Universidade de Aveiro
2012
Departamento de Eletrónica, Telecomunicações e Informática

Miguel Gerardo Assis do Souto

Implementação de um Sistema Simples de Vídeo-Conferência

Universidade de Aveiro
2012
Departamento de Eletrónica, Telecomunicações e Informática

Miguel Gerardo Assis do Souto

Implementação de um Sistema Simples de Vídeo-Conferência

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica do Doutor Paulo Salvador e do Doutor António Nogueira, Professores Auxiliares do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri

presidente

Prof. Doutor Amaro Fernandes de Sousa

professor auxiliar da Universidade de Aveiro

Prof. Doutor Joel José Puga Coelho Rodrigues

professor auxiliar da Universidade da Beira Interior

Prof. Doutor Paulo Jorge Salvador Serra Ferreira (orientador)

professor auxiliar da Universidade de Aveiro

Prof. Doutor António Manuel Duarte Nogueira

professor auxiliar da Universidade de Aveiro

agradecimentos

Para a realização deste trabalho agradeço a Liliana, minha companheira, aos meus pais e irmã pelo seu apoio, agradeço também aos meus orientadores sem os quais a realização desta dissertação não seria possível. Agradeço também ao Isaac, ao Carlos e a Sónia e a todos os que de alguma maneira contribuíram na minha dissertação e percurso académico.

palavras-chave Vídeo-conferência, baixo custo, código aberto, H323.

resumo Esta dissertação propõe-se contribuir para a implementação um sistema modular de vídeo-conferência simples, intuitivo, de baixo custo e de código aberto. O sistema deverá ser capaz de interagir em múltiplas camadas protocolares com os sistemas de vídeo-conferência comerciais. Este documento descreve a implementação e teste do módulo de comunicação H.323 e é composto pela introdução da importância dos sistemas de videoconferência nos dias de hoje, seguido da descrição do conjunto de protocolos H.323, a que se segue uma introdução às bibliotecas libopal e gtk utilizadas para o desenvolvimento do modulo de comunicação H.323. Finalmente, é descrita a implementação e estrutura do módulo implementado, sua apresentação e utilização.

keywords Video-conference, low cost, open source, H323.

abstract This dissertation objective is to contribute to the implementation of a modular simple, intuitive, low-cost and open source video-conference system. The system should be able to interact in multiple protocol layers with the commercial video conferencing systems. This document describes the implementation and testing of the system's H.323 communication module. The document starts with an introduction about existing videoconferencing systems, followed by a description of the H.323 protocol suite, an introduction to the libopal libraries and the gtk used in the development of H.323 communication module. Finally, the document describes the structure, implementation, appearance and usage of the developed module.

Conteúdo

1 Introdução

Os sistemas de videoconferência são hoje uma área de forte crescimento no desenvolvimento de sistemas telecomunicações. Este impulso é resultado de várias necessidades e interesses, tanto por parte da indústria de telecomunicações, como dos consumidores e da sociedade.

O rápido crescimento da tecnologia por parte dos produtores de Hardware ou dispositivos físicos de videoconferência tem um grande impacto neste desenvolvimento. A cada vez maior disponibilidade destes dispositivos, assim como a cada vez menor despesa associada à aquisição deste tipo de comodidades, qualquer que seja a sua forma (telefones,telefone com vídeo, telemóveis, *smarphones*, computadores portáteis ou fixos, Televisões, *Webcams*), possibilitou e alavancou uma mudança na nossa sociedade e no nosso modo de vida. Na atualidade podemos falar em videoconferência com os nossos familiares, amigos, parceiros profissionais ou clientes a partir de uma cada vez maior área geográfica, usando um *smartphone* e as redes 4G da nossa operadora, ou mesmo utilizar a nossa televisão que já integra um sistema de vídeo conferência.

A necessidade que os utilizadores sentem de possuir uma cada vez maior facilidade na integração destes sistemas entre si e com as redes de telecomunicações (PTSN ou PSN) é saciada pela indústria e pelas empresas de comunicações, sempre no sentido de corresponder aos desejos e exigências de uma cada vez maior e mais bem informada base de utilizadores. Estes utilizadores requerem melhores comunicações a menores preços e com mais alternativas, possibilidades e convergências. Esta mudança de paradigma em relação à antiga oposição de forças de várias vertentes (telecomunicações, informática, consumidor) levou à fusão da indústria e serviços de telecomunicações a níveis muito superiores aos que eram possíveis de prever à alguns anos atrás. Os sistemas de videoconferência são hoje uma emergente e quase omnipresente necessidade em qualquer serviço de telecomunicações ou informação. Estes serviços serão cada vez mais utilizados quanto maior for a sua conveniência, versatilidade, diversificação e originalidade.

Citando John Wooden

“It isn't what you do, but how you do it.”

podemos também dizer

“It isn't what you comunicate, but how you comunicate”

Esta importância cada vez maior deste tipo de aplicação e serviço é a principal motivação para esta dissertação. Assim, pretende-se implementar um sistema de videoconferência de baixo custo, código aberto e de fácil utilização, que possa ser utilizado por qualquer tipo de utilizador, seja ele doméstico ou empresarial. A principal contribuição desta dissertação foi o desenvolvimento de um módulo de comunicação vídeo utilizando a camada de protocolos H.323.

Os assuntos abordados nos capítulos seguintes são: no capítulo 2 é feita uma resenha do estado da arte relacionado com os diversos tópicos abordados nesta dissertação: sistemas de videoconferência e bibliotecas; no capítulo 3 é feita a apresentação exaustiva da pilha protocolar H.323; o capítulo 4 descreve a implementação do módulo H.323 que foi desenvolvido; o capítulo 5 descreve as funcionalidades do módulo desenvolvido; finalmente, o capítulo 6 apresenta as principais conclusões do trabalho realizado e aponta algumas direções para trabalho futuro.

2 Estado da arte

2.1. Sistemas de videoconferência

Desde a década de 90 que os termos VoIP - *voice over internet protocol,* Telefonia Internet e Telefonia IP*,* têm sido apresentados como uma revolução nas telecomunicações, possibilitando a convergência das redes historicamente separadas de dados e de voz, reduzindo custos, e integrando voz, dados e vídeo nas mesmas aplicações.

O termo VoIP significa transportar voz por uma rede PSN, e refere os aspectos da digitalização de um sinal áudio e sua fragmentação e transmissão através da rede. O termo telefonia IP significa a comunicação de voz, média e dados numa rede (PSN), e refere a comunicação de vários médias, incluindo áudio e vídeo. O termo telefone Internet significa a realização de chamadas através da Internet e pode usar VoIP, mas também outros tipos específicos e proprietários de Software e Hardware, e representa uma forma mais global de telefonia IP que se estende a redes de larga escala e à própria Internet.

A primeira implementação de um sistema para transmitir voz numa rede de dados foi efetuada com o NVP - *Network Voice Protocol* na ARPANET em 1995. A companhia VOLTATEC apresentou o primeiro software chamado *"internet phone"* que corria num computador equipado com placa de som, colunas , microfone e um modem. As suas limitações eram a necessidade de ambos os interlocutores de uma chamada telefónica precisarem de usar o mesmo software e hardware, além da qualidade de som ser muito inferior ao de uma chamada utilizando os meios tradicionais.

Em 1996, as companhias de serviços telefónicos nos estados unidos pedem ao congresso a proibição da tecnologia de Internet *phone.* Em 1998, o VoIP já começava a progredir e demonstrava potencial com a existência de alguns *gateways* que permitiam chamadas *computer-to-phone* e também a existência de algumas soluções IP para chamadas *phone-to-phone.* Hoje em dia, na maioria dos países desenvolvidos e em desenvolvimento, existe já acesso de Internet de banda larga para consumidores domésticos, permitindo o alastramento alargado de tecnologias VoIP. Também as aplicações de *instant messaging* que combinam voz e vídeo são cada vez mais comuns e apreciadas. Os produtores dos principais sistemas operativos (Microsoft®, Apple® e

Linux®) também providenciam suporte para comunicações VoIP, potenciando assim o aumento na qualidade e no número destes sistemas.

A telefonia IP traz muitas vantagens em relação ao utilizador de um sistema convencional (PSTN), nomeadamente em termos de custo (os provedores de VoIP tendem a oferecer preços mais atrativos principalmente em chamadas internacionais), flexibilidade (para um sistema PTSN não é possível transportar o seu sistema, mas para telefonia IP é possível e basta instalar o seu sistema num local servido com uma rede Internet*).* Uma das partes mais importantes de uma chamada de telefone é o estabelecimento da mesma. Numa rede de dados por pacotes isto é executado pelo protocolo de sinalização, dos quais os mais importantes são o H.323 e o SIP.

2.2. Protocolos e Bibliotecas

Existem vários protocolos e bibliotecas para a implementação de sistemas de videoconferência. Os protocolos mais globalmente utilizados são os protocolos H.323 e SIP [5,12,16]. Ambos coexistem já há alguns anos, ainda que o protocolo H.323 tenha entrado em vigor alguns anos antes do protocolo SIP. O Protocolo H.323 foi divulgado na sua primeira versão em novembro de 1996 e já recebeu varias revisões, estando neste momento em vigor a revisão nº7 de dezembro 2009. É desenvolvido e mantido pelo ITU-T, ramo de normalizações do ITU - *International Telecommunications Unit,* principal organização na implementação de normas de telecomunicações a nível mundial [10]. O protocolo SIP divulgado em 2002 pelo IETF – *Internet Engineering Task Force,* organização que regula o desenvolvimento da *Internet,* recebeu a sua ultima revisão em julho de 2012. Ambos os protocolos refletem o saber cumulativo e tradições das organizações que os desenvolveram [16].

2.2.1. Seleção de um protocolo

A escolha do protocolo no decorrer desta dissertação baseou-se em vários critérios, sendo estes suporte, complexidade, versatilidade, atualidade. Entre algumas possibilidades considerámos várias hipóteses, entre as quais: H.323, SIP, XMPP. A escolha preferencial recaiu sobre o protocolo H.323 que, apesar dos anos, é ainda atual, recebe revisões e atualizações e continua a ser dos mais usados.

2.2.2. Seleção das bibliotecas

Os critérios de seleção das bibliotecas utilizadas na implementação da aplicação para esta dissertação foram: compatibilidade, suporte, atualidade, complexidade, *bindings*, consistirem em software livre, e portabilidade.

