Belmiro Antônio Kolakoski Júnior

Serviço Distribuído de Áudio e Videoconferência baseado em SIP e WebRTC para Aplicações Web

São José - SC

Março/2018

Belmiro Antônio Kolakoski Júnior

Serviço Distribuído de Áudio e Videoconferência baseado em SIP e WebRTC para Aplicações Web

Monografia apresentada à Coordenação do curso de tecnologia em Sistemas de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma tecnólogo em Telecomunicações.

Instituto Federal de Santa Catarina – IFSC

Campus São José

Sistemas de Telecomunicações

Orientador: Prof. Ederson Torresini, Me.

São José - SC

Março/2018

Belmiro Antônio Kolakoski Júnior
Serviço Distribuído de Áudio e Videoconferência baseado em SIP e WebRTC para Aplicações Web/ Belmiro Antônio Kolakoski Júnior. – São José - SC, Março/2018-
Orientador: Prof. Ederson Torresini, Me.

Monografia (Graduação) – Instituto Federal de Santa Catarina – IFSC
Campus São José
Sistemas de Telecomunicações, Março/2018.

1. OpenSIPS. 2. Contêiner. 3. Docker. I. Ederson Torresini. II. Instituto Federal de Santa Catarina. III. Campus São José. IV. Serviço Distribuído de Áudio e Videoconferência

Belmiro Antônio Kolakoski Júnior

Serviço Distribuído de Áudio e Videoconferência baseado em SIP e WebRTC para Aplicações Web

Monografia apresentada à Coordenação do curso de tecnologia em Sistemas de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma tecnólogo em Telecomunicações.

Trabalho aprovado. São José - SC, 15 de dezembro de 2017:

Prof. Ederson Torresini, Me.
Orientador

Prof. Eraldo Silveira e Silva, Dr. Eng.
IFSC

Humberto José de Sousa
Analista de TI

São José - SC
Março/2018

Dedico este trabalho a todos que de alguma forma contribuíram positivamente na minha jornada para conquista do diploma.

Agradecimentos

Ao Prof. Orientador Ederson Torresini, por acreditar que era possível a realização deste trabalho e por toda sua paciência, disponibilidade e contribuição na construção desta monografia.

À minha família, por todo suporte dado para o ingresso e finalização deste curso.

Aos professores, amigos e colegas de curso pela convivência e auxílio durante todo o curso.

Aos demais profissionais do campus pela disponibilidade e suporte aos alunos sempre que necessário.

Ao professor coordenador do curso Alexandre Moreira, por toda paciência e boa vontade ao ceder-me o tempo que foi preciso para o término desta monografia.

A altura das suas realizações será igual
à profundidade das suas convicções.
(William F. Scolavino)

Resumo

Este trabalho busca uma alternativa para o provimento de telefonia e videoconferência no câmpus São José do Instituto Federal de Santa Catarina (IFSC), utilizando-se de uma plataforma que lida com a comunicação via voz, texto e vídeo através do protocolo SIP e que possui suporte a WebRTC possibilitando servir de base para futuras aplicações web que poderão ser desenvolvidas para ambientes como moodle afim de disponibilizar áudio e/ou vídeo para o ensino a distância.

O serviço será desenvolvido para rodar no ambiente de aplicações em contêineres que está sendo implementado no câmpus, sua utilização dentro de contêineres possibilitará o melhor uso dos recursos e alta escalabilidade conforme demanda, além da maior facilidade nos testes e implementações futuras de novas funcionalidades a plataforma.

Palavras-chave: OpenSIPS. Contêiner. Áudio. Videoconferência.

Abstract

This work seeks an alternative for the provision of telephony and videoconferencing in the São José Campus of the Federal Institute of Santa Catarina (IFSC), using a platform that deals with communication via voice, text and video through the SIP protocol and that has support for WebRTC allowing to serve as a basis for future web applications that could be developed for environments such as moodle to provide audio and/or video for distance learning.

The service will be developed to run in the container application environment that is being implemented on the campus, its use inside containers will enable the best use of resources and high scalability on demand, as well as greater ease in testing and future implementation of new functionalities the platform.

Keywords: OpenSIPS. Container. Audio. Video conference.

Lista de ilustrações

Lista de abreviaturas e siglas

LDAP *Lightweight Directory Access Protocol* 41

RVSP *Resource Reservation Protocol* 23

RTSP *Real-Time Streaming Protocol* 23

SDP *Session Description Protocol* 23

UA *User Agent* 25

UAC *User Agent Client* 25

UAS *User Agent Server* 25

Sumário

1 Introdução

A telefonia ocupa um papel fundamental na gestão de instituições de todos os portes e setores de atuação, superando outros meios eletrônicos como o e-mail. Ela traz mais agilidade na comunicação e consequentemente acelera o andamento dos processos das instituições. A comunicação telefônica através da comutação de circuitos está se tornando obsoleta dando lugar a telefonia IP, a qual utiliza a comutação de pacotes e a infraestrutura já existente de rede de computadores.

