes de Computadores 2012. v. 1, 2012. Disponível em <http://gtavcs.lavid.ufpb.br/wp-content/uploads/2011/11/ARTHRON_2-O_salao-ferramentas-SBRC-2012_final.pdf>. Acesso em: 21 ago. 2015.

Sobre o autor

Pedro lacerda é Programador e usuário de computadores. É bacharel em Ciência & Tecnologia pela Universidade Federal da Bahia. Possui experiência em linguagens de programação na etnomatemática, desenvolvimento de sistemas, modelagem e simulação de sistemas para biofisicoquímica farmacêutica, e transmissão e codificação de vídeo para dança digital. Atualmente investiga inteligência artificial e suas aplicações. No seu tempo livre se diverte com interfaces entre diferentes linguagens de programação.