A biblioteca do interface gráfico selecionada foi a biblioteca GTK, porque: (i) é muito utilizada principalmente em sistemas operativos Linux; (ii) possui compatibilidade com um grande número de bibliotecas, permitindo uma maior versatilidade; (iii) possui *bindings* com as linguagens de programação conhecidas, facilitando a utilização da biblioteca; (iv) possui compatibilidade com outros sistemas operativos, tornando possível a portabilidade para um maior número de utilizadores e sistemas; (v) existem vários fóruns de desenvolvimento para esclarecimento e aprendizagem da sua utilização.

A biblioteca H.323 selecionada foi a biblioteca *libopal.* As motivações para a sua seleção foram: (i) linguagem de programação em que pode ser utilizada; (ii) existência de outras aplicações que a utilizam, revelando robustez e versatilidade; (iii) uma documentação adequada e percetível; (iv) portabilidade, pois opera sobre a biblioteca libpt, permitindo a sua adaptação para outros sistemas operativos.

3 Protocolo H.323

3.1. Descrição

O protocolo H.323 [10] é uma recomendação do ITU-T que descreve as especificações de um sistema de videoconferência. Este protocolo depende de outros protocolos de comunicação que utiliza para efetuar as diferentes tarefas necessárias para o seu funcionamento.

3.2. Arquitetura

Um sistema de comunicação H.323 é constituído por vários elementos estruturais físicos, cujo objetivo é permitir a comunicação multimédia (áudio, vídeo,media) entre 2 ou mais utilizadores. Estes elementos são: Terminais, Unidades de Controlo Multi-ponto (MCU - *multipoint control units)* e *Gateways.* Estes 3 elementos também são regularmente referenciados como *Endpoints.* Além destes existem também outros elementos, nomeadamente *Gatekeepers,* elementos de fronteira e *peer elements.*

A configuração simplificada de um sistema de comunicação é constituída por 2 terminais no mínimo. Esta configuração permite a comunicação entre 2 utilizadores ou máquinas. No entanto, para aumentar as capacidades de comunicação, tanto a nível de número como de capacidades, outros elementos serão forçosamente necessários.

3.2.1. Descrição dos dispositivos

Terminais – São a unidade base dos sistemas H.323. Os dispositivos deste tipo mais conhecidos são o *IP-Phone* e, mais recentemente, o vi*deo-phone* ou um computador doméstico. Cada um destes elementos terá obrigatoriamente de possuir funcionalidades de comunicação H.323, sendo estas adquiridas pela existência de um *protocol-stack* definida para um sistema H.323. A *protocol-stack* será composta por um conjunto de protocolos definidos pelas recomendações do ITU. A Figura 1 ilustra uma implementação típica da *protocol-stack* de um equipamento H.323.

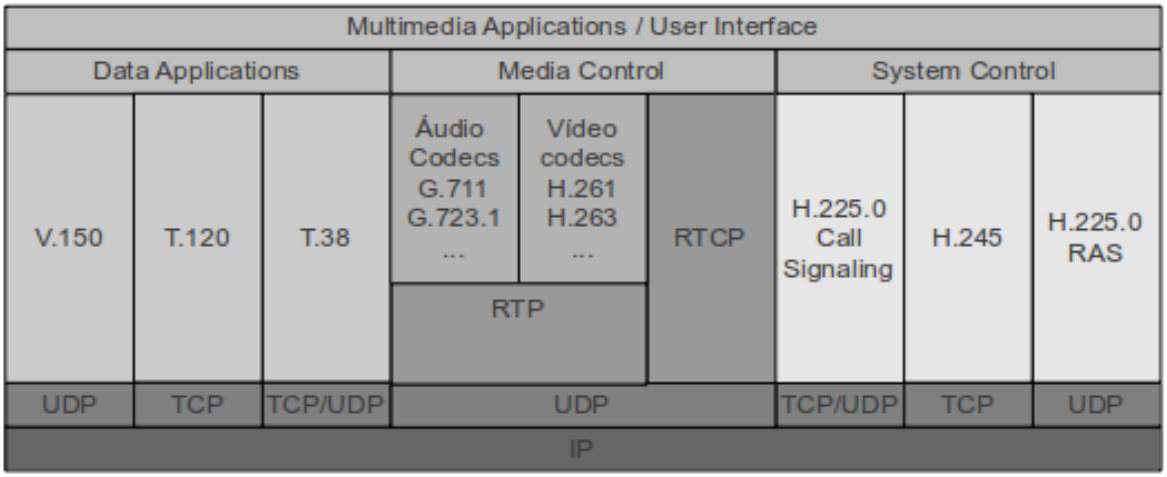

Figura 1: H.323 protocol-stack.

Unidades de controlo multi-ponto (MCU) – São os elementos que permitem a implementação de sistemas com capacidade para videoconferência, permitindo assim a um grupo de utilizadores estabelecer chamadas vídeo com mais de 2 utilizadores e em que todos os intervenientes da conferência estejam em comunicação.

Gateways – São os elementos que permitem o estabelecimento de comunicações entre redes H.323 e outras redes, como redes ISDN ou PSTN, possibilitando assim a comunicação entre vários utilizadores que utilizem equipamentos não H.323 ou cuja conexão passe por redes não H.323. Estes equipamentos são cada vez mais necessários e importantes devido à necessidade dos sistemas H.323 serem cada vez mais abrangentes, mesmo usando redes de comunicação heterogéneas. Os *gateways* também têm um papel importante porque permitem assim na comunicação entre utilizadores de diferentes serviços de modo transparente e impercetível para os utilizadores, permitindo a comunicação entre utilizadores de telefonia simples (PSTN) e utilizadores de *IP-Phone* H.323. Estes serviços têm uma importância muito grande a nível empresarial devido a ser a melhor forma logística e eficiente no funcionamento de, por exemplo, *callcenters.* A ligação entre redes H.323 e outras redes tem recebido também uma cada vez maior atenção devido à atual presença de chamadas de vídeo de dispositivos móveis, como *smartphones*, o que permite a comunicação áudio-vídeo entre estes dispositivos e utilizadores de terminais H.323.

Gatekeepers – Os *gatekeepers* são elementos que efetuam ou disponibilizam vários serviços a outros componentes de uma rede H.323, sejam eles terminais, MCUs ou *gateways.* Estes serviços, apenas 1 ou vários, incluem registo de *endpoint,* autenticação dos utilizadores, controlo de admissão. Argumentativamente, o mais importante é a

resolução de endereço que permite a dois *endpoints* poderem comunicar sem que nenhum deles possua o endereço IP do outro. Estes podem ser implementados com dois modos de sinalização distintos, cada um com o seu modo de operação. Estes dois modos possuem as suas vantagens e desvantagens, mas possibilitam entre os dois uma maior abrangência de capacidades e necessidades por parte de uma rede H.323. No primeiro *direct routed*, ou roteamento direto, o *gatekeeper* fornece aos *endpoints* o endereço IP um do outro, de modo a que estes possam ligar-se diretamente um ao outro, e assim estabelecer a comunicação. Este modo usa o protocolo RAS e é o mais eficiente no que diz respeito à utilização de recursos do *gatekeeper* e da rede H.323, razão pela qual é o mais usado e diversificado. O segundo modo *gatekeeper routed*, ou roteamento pelo *gatekeeper*, obriga os *endpoints* a comunicar através dele. Embora este modo de funcionamento seja menos eficiente e mais dispendioso dos recursos do *gatekeeper* e da rede, possibilita um maior controlo da comunicação por parte do *gatekeeper*, possibilitando uma maior diversificação de serviços prestados pelo *gatekeeper* aos *outros* elementos da rede. Para além disso, assegura uma segurança acrescida aos utilizadores devido ao desconhecimento por parte destes do endereço IP dos utilizadores em comunicação. Todo o conjunto de *endpoints* registados num *gatekeeper* forma uma zona, o que representa a área de domínio de um *gatekeeper* que controla todas as comunicações entre estes, ou com outros utilizadores que estejam registados em outros *gatekeepers*, se estas duas zonas estiverem ligadas entre si pelos *gatekeepers.*

Border elements* e *Peer elements - Os *border elements* são os dispositivos que fazem a ligação entre domínios administrativos, de modo a poder estabelecer chamadas entre utilizadores que, apesar de estarem em zonas administradas por pessoas ou entidades diferentes, estas possam estabelecer comunicações. As zonas administrativas a que correspondem zonas ou conjuntos de zonas, cuja administração é de uma única entidade ou pessoa, podem ter uma escala variável e necessitam de um *border element* sempre que estejam implementadas, de modo a poderem ligar-se umas às outras. Os *peer elements* são dispositivos distribuídos dentro das zonas administrativas e aos quais é atribuída a função de difundir a informação coletada dos *border elements,* de modo a que todos os utilizadores nas zonas administrativas possam estabelecer comunicações com outros utilizadores de outras. Estes últimos elementos são necessários quando a escala das zonas administrativas assim o justifique, não sendo utilizados, por exemplo, quando uma zona administrativa é constituída por apenas um *gatekeeper* e seus *endpoints.*

3.3. Sinalização H.323

A transmissão de informação através de uma rede H.323 é executada utilizando a sintaxe descrita pela notação ASN.1, e as regras de codificação PER - *Packet Encoding Rules*, de modo a uma comunicação eficiente nos cabos de transmissão. É efetuada utilizando outros protocolos divulgados pelo ITU-T entre os quais os mais importantes são o protocolo H.225[7] e H.245[13].

3.3.1. Sinalização H.225 mensagens e *callflow*

O protocolo H.225 [7] é utilizado em redes H.323 [10] para controlar chamadas entre os *endpoints* através da rede para estabelecer a comunicação entre estes. Este protocolo é o standard utilizado nestas redes e é responsável por *Call signaling* – Sinalização da chamada, RAS *signaling* – Sinalização *Request Admition* Status e empacotamento de media. A Sinalização de chamada H.225 visa permitir ao protocolo estabelecer, controlar e encerrar as chamadas Utiliza um conjunto de mensagens de controlo para efetuar essa tarefa. Estas mensagens são descritas no protocolo como:

Setup - esta mensagem é enviada pelo do endpoint1 (dispositivo iniciador)para o endpoint2 (dispositivo recetor), para que este tenha conhecimento que é destinatário de uma chamada.

Setup Acknowledge – mensagem de resposta a uma mensagem de *(Setup),* do enpoint2 ao endpoint1 que tem como objetivo indicar a este ultimo que são necessárias mais informações antes do estabelecimento da chamada.

Call Proceeding – Esta mensagem é enviada do endpoint2 para o endpoint1, informando-o que tem todos as informações necessárias para encaminhar a chamada para o seu destino.

Progress – mensagem que informa um *gateway* do progresso de uma chamada. Tipicamente este tipo de mensagem é utilizado em chamadas de redes heterogéneas, que passam por diferentes redes de comunicação.

Alerting – mensagem que informa o endpoint1 que o endpoint2 alertou um utilizador e que esta a espera que este autorize a chamada.

Connect – mensagem do endpoint2 para o endpoint1 que a chamada foi aceite.

User Information – mensagem que é enviada se for necessária informação adicional como, por exemplo, no caso de *overlap.* Para enviar informações relativas a chamada, pode também ser usado, para substituir a mensagem de *alerting* que em alguns sistemas de H.323 não é enviada.

Release Complete – mensagem enviada para sinalizar o facto que a chamada foi desligada.

Status Inquiry – esta mensagem é enviada para requerer informação sobre o status da chamada. Quando recebido por qualquer um dos *enpoints* indica-lhe para retornar informação especifica referente à chamada.

Status – mensagem de resposta a uma mensagem de (*Status Inquiry*) ou a uma mensagem de sinalização de chamada desconhecida.

Information – mensagem enviada com informações especificas relativas a chamada. Por exemplo, na existência de *overlap.* Cada digito é transmitido um de cada vez.

Notify – mensagem de alerta para informar um *endpoint* que uma mudança foi efetuada na chamada.

Estas mensagens são são transmitidas sobre TCP, exceto no caso da mensagem *connect* que pode ser transmitida como UDP no caso da transmissão ter sido iniciada em modo *fast connect.*

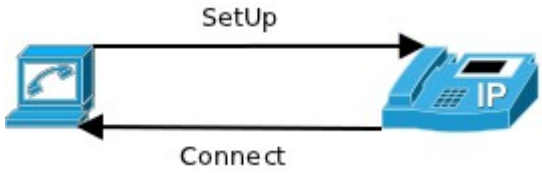

Figura 2: Call flow

A Figura 2 exemplifica o caso mais simples de *call signaling flow* de uma comunicação H.323. Neste caso, 2 *enpoints* comunicam diretamente um com o outro, sem utilização de um *gatekeeper*, e recorrendo a capacidade de *fast connect* existente no protocolo podendo assim estabelecer uma comunicação com o mínimo de mensagens necessárias. Neste caso, o enpoint1, com uma mensagem de *setup*, sinaliza ao endpoint2 que quer estabelecer uma comunicação, ao que este responde com um sinal de *connect* que sinaliza que a comunicação foi aceite e pode proceder. Quando o *fast connect* é utilizado, a mensagem de negociação de média em H.245 necessária para estabelecer a chamada é encapsulada na mensagens de *setup* e *connect* H.225, transmitida pelo mesmo canal. Este procedimento evita o estabelecimento de mais um canal e também a troca de mais mensagens. O passo seguinte é a transmissão de média por canal RTP entre os *endpoints* até que a chamada seja terminada. No final da chamada uma mensagem *ReleaseComplete* será enviada pelo *endpoint* que termina a chamada ao outro.

3.3.2. Sinalização H.225 RAS - mensagens e *callflow*

Media Packetization (empacotamento de media) – opcionalmente o protocolo prevê a possibilidade de utilizar o protocolo H.225 de modo a encapsular mensagens do protocolo H.245 para a negociação dos *streams* de média. Este modo, nomeado *fast connect,* permite fazer a negociação de média, sem ter de criar um canal próprio para o protocolo H.245 poupando algumas mensagens nessa negociação.

RAS *signaling* [7] (Sinalização RAS) - A sinalização *Request Admition* Status é utilizada para a comunicação de um endpoint com um *gatekeeper* ou entre dois *gatekeepers.* É responsável pelo registo, admissão, status, cancelamento dos dispositivos, assim como pelo controlo de largura de banda. Esta comunicação é feita por um canal separado (RAS *channel).*

Para este efeito o protocolo dispõe de uma variedade de mensagens UDP para estabelecer e controlar registos e comunicações. Estas são:

Gatekeeper ReQuest (GRQ) - pedido de um dispositivo *endpoint*, *gateway,* MCU, *gatekeeper*, que alerta aos *gatekeepers* para a sua presença e que requer a estes que lhe seja concedida a possibilidade de registo em cada um destes. Esta mensagem será respondida por estes por uma mensagem de (GRJ) ou de (GCF).

Gatekeeper ReJect (GRJ) - Resposta negativa ao pedido (GRQ), indica ao dispositivo emissor do (GRQ) que o *gatekeeper* que respondeu existe, mas não pode aceitar o seu registo, informação relevante, e a razão pela qual o pedido foi rejeitado também faz parte desta mensagem.

Gatekeeper ConFirm (GCF) - Resposta positiva ao pedido de (GRQ), informa o dispositivo que emitiu o (GRQ) que este *gatekeeper* existe e pode aceitar o seu registo.

Registration ReQuest (RRQ) - Pedido de um dispositivo *endpoint, gateway,* MCU, *gatekeeper,* para um *gatekeeper* para efetuar o seu registo.

Registration ReJect (RRJ) - Resposta negativa a um pedido (RRQ), indica ao dispositivo que requereu o pedido de registo, que terá de registar-se noutro *gatekeeper.* Esta mensagem também informar o dispositivo do motivo pelo qual o seu pedido foi rejeitado.

Registration ConFirm (RCF) - Resposta positiva a um pedido de registo (RRQ). A partir deste momento o dispositivo que se registou efetuará as suas comunicações através do *gatekeeper* em que se registou.

Unregister ReQuest (URQ) - Mensagem de cancelamento de registo, informa um dispositivo registado a um *gatekeeper* ou um *gatekeeper* onde um dispositivo esta registado que a sua associação é cancelada.

Unregister ReJect (URJ) - Resposta informativa que informa ao dispositivo, *gatekeeper* emissor de um pedido de (URQ), que este não se encontrava associado a este dispositivo *gatekeeper*, razão pelo qual o cancelamento da associação não pode ser efetuado.

Unregister ConFirm (UCF) - Resposta de confirmação que informa um dispositivo que efetuou um pedido (URQ),requerendo o cancelamento de uma associação *enpoint-gatekeeper* ou *gatekeeper-gatekeeper*, que a associação foi cancelada.

Admission ReQuest (ARQ) - Pedido por parte de um dispositivo *endpoint* de acesso a rede de comutação por pacotes, este pedido que é endereçado ao *gatekeeper* em que ele esta associado.

Admission ReJect (ARJ) - resposta negativa de um *gatekeeper* para um dispositivo endpoint a um pedido acesso a rede (ARQ), submetido por este *endpoint.* Esta mensagem também informa o endpoint do motivo pelo qual o pedido foi negado.

Admission ConFirm (ACF) - resposta positiva do *gatekeeper* que submeteu o (ARQ), permitindo a este que este possa comunicar com outro dispositivo.

Bandwidth ReQuest (BRQ) - Pedido de mudança de valor de largura de banda por parte de um *endpoint* para um *gatekeeper*, este pedido pode requerer uma aumento ou uma diminuição.

Bandwidth Reject (BRJ) - resposta negativa do *gatekeeper* ao *endpoint* que submeteu o pedido de mudança de largura de banda (BRQ). O motivo pode ser devido a falta de recursos disponíveis por parte do *gatekeeper*, no caso de pedido de aumento de largura de banda ou devido a falta de suporte para a largura de banda requerida por parte do *gatekeeper* no caso de um pedido de diminuição.

Bandwidth ConFirm (BCF) - Resposta positiva ao pedido de mudança de largura de banda.

Disengage ReQuest (DRQ) - Mensagem envida de *enpoint* para *gatekeeper* ou de *gatekeeper* para *endpoint*, esta mensagem indica ao *gatekeeper* que o *endpoint* vai desligar a chamada, ou informa o *endpoint* que este tem de terminar a chamada.

Disengage ReJect (DRJ) - Resposta negativa a uma mensagem (DRQ), de *gatekeeper* para *endpoint*, que indica ao requerente que este não esta registado no *gatekeeper.*

Disengage ConFirm (DCF) - Resposta positiva a um pedido de (DRQ) esta indica *gatekeeper* que a pedido de (DRQ) foi recebido.

Location ReQuest (LRQ) - Pedido de resolução de endereço de um *gatekeeper* para outro, esta mensagem indica ao *gatekeeper* inquirido que o requerente procura o endereço RAS especifico de um *endpoint* do qual só conhece o alias.

Location ReJect (LRJ) - Resposta negativa a uma mensagem de resolução de endereço (LRQ), esta resposta indica que o *enpoint* específico requerido pelo gatekeeper é desconhecido ao gatekeeper inquirido.

Location ConFirm (LCF) - Resposta positiva a um pedido de (LRQ), indica ao gatekeeper requerente que o *endpoint* está registado neste *gatekeeper* e o seu endereço RAS.

Information ReQuest (IRQ) - mensagens de *information request* são enviadas de gatekeepers para enpoints, requerendo algum tipo de informação de estado relacionada com alguma comunicação entre esse endpoint e outro.

Information Request Response (IRR) - mensagem de resposta a um pedido de (IRQ), é enviada de um *endpoint* para um *gatekeeper*, contém as informações solicitadas na mensagem de (IRQ). Estas mensagens podem ser enviadas automaticamente sem ocorrência de um (IRQ), no caso da mensagem de (ACF) que foi recebida para confirmação do uso da rede solicitar o envio automático destas mensagens.

Information ACKnowledgement (IACK) - mensagem de resposta do *gatekeper* para o *endpoint* que o informa que a mensagem de (IRR) foi recebida. Esta mensagem é enviada quando o *endpoint* especificou na mensagem de IRR que precisava de confirmação da receção.

Request In Progress (RIP) - mensagem de resposta a um pedido que ainda não pode ser respondido. Esta mensagem informa quem solicitou o pedido, que este ainda não pode ser respondido, assim como o tempo de espera que o dispositivo deve aguardar antes de voltar a repetir o pedido.

Resource Availability Indication (RAI) - mensagem de uma *gateway* a um *gatekeeper* que o informa das suas capacidades e suporte a cada um dos protocolos H.xxx.

Resource Availability Confirm (RAC) - mensagem de resposta a uma mensagem (RAI) do *gatekeeper*, que informa o *gateway* que o *gatekeeper* já possui a informação de relativa a este.

Service Control Information (SCI) - mensagem enviada de um *gatekeeper* para um *endpoint* . Serve para informar um utilizador que a rede aceita certos tipos de serviço. No

caso de ser enviada de um *enpoint* para um *gatekeeper*, visa transmitir a rede informação relevante para o processamento da chamada.

Service Control Response (SCR) - mensagem que sinaliza a correta receção de um (SCI).

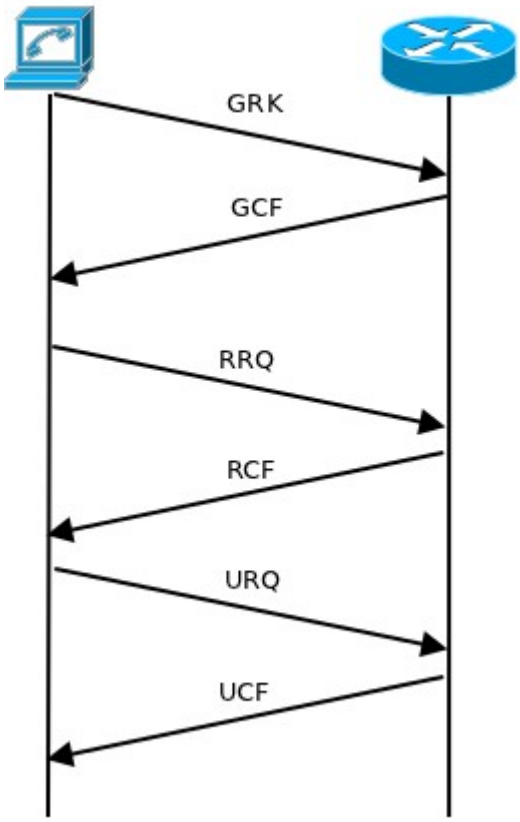

Figura 3: Sequência típica de mensagens num sistema H.323.

O processo de descoberta dos *gatekeepers* de modo a estes estarem informados da presença uns dos outros e de registo dos *endpoints* na zona do *gatekeeper* é realizado com uma troca de mensagens entre os *endpoints* e *gatekeepers. A* ilustração da Figura 3 descreve uma típica sequência de mensagens num sistema H.323 em que este processo é concluído. A descrição deste processo pode ser comentada da seguinte forma:

1º - Quando um *endpoint* é ligado, salvo configuração em contrário, emitira um sinal (GRQ)(*gatekeeper request),* alertando os *gatekeepers* para a sua presença. Este sinal será recebido pelos *gatekeepers* presentes que lhe responderão para sinalizar a sua presença e também para transmitir a informação se aceitam o seu registo.

2º- Se o registo do *enpoint* for possível, os *gatekeepers* responderão com uma mensagem de (GCF)(*gatekeeper confirm*); ou, se for impossível, com uma mensagem de *(GRJ)(gatekeeper reject*).

3º – O *endpoint* emitirá uma mensagem de (RRQ) (*register request),* efetuando um pedido de registo num dos *gatekeepers* que lhe responderam aceitar o seu registo.

4º – O *gatekeeper* confirmará o registo do *endpoint* enviando-lhe uma mensagem (RCF)(*register confirm*), ou negará o registo do *endpoint* emitindo uma mensagem (RRJ) (*Register reject*).

As primeiras duas mensagens não são obrigatórias visto que os *enpoints* podem ser configurados para se registarem num *gatekeeper* especifico. Nesse caso, assim que for ligado, o *endpoint* iniciará o processo de registo a partir da mensagem 3.

O e cancelamento de um registo por parte de um *endpoint* numa zona é ilustrado também na figura() corresponde a:

5º – O endpoint assinala a sua intenção de cancelamento de registo no *gatekeeper*, emitindo a este uma mensagem de (URQ)(*unregister request)*

6º – O *gatekeeper* responderá ao *enpoint* com uma confirmação de cancelamento de registo (UCF)(*unregister confirm*) ou informando-o que ele não se encontra registado, emitindo um (URJ)(*unregister reject*).

O controlo das chamadas e da largura de banda na zona dos *gatekeepers* é totalmente governado por estes e são sempre inquiridos pelos *endpoints*, quando estes pretendem efetuar uma chamada ou uma mudança da largura de banda disponível. No caso de um endpoint2 receber um pedido de comunicação ou de mudança de largura de banda de outro endpoint1, terá obrigatoriamente de inquirir ao *gatekeeper* se este autoriza esta mudança. Mesmo que já esteja autorizada ao endpoint1. Em comunicações entre *endpoints* que pertencem a zonas diferentes, a largura de banda terá obrigatoriamente de ser negociada entre estes.

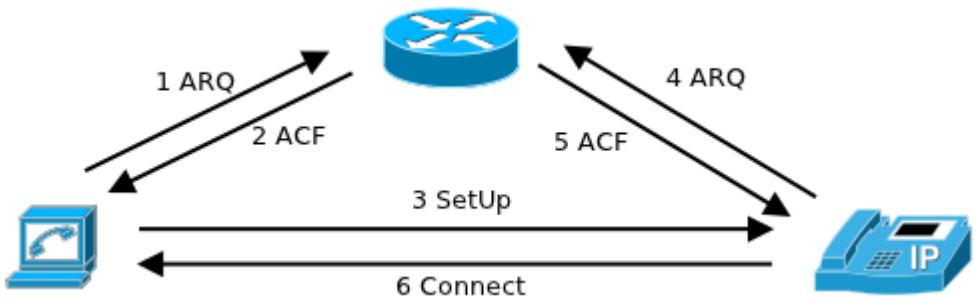

Figura 4: Comunicação entre dois endpoints na zona de influencia de uma gatekeeper.

A negociação de comunicações entre dois *endpoints* na zona de influencia de uma *gatekeeper,* ilustrada na Figura 4, pode ser descrita como:

1º mensagem de (*arq*) o enpoint1 pede permissão ao *gatekeeper* para utilizar a rede de modo a entrar em comunicação com outro endpoint2.

2º mensagem de (*acf*) que autoriza o endpoint1 a utilizar a rede,ou mensagem de (arj) detalhando o motivo pelo qual o acesso foi negado

3º mensagem de (*setup*) é enviada do endpoint1 para o *endpoint* 2 declarando a sua intenção de iniciar uma chamada.

4º mensagem de (*arq*) do endpoint2 para o *gatekeeper* pedindo autorização para responder a chamada.

5º mensagem de (*acf*) do *gatekeeper* para o endpoint2 autorizando-o a responder a chamada, ou mensagem de (arj) rejeitando o pedido.

6º mensagem de (*connect*) do endpoint2 para o endpoint1 autorizando a chamada.

3.3.3. Sinalização H.245 mensagens e *callflow*

O protocolo H.245 [13] descreve a negociação de média numa chamada H.323, para isso recorre a duas negociações distintas entre os *endpoints*. A *Master-Slave negociation* e a *capability negociation*. Para a realização das suas funções. O protocolo dispõe das seguintes mensagens TCP:

Master-Slave Determination (MSD) - Inicia a negociação do estatuto dos *endpoint*. As respostas a esta mensagem podem ser *Acknowledge* em caso de confirmação do estatuto , *Reject* no caso de a negociação não ter resultado, *Release* no caso de *timeout*. Em caso de MSD - *Reject* a negociação do estatuto será reiniciada.

Terminal Capability Set (TCS) - Envia informações acerca das capacidades de média do *endpoint*. As respostas a esta mensagem podem ser *Acknowledge* em caso de confirmação das configurações de média, *Reject* no caso do *endpoint* não permitir estas configurações e *Release* em caso de *timeout*.

Open Logical Channel (OLC) - Pedido de abertura de um canal para transmissão de média entre os *endpoints*, as respostas a esta mensagem podem ser *Acknowledge* em caso de confirmação, *Reject* no caso das configurações do canal excederem a largura de banda permitida ou se o *master endpoint* necessitar de uma configuração simétrica das configurações de média e se as configurações propostas pelo slave não o forem.

Close Logical Channel (CLC) – Pedido de cancelamento de um canal entre dois *endpoints*. A resposta a este pedido será *Acknowledge*.

Request Mode (RM) – Pedido de transmissão num modo especifico, o modo pode ser *audio*, *video*, data ou *encriptation*. As possíveis respostas a esta mensagem podem ser *Acknowledge* em caso de confirmação, *Reject* em caso do *endpoint* não poder obedecer ao pedido

Send Terminal Capability Set – Pedido de envio de uma ou varias mensagens (STC) de modo a poder negociar as configurações de media. A resposta a este pedido é uma mensagem (STC).

End Session Command – indicação de terminação das mensagens no protocolo H.245.

A primeira negociação de *Master-Slave* [13] é importante porque autoriza um dos *endpoints* a resolver todos os conflitos de negociação que possam surgir. Esta negociação é realizada pela comparação de dois valores dos *endpoints*. O *terminaltype* intrínseco a cada tipo de dispositivo existem quatro MCU, *gateway*, *gatekeeper*, *enpoint*. Em caso de serem dispositivos do mesmo tipo o *StatusDeterminationNumber* que é um valor aleatório será comparado. Estes dois valores estão presentes nas mensagens de negociação *Master-Slave* (MSD) que serão trocadas pelos *endpoints*. A outra negociação realizada é a troca de mensagens (TCS) entre os dois *endpoints* onde estes declaram as configurações de média que cada um pode aceitar. Depois das negociações terem sido concluídas é iniciada abertura dos canais de ambos os *endpoints* com uma mensagem (OLC). A receção dos dois *endpoints* de uma mensagem (*OLC ack*) inicia a abertura do canal RTP por onde os dados de média serão encaminhados.

4 Módulo H.323 para Videoconferência

4.1. Biblioteca GTK+

4.1.1. Dependências

A biblioteca Gtk+ 2.24.2 utilizada nesta dissertação tem como requisitos para a sua instalação a dependência das bibliotecas, *glib*, *pango*, *atk*, *gdkpixbuf*, *gdk*. Esta biblioteca é uma ferramentas de implementação de interfaces gráficos para plataformas Linux e Windows.

4.1.2. Descrição

A sua inicialização é implementada na função *main* inicial usando a função *gtk_init()*. Seguidamente todos os *widgets* que serão implementados numa função tem de ser declarados e guardados. Os *widgets* incluem as janelas, *frames*, botões, caixas de texto, diálogos dos nossos interfaces. Vários exemplos da declaração dos *widgets* são:

```
GtkWidget *window;            //no caso de uma janela
GtkWidget *container_v;     //no caso de um container vertical
GtkWidget *container_h;   //no caso de um container horizontal
GtkWidget *button;               //no caso de um botão
GtkWidget *entry;                //no caso de uma caixa de entrada de texto
```

A implementação dos widget é executada com diferentes funções da biblioteca, dependendo da função de cada widget.

```
window=gtk_window_new(GTK_WINDOW_TOP_LEVEL);
container_v= gtk_vbox_new(TRUE,2);
container_h= gtk_hbox_new(TRUE,2);
container_h2= gtk_hbox_new(TRUE,2);
button=gtk_button_new_with_label("press me");
entry=gtk_entry_new();
```

Após a implementação dos widgets, criam-se as ligações dos *widgets* que necessitam de sinalizar as funções de *callback* executadas no momento da sua activação.

```
g_signal_connect(window,"delete-event",
G_CALLBACK(delete_event),NULL)

g_signal_connect(window,"destroy",
```

```
G_CALLBACK(destroy),NULL)

g_signal_connect(button,"clicked",
G_CALLBACK(function_button),NULL);

g_signal_connect(entry,"activate",
G_CALLBACK(entry_activated),entry_text);
```

No caso da implementação de uma função em intervalos de tempo definidos.

```
g_timeout_add(100,(GSourceFunc)timed_func,NULL);
```

Seguidamente, empacotamos os widgets de forma a serem apresentados no interface gráfico.

```
gtk_box_pack_start(
GTK_BOX(container_h),entry,TRUE,TRUE,10);

gtk_box_pack_start(
GTK_BOX(container_h2),button,TRUE,TRUE,10);

gtk_box_pack_start(
GTK_BOX(container_v),container_h,TRUE,TRUE,10);

gtk_box_pack_start(
GTK_BOX(container_v),container_h2,TRUE,TRUE,10);

gtk_container_add(GTK_CONTAINER(window),container_v);

Sucessivamente apresentamos os widgets.

gtk_widget_show(entry);
gtk_widget_show(button);
gtk_widget_show(container_h);


gtk_widget_show(container_h2);
gtk_widget_show(container_v);
gtk_widget_show(window);
```

Por fim chamamos a função *gtk_main* que executará o nosso interface gráfico.

```
gtk_main();
```

As funções de *callback* serão implementadas de modo estático e antes da função *main*.

```
static gboolean delete_event(Gtkwidget *widget, gpointer data)
{
  gtk_main_quit();
}

static gboolean destroy(Gtkwidget *widget, gpointer data)
{
  gtk_main_quit();
}

static void function_button(Gtkwidget *widget, gpointer data)
{
  //codigo a executar dentro da função callback
}

static void timed_func(Gtkwidget *widget, gpointer data)
{
  //código a executar dentro da função callback
}
```

No caso de uma função com parâmetros de entrada.

```
static void entry(Gtkwidget  *widget, Gtkwidget *entry_text, gpointer data)
{
  //código a executar dentro da função callback
}
```

4.2. Biblioteca libopal e libpt

4.2.1. Dependências

A biblioteca Libopal 3.6.8 utilizada nesta dissertação tem como requisitos para a sua instalação dependências de outras bibliotecas, nomeadamente a biblioteca gcc 3.22, a biblioteca *bison parser generator YACC compatible*, a biblioteca *flex lexical Analyser Generator* e a biblioteca *libpt* 2.6.7. Esta biblioteca tem como objetivo integrar diferentes protocolos de telefonia num só modelo, de modo a possibilitar a implementação de aplicações que permitem estabelecer chamadas entre vários dispositivos numa rede e de modo transparente para os utilizadores. Permite a comunicação através de dispositivos e protocolos distintos. O modelo utilizado para estas aplicações é um modelo em camadas que pode ser representado de um modo aproximado pela Figura 5.

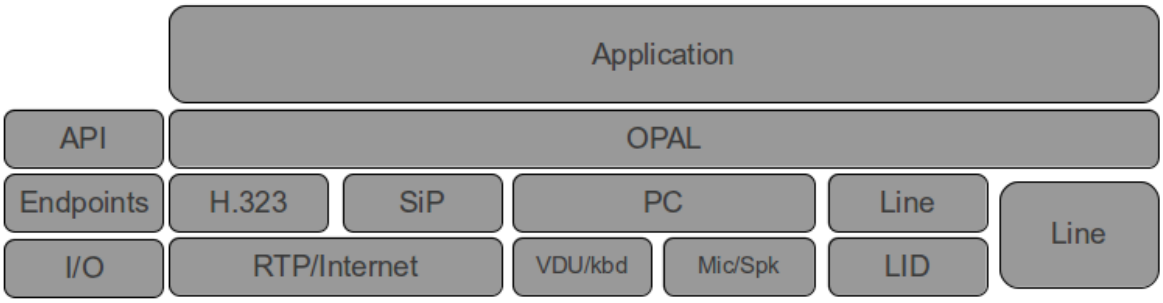

Figura 5: Modelo de aplicações com API OPAL.

A biblioteca *libopal* divide os protocolos em várias entidades separadas que representam diferentes aspetos do modelo. Estas entidades são:

Endereço - é definido como 2 strings separadas por ":" sendo a primeira um identificador único que referencia o tipo de endereço utilizado, dependendo do protocolo ou tipos de *endpoint* utilizados, a segunda um identificador que refere o endereço em si num formato de acordo com as regras definidas pelo primeiro identificador. Por exemplo (h323:192.168.1.93)

Endpoints - personificam/representam os componentes permanentes do modelo de abstração telefónica da biblioteca. Existem dois tipos principais de *endpoints:* e*ndpoints* terminais e *endpoints* de rede. Os terminais são usualmente *endpoints* que representam o estágio final da comunicação, por exemplo uma porta POTS ligada a um telefone. O segundo grande tipo é o *endpoint* de rede, este tipo de *endpoint* liga-se através de uma ligação de rede, tipicamente uma LAN *network* ou ligação de Internet, mas não necessariamente. Um exemplo deste tipo de *endpoint* é o *endpoint* H323. Cada *endpoint* é configurável e tem um conjunto de atributos que controlam o seu comportamento por defeito. Por exemplo, um *endpoint* pode ser terminal ou de rede, pode permitir uma ou mais ligações, pode receber ou também efetuar chamadas.

Ligações - representam os componentes mais transitórios do modelo. Inclui toda a informação de ligação e de estado de modo a manter a ligação ativa. Por exemplo, numa chamada H.323 a ligação H.323 contém sockets que sustentam a sinalização e controlam os canais entre o local e o remoto.

Chamada - é a entidade que liga uma ou mais ligações. Tipicamente são criadas pela aplicação em resposta as *callbacks* que recebem *incoming calls*. A aplicação pode proceder ao encaminhamento da chamada do modo que pretender desde que disponíveis e cria uma segunda ligação utilizando o protocolo disponível preferido. É

então estabelecida a chamada que abrange as varias ligações. O tempo de vida da chamada é controlado pela aplicação e é tipicamente existente enquanto pelo menos uma ligação existente está agregada a ela.

Mediastream - Incorpora a *source* e *sink* dos dados num formato de media particular, áudio ou vídeo. Cada protocolo usa um *mediastream* particular tornando as características de cada um bastante heterogéneas. O tempo de vida dos *mediastream* é bastante variável e dependente do protocolo utilizado. Por exemplo numa chamada H.323 o *mediastream* é criado depois da chamada iniciada e é interrompida após pedido do utilizador no momento da conclusão da chamada. Contrastando com este, no protocolo LID o *mediastream* é criado logo após a ligação e é valido enquanto essa chamada existir.

Uma típica aplicação de comunicação que utiliza a biblioteca libopal criará configurações iniciais de acordo com as suas necessidades e capacidades e criará uma instância da classe *OpalManager*. Deste modo poderá administrar e controlar as *callbacks* da biblioteca disponíveis para esse efeito. permitindo aos utilizadores o estabelecimento, receção e controlo de chamadas áudio ou áudio/vídeo. Também são necessários os *endpoints* para o seu funcionamento: no nosso caso, os enpoints de *PCSoundSystem* que serão criados, administrados pela classe *OpalPCSSEndpoint*, estes servirão para o funcionamento dos *transcoders* da biblioteca de modo a utilizar os sistema de som do computador; além disso, os sinais de alerta de receção de chamada e de alerta de chamada a tocar no utilizador remoto também são controlados por esta classe. Também será criado um *endpointH323* que além de facultar o meio para estabelecer uma chamada também será responsável pelo *listener* que permite a receção de chamadas. Um *endpoint* de *source* também será criado e representará o *endpoint* terminal do utilizador local.

4.3. Implementação

A execução da aplicação simpH323 inicia a função a função *main*, esta lança a biblioteca *gtk*, prepara os *handlers* de sinais e verifica os argumentos passados por linha de comandos. Seguidamente é instanciado um processo da classe *SimpProcess* descendida da classe *pProcess*. Esta instância *pInstance* é então inicializada e nela decorrerão todos os processos e *threads* das bibliotecas *libpt* e *libopal* que compõem esta aplicação. No momento do retorno desta instância o valor de terminação é registado. A instância é destruída e o valor de terminação recebido da instância é retornado

finalizando assim a aplicação. A dinâmica desta função é representada no diagrama de fluxo da Figura 7.

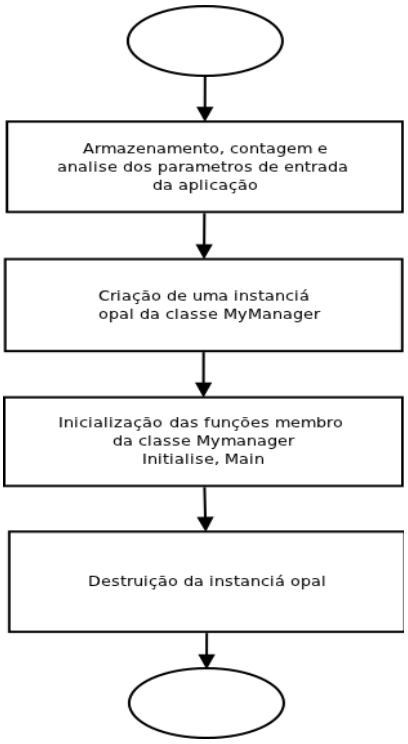

Figura 6

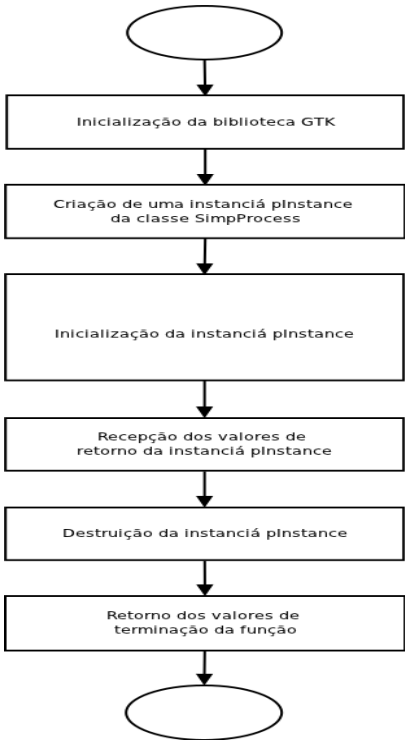

Figura 7

O processo *SimpProcess* é derivado da classe *pProcess* e é iniciado por uma função da biblioteca *gtk* de cumprimento mas implementada no intuito de a modificar de modo a poder receber uma função de carregamento de valores *default* num modo de inicialização rápida. Sucessivamente os argumentos passados na chamada do processo são adquiridos e registados e contabilizados, seguidamente estes parâmetros são analisados e separados. Posteriormente é criada uma classe *Opal*, de base *MyManager* classe derivada da classe *OpalManager* que é a classe central para a administração de chamadas na biblioteca *libopal*. Sucessivamente são chamadas duas funções membros da classe *MyManager*, a função *Initialise* que vai configurar *endpoints*, canais, *codecs*, dispositivos de captura áudio vídeo e a função *Main* que vai gerir as chamadas que o utilizador quer efetuar ou receber. Por ultimo a classe *Opal* é destruída e o processo *SimpProcess* conclui. Este comportamento esta representado no diagrama de fluxo da Figura 6.

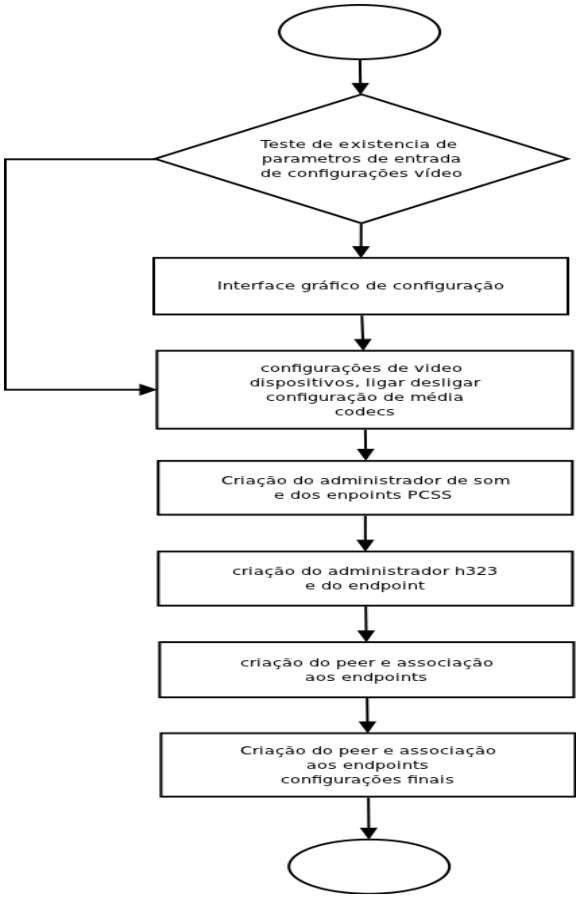

Figura 8

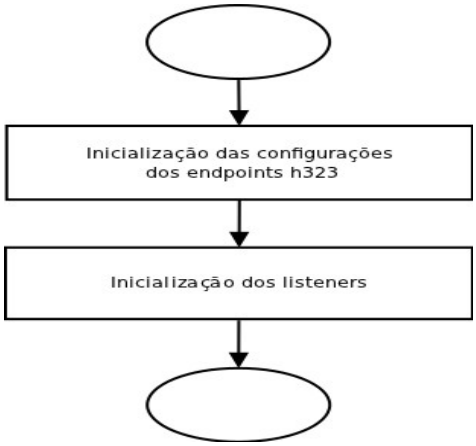

Figura 9

A função *Initialise* membro da classe *SimpProcess*. Tem como objetivo efetuar as configurações necessárias para o lançamento da função *Main* de modo a poder desempenhar as suas funções de administração de chamadas. Inicialmente esta é constituída por um *widget* de controlo de modo a escolher o dispositivo de captura vídeo entre as opções disponíveis, seguidamente de acordo com a seleção do utilizador ativa ou desactiva a presença de vídeo durante as chamadas. Posteriormente são configurados os dispositivos de áudio e vídeo assim como os *codecs* e formatos de média que vão ser utilizados na chamada. Após estas configurações iniciais são criados os *enpoints* de áudio e vídeo que são associados aos respetivos dispositivos de reprodução e captura áudio e vídeo. Seguidamente, é implementado o administrador de H.323 que gere e cria o endpointh323 e chama a função InitialiseH323EP, representada na Figura 9, que configura o *enpoint* e lança o *listener*, função que alerta o utilizador se receber chamadas. Finalmente é criado o *peer* que além de configurar as opções de encaminhamento, associa ao *endpoint* terminal os formatos de media que serão

utilizados. Seguidamente a função *Initialise* retorna. Terminando a sua execução. O diagrama da Figura 8 representa este comportamento.

A ativação ou desativação da captura de vídeo é executada pela função *SetAutoStartTransmitVideo*, a receção do vídeo do *remote user* é ativada ou desactivada pela função *SetAutoStartReceiveVideo*, ambas estas funções são membros da classe *OpalManager*. A seleção do dispositivo de captura de vídeo é feita usando a *funçãoSetVideoInputDevice*, função membro da classe *OpalManager*. As opções de *codecs* e formatos eliminados da utilização são filtrados utilizando a função *SetMediaFormatMask*, função membro da classe *OpalManager*. Neste caso o *codec* "GSM-AMR" é descartado devido ao ao não funcionamento do mesmo na aplicação. O ordenamento dos media *codec* e formatos será executado pela função *SetMediaFormatOrder*, função membro da classe *OpalManager*. A seleção dos dispositivos de som é implementada lançando a função *SetSoundDevice*, função membro da classe *MyPCSSEndPoint*, classe derivada da classe *OpalPCSSEndPoint* que administra o *endpoint* de som criado.

Figura 10

A função *Main* ilustrada na Figura 10 tem como objetivo permitir ao utilizador estabelecer e receber chamadas. Na sua inicialização chama a função console instância da classe *PconsoleChannel* que abre um canal de input output de modo a utilizar a função para a receção e transmissão de dados entre o utilizador local e a função. Após esta definição inicial é criada a *thread guithread* onde será implementada a interface gráfica de controlo do utilizador, esta decorrerá num *thread* simultâneo de modo a podermos controlar a aplicação. Seguidamente é implementado um ciclo for infinito que repetira a aquisição de dados recebidos dos alertas do sistema ou comandos do utilizador. Estes comandos do utilizador são lançados por um *switch* que lançará as diferentes funções da classe *OpalManager* que permitem Controlar a aplicação. Um dos

ramos do *switch* permite a saída do ciclo for para um *label endsimp*, a chegada a este *label* termina a função *Main* retorna para o processo *SimpProcess* as funções lançadas dentro do case são:

- *AcceptIncomingConnection* função membro da classe *OpalPCSSEndPoint* que permite atender uma chamada.
- *RejectIncomingConnection* função membro da classe *OpalPCSSEndPoint* que permite rejeitar uma chamada.
- *Hangupcurrentcall* função membro de *MyManager* que permite desligar uma chamada ou rejeitar uma chamada. Representada na Figura 11.
- *StartCall* função membro de *MyManager* que permite estabelecer uma chamada. Representada na Figura 12.

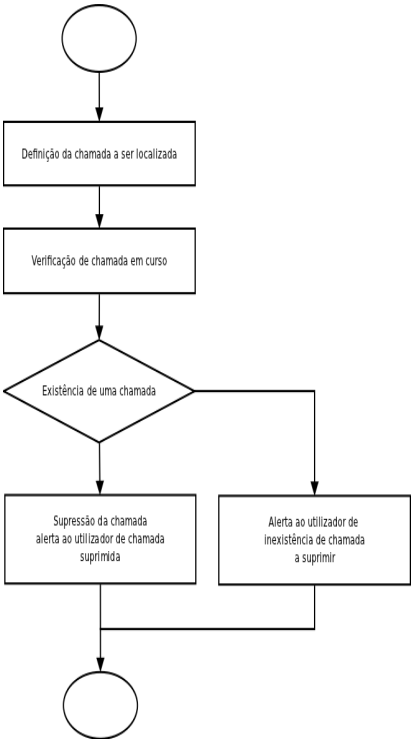

Figura 11

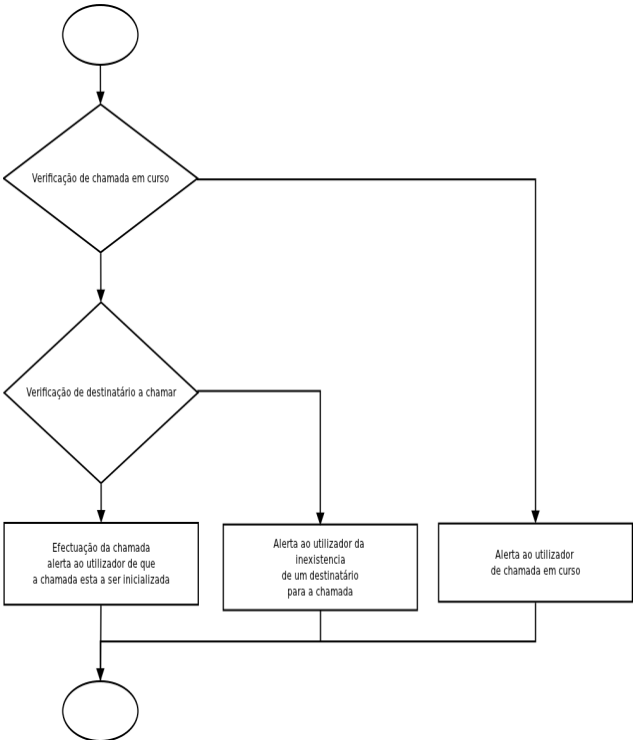

Figura 12

- A função *Hangupcurrentcall*, ilustrada na Figura 11, verifica se existe uma chamada para desligar ou rejeitar e em caso afirmativo chama a função *clear* da classe *OpalCall*. Em caso negativo alerta o utilizador de que não existe chamada para desligar/rejeitar.
- A função *StartCall*, ilustrada na Figura 12, verifica se existe um endereço introduzido pelo utilizador e em caso afirmativo lança a função *SetUpCall* membro da classe *OpalManager*. Esta criará as ligações necessárias de modo a

estabelecer a chamada. Por fim alerta o utilizador de que esta uma chamada em curso ou de que este não inseriu um destinatário.

- A função *OnEstablishedCall* alerta o utilizador de que está em conversação.
- A função *OnClearedCall* alerta o utilizador do conclusão de uma chamada e também do motivo da sua conclusão. Está representada na Figura 13.
- A função *OnOpenMediaStream* indica a classe *MyManager* se a chamada da função *OnOpenMediaStream* da classe *OpalManager* foi bem sucedida. Esta ultima função lança 3 funções de modo a estabelecer o *stream* dos diferentes media. São estas *FindMatchingCodecs*, *CreateMediaStream* e *CreateMediaPatch*.

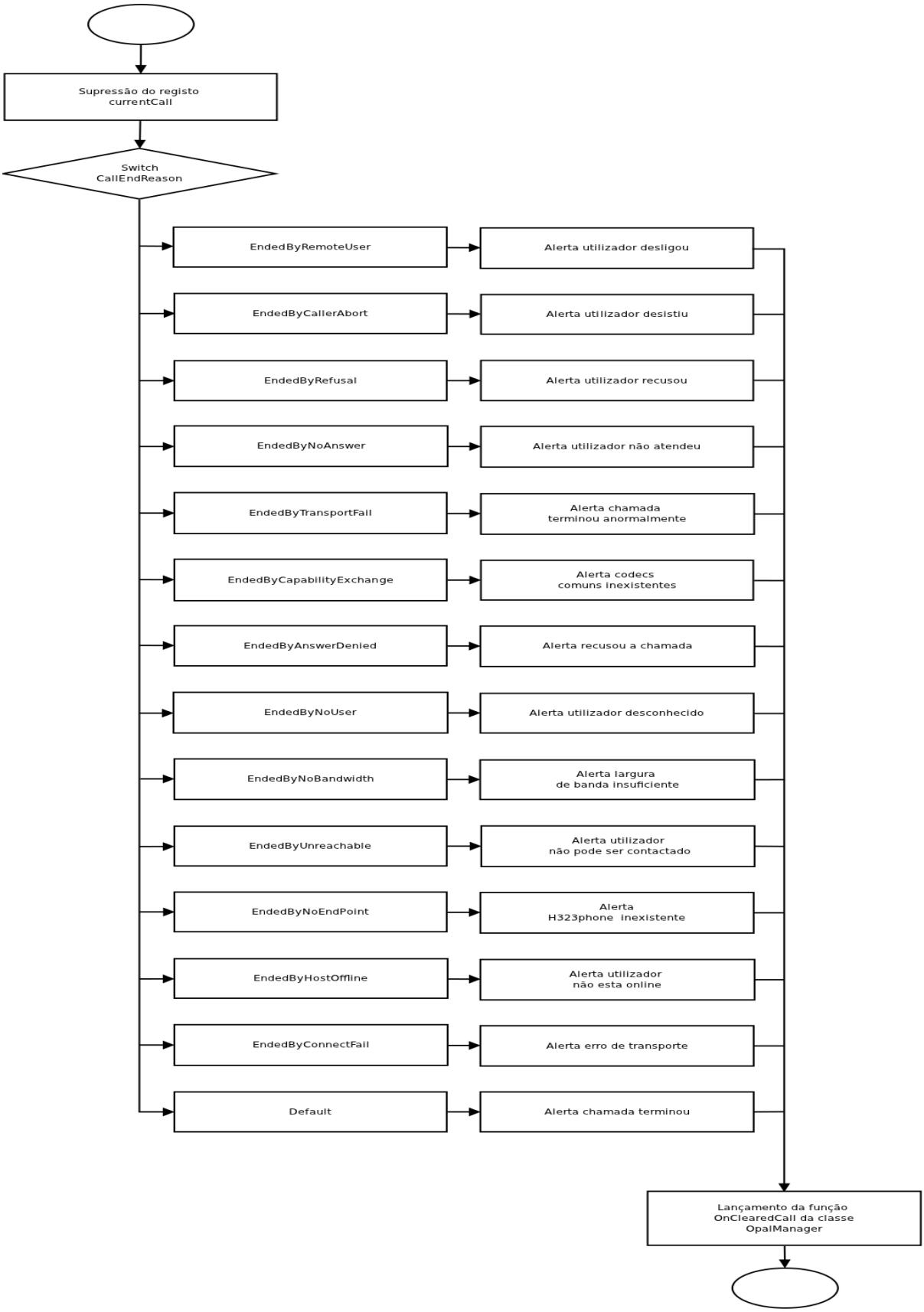
Supressão do registo
currentCall
Switch
CallEndReason
EndedByRemoteUser
Alerta utilizador desligou
EndedByCallerAbort
Alerta utilizador desistiu
EndedByRefusal
Alerta utilizador recusou
EndedByNoAnswer
Alerta utilizador não atendeu
EndedByTransportFail
Alerta chamada
terminou anormalmente
EndedByCapabilityExchange
Alerta codecs
comuns inexistentes
EndedByAnswerDenied
Alerta recusou a chamada
EndedByNoUser
Alerta utilizador desconhecido
EndedByNoBandwidth
Alerta largura
de banda insuficiente
EndedByUnreachable
Alerta utilizador
não pode ser contactado
EndedByNoEndPoint
Alerta
H323phone inexistente
EndedByHostOffline
Alerta utilizador
não esta online
EndedByConnectFail
Alerta erro de transporte
Default
Alerta chamada terminou
Lançamento da função
OnClearedCall da classe
OpalManager

Figura 13

5 Apresentação do Módulo Desenvolvido

5.1. Apresentação da aplicação implementada SimpH323

A aplicação resultante desta dissertação inicia-se apresentando uma janela de inicialização (Figura 14). Esta janela existe para posteriormente ser modificada de modo a poder escolher um modo de configuração automática que carregue os dados anteriormente salvos pelo utilizador, em vez de obrigar o utilizador a fazer a configuração inicial.

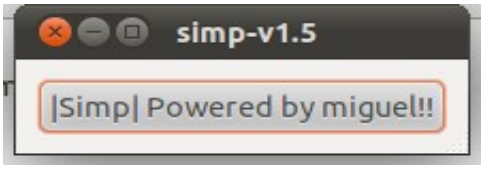

Figura 14: Janela de inicialização.

Seguidamente apresenta uma janela de configuração (ver Figura 15). Os dois primeiros botões quando selecionados permitem ao utilizador ligar ou desligar a transmissão vídeo e a receção vídeo. Deste modo, o utilizador pode iniciar uma chamada apenas como uma chamada telefónica ou utilizar apenas a transmissão de vídeo para um dos interlocutores. O 3º botão é um botão rotativo que permite escolher a câmara existente no sistema a escolher para utilização da aplicação. O 4º botão é o botão de confirmação de configuração e permite finalizar a configuração.

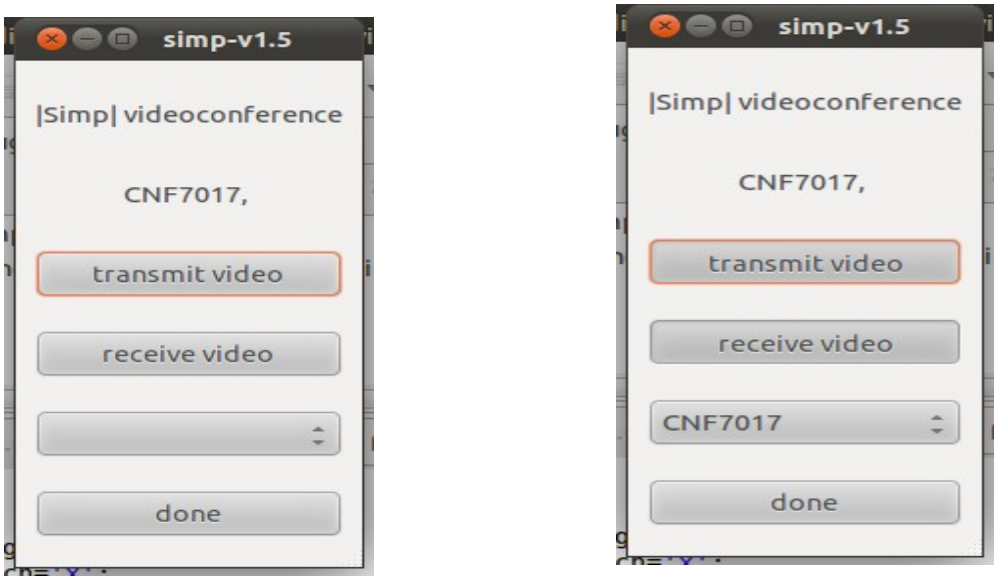

Figura 15: Janela de configuração (2 exemplos).

Depois de finalizada a configuração, a aplicação apresenta o cliente de comunicação. É constituído por um *display*, três botões, e uma caixa de entrada. O *display* permite ao utilizador saber informações sobre as chamadas que está a receber ou efetuar. Os botões permitem ao utilizador efetuar, receber e terminar chamadas. A caixa de entrada serve para introdução do endereço H323 do destinatário. Existem várias situações ilustradas: (Figura 16) quando a aplicação inicia, (Figura 17) quando efetua uma chamada e está à espera de ser atendida, (Figura 18) quando está numa chamada estabelecida, (Figura 19) quando o interlocutor remoto desligou a chamada e (Figura 20) quando o interlocutor local desligou uma chamada.

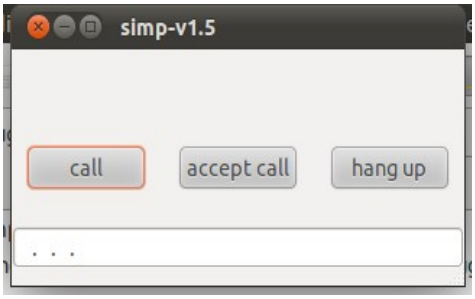

Figura 16 - Estabelecimento de chamada.

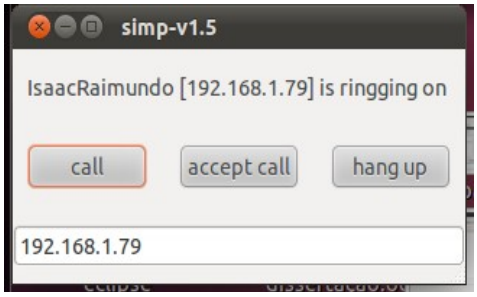

Figura 17 - Chamada a chegar.

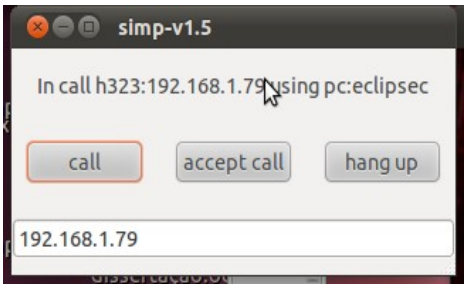

Figura 18 - Chamada em curso.

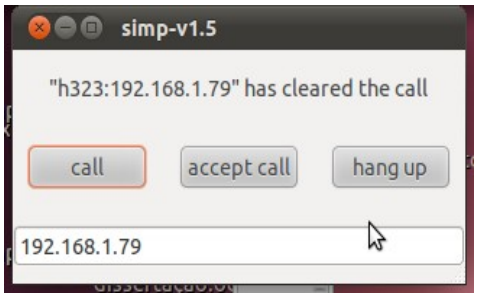

Figura 19 - Fim [remoto] de chamada.

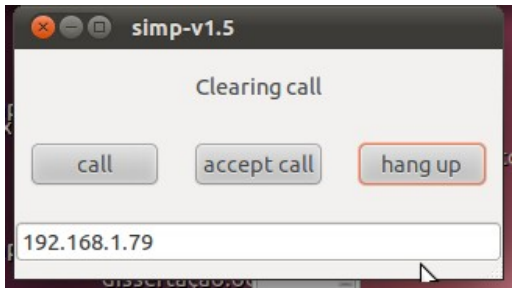

Figura 20 - Fim [local] de chamada.

A Figura 21 ilustra a janela vídeo do interlocutor remoto.

Figura 21: Chamada de vídeo em curso [interlocutor remoto].

5.2. Condições de realização de testes e resultados

Os resultados desta dissertação ficaram aquém das minhas expectativas: alguns dos meu objetivos foram concretizados mas outros ficaram para o atingir. Foi concretizada a implementação de um sistema de videoconferência de baixo custo utilizando computadores e redes NAT. A implementação de uma aplicação de videoconferência

utilizando bibliotecas e software *opensource* também foi concretizada. A aplicação desenvolvida permite a comunicação áudio, vídeo, áudio-vídeo utilizando o protocolo h323 e vários *codecs* de transmissão de áudio assim como *codecs* h261 para vídeo. Pela negativa, não consegui atingir as minhas expectativas de conseguir otimizar a aplicação de modo a possibilitar a comunicação. A aplicação implementada foi testada em vários cenários diferentes com sucesso:

- Utilização de duas tipologias de rede diferentes - 1 LAN, 1 WIFI
- Utilização da aplicação a correr em dois computadores diferentes em comunicação um com o outro.
- Utilização da aplicação para comunicar com outra máquina que opera sobre o mesmo sistema operativo (Ubuntu) mas com outra aplicação de comunicação h323 (Ekiga).
- Utilização da aplicação para comunicar com outra máquina que opera sobre outro sistema operativo (Windows7) e com outra aplicação de comunicação H.323 (Ekiga).
- Utilização da aplicação para comunicar com outra máquina que opera sobre outro sistema operativo (Mac OS Leopard) e com outra aplicação de comunicação h323 (xmeeting).

6 Conclusão

A videoconferência é hoje uma aplicação imprescindível das redes de telecomunicações, tendo evoluído de modo fascinante tanto em número de utilizadores como na forma como é utilizada, desde o conforto do lar até ao nosso meio de transporte. A implementação de sistemas com capacidade de videoconferência é implementada de várias formas e em várias plataformas e utilizando vários protocolos de comunicação. O conjunto de protocolos H.323 constituiu o objeto de implementação de um sistema de comunicação audiovisual utilizando uma plataforma comum e bibliotecas de código aberto. A aplicação desenvolvida foi implementada com sucesso e funcionou com outras aplicações comerciais semelhantes. Concluímos que o objetivo mínimo desta dissertação foi atingido mas diversas melhorias podem ser implementadas no sentido de criar uma aplicação mais flexível e abrangente.

6.1. Futuro e desenvolvimento

O futuro desta aplicação passa, como na maioria das aplicações comerciais ou académicas, por uma ampliação e melhoramento das suas características, propriedades e valências. Assim dos muitos aspetos a melhorar numa aplicação deste tipo, proponho as que em minha opinião se sobrepõem em valor e necessidade.

- Desenvolvimento de módulos adicionais para outros protocolos de comunicação que integram a aplicação (ex: SIP e XMPP).
- Aumento do número de chamadas simultâneas possíveis.
- Automatização da configuração e negociação de chamadas em videoconferência multi-ponto.
- Maior abrangência dos protocolos de transporte de áudio e vídeo de modo a melhorar a qualidade das comunicações.
- Melhoramento do interface gráfico, permitindo a possibilidade de introduzir mais opções de utilização.

A integração de algumas destas características permitiriam a esta aplicação a possibilidade de comercialização ou de uma utilização mais globalizada.

Referências

[1] Matthew Berry. (2011, Janeiro). Cisco Voice Over IP (CVOICE). CiscoVoiceGuru. Minneapolis [pdf file]. Disponível em: http://ciscovoiceguru.com/.

[2] Wanjiun Liao, “Mobile Internet telephony: mobile extensions to H.323”, apresentado no INFOCOM99, Nova Iorque, NY, 1999.

[3] A. Z. R. Langi, Y. Yonata, “A Performance and Interoperability Study of an OPENH323-bases Multimedia Over IP (MOIP) System”, apresentado no APCCAS2002, Bali, Indonésia, 2002.

[4] Chong Li, Weiyue Liu, Bo Zhong, Zhidi Jiang, “The Development of Codec Plug-in in Open H323”, apresentado no Multimedia Technology, Hangzhou, China, 2011.

[5] Vasco N. G. J. Soares, Paulo A. C. S. Neves, Joel J. P. C. Rodrigues, “Past, Present and Future of IP Telephony”, apresentado no CTRQ2008, Bucareste, Roménia, 2008.

[6] Minzhe Zhao, Mingshu Li, “T.120/H.32x-based Multimedia Conferencing Design”, apresentado no ICTM2012, NingBo, China, 2010.

[7] Series H. Audiovisual and Multimedia Systems – Infrastructures of audiovisual Services – Transmission multiplexing and Synchronization, ITU-T H.225.0, 05/2003.

[8] Series H. Audiovisual and Multimedia Systems – Coding of moving Video – Codec for Audiovisual Services, ITU, H.261, 11/1988.

[9] Series Q: Switching and Signalling – Digital Subscriber Signalling System no. 1 – Network Layer, ITU-T Q.931, 05/2003.

[10] Series H. Audiovisual and Multimedia Systems – Infrastructures of audiovisual Services – Systems and terminal equipment for audiovisual services, ITU-T H.323, 12/2009.

[11] Tabasi, Shekhar Prasad, Nabil Seddigh, Ioannis Lambadaris, “A Comparative Study of the SIP and IAX Protocols”, apresentado no CCECE/CCGEI2005, Saskatoon, Canada, 2005.

[12] Josef Glasmann, Wolfgang Kellerer, Harald Muller, “Service Development and Deployment in H.323 and SIP”, apresentado no ISCC2001, Hammamet, Tunísia, 2001.

[13] Series H. Audiovisual and Multimedia Systems – Infrastructures of audiovisual Services – Communication procedures, ITU-T H.245, 05/2011.

[14] Ge Zhang, Markus Hillenbrand, Paul Muller, “Facilitating the Interoperability among different VoIP Protocols with VoIP Web Services”, apresentado no DFMA’05, Besançon, França, 2005.

[15] Sajal K. Das, Enoch Lee, Kalyan Basu, Naveen Kakani, Sanjoy K. Sen, “Performance Optimization of VoIP Call over Wireless Links Using H.323 Protocol”, apresentado no INFOCOM2002, Nova Iorque, NY, 2002.

[16] Ilija Basicevic, Miroslav Popovic, Dragan Kukolj, “Comparison of SIP and H.323 Protocols”, apresentado no ICDT’08, Bucareste, Roménia, 2002.

[17] Si Haiyang, Dong Liyan, Liu Zhaojun, “Research on SIP and H.323 Protocol Conversion”, apresentado no ICICIC2008, Dalian, China, 2008.
[18] Gary A. Thom. (1996, dez). H.323: The Multimedia Communications Standard for Local Networks, [pdf]. Pág 52-56.
[19] Josef Glasmann, Wolfgang Kellerer, Harald Muller. (2003). Service Architectures in H.232 and SIP: a Comparison. IEEE Communications Surveys [pdf]. Volume 5, n. 2, pág 32-47.