O termo Telefonia IP é genérico e define as tecnologias de redes que utilizam o protocolo IP para trafegar dados e voz, sejam elas redes públicas (como a internet) ou redes privadas. Surgiu no mercado de telecomunicações em 1995, e desde então, os fabricantes vêm se esforçando para desenvolver novos equipamentos com preços mais acessíveis e com tamanho reduzido a fim de difundir a tecnologia (ROSS, 2007).

Nos serviços de telefonia convencional, a voz é transmitida através da Rede de Telefonia Pública Comutada (RTPC). Nos serviços de telefonia IP, a voz passa por um processo de digitalização para que este possa viajar pela rede na forma de bits. Uma vez digitalizada, a voz é transmitida na forma de pacotes de dados usando protocolo IP dentro de uma rede privativa ou rede onde pode haver garantia do serviço oferecido, isto é, reduzir quando possível atrasos que comprometam a qualidade da voz transmitida, como por exemplo uma *Virtual Private Network* (VPN).

A videoconferência vem junto com a telefonia se tornando essencial nas instituições pois permite a interação visual muitas vezes indispensável em reuniões, aulas entre outros. Os serviços de áudio e vídeo são dinâmicos demandando mais ou menos recursos dos servidores. Criou-se assim uma necessidade do provimento destes serviços da melhor maneira possível garantindo a qualidade do serviço, adaptabilidade e escalabilidade.

No Instituto Federal de Santa Catarina (IFSC) câmpus São José, em particular, o cenário atual é ainda mais favorável. A Coordenadoria de Tecnologia da Informação e Comunicação (CTIC) está implantando uma nova infraestrutura de servidores e serviços baseada nos trabalhos de conclusão de curso de Oliveira (2017) e Reis (2017) e, com isso, a telefonia IP pode acompanhar essa tendência utilizando melhor os recursos do câmpus.

1.1 Objetivo

O objetivo deste trabalho é oferecer ao IFSC câmpus São José uma proposta de implementação de telefonia IP escalável baseada em orquestração de serviços em contêineres, para que seja possível atender a uma grande carga de processamento de

chamadas telefônicas, incluindo áudio e videoconferência para Ensino a Distância (EaD), e com capacidade adaptável.

1.1.1 Objetivo secundário

Como objetivo secundário deste trabalho tem-se o suporte a WebRTC para os terminais telefônicos, uma vez que a solução de orquestração de serviços em contêineres é baseada em aplicações Web.

1.2 Organização do texto

O texto está organizado da seguinte forma: no Capítulo 2 é apresentado a fundamentação teórica dando uma visão ampla do protocolo SIP e do software livre escolhido para o desenvolvimento do trabalho, o OpenSIPS. No Capítulo 3 é apresentado o desenvolvimento do trabalho. No Capítulo 4 é apresentado a implementação e testes da proposta. Por fim, no Capítulo 5 são apresentadas as conclusões sobre este trabalho.

2 Revisão Bibliográca

2.1 Introdução ao SIP

O SIP é um protocolo da camada de sessão do modelo OSI (camada de aplicação no modelo TCP/IP) que pode estabelecer, modificar e encerrar sessões ou chamadas multimídia. Protocolos de internet baseados em texto como *HyperText Transfer Protocol* (HTTP) e o *Simple Message Transfer Protocol* (SMTP) inspiraram o SIP que foi criado para estabelecer, mudar e terminar chamadas de um ou mais usuário de forma independente do conteúdo da chamada. Foi especificado inicialmente por Handley et al. (1999) e posteriormente atualizada por Rosenberg et al. (2002). Hoje em dia, SIP é um dos protocolos mais utilizados para VoIP e está presente em quase todos os telefones IP no mercado.

- O protocolo SIP suporta os seguintes quatro recursos para estabelecer e encerrar sessões multimídia:
  - Localização do usuário: define o endereço de destino final usado para comunicação.
  - Disponibilidade do usuário: define se o usuário está disponível para estabelecer uma sessão.
  - Estabelecimento de chamada: determina os parâmetros para o chamador e o chamado e informa as duas partes sobre o andamento da chamada (toque, retorno de chamada, congestionamento).
  - Gerenciamento de chamadas: transferência e fechamento de sessão.

2.2 SIP: Fluxo Básico

O protocolo SIP foi projetado como parte de uma arquitetura multimídia contendo outros protocolos como *Resource Reservation Protocol* (RVSP), *Real-Time Protocol* (RTP), *Real-Time Streaming Protocol* (RTSP) e *Session Description Protocol* (SDP) - não dependendo deles para sua operação.

O formato do endereço SIP, que assemelha-se ao do e-mail, seguia as orientações de Berners-Lee, Fielding e Masinter (1998), que já se tornou obsoleta e deu lugar a Berners-Lee, Fielding e Masinter (2005):

- `sip:belmiro@sip1.com`

- `sip:+5548987654321@sip1.com`
- `sip:987654321@sip1.com`

Na arquitetura SIP, temos agentes e servidores de usuários.

O SIP é bastante flexível em termos de arquitetura. Para este documento, para fins didáticos, será adotado o modelo distribuído *peer-to-peer* com um servidor de sinalização, conforme a figura 1. O servidor trata apenas a sinalização, enquanto os clientes de agente de usuário e os servidores de agente de usuário manipulam sinalização e mídia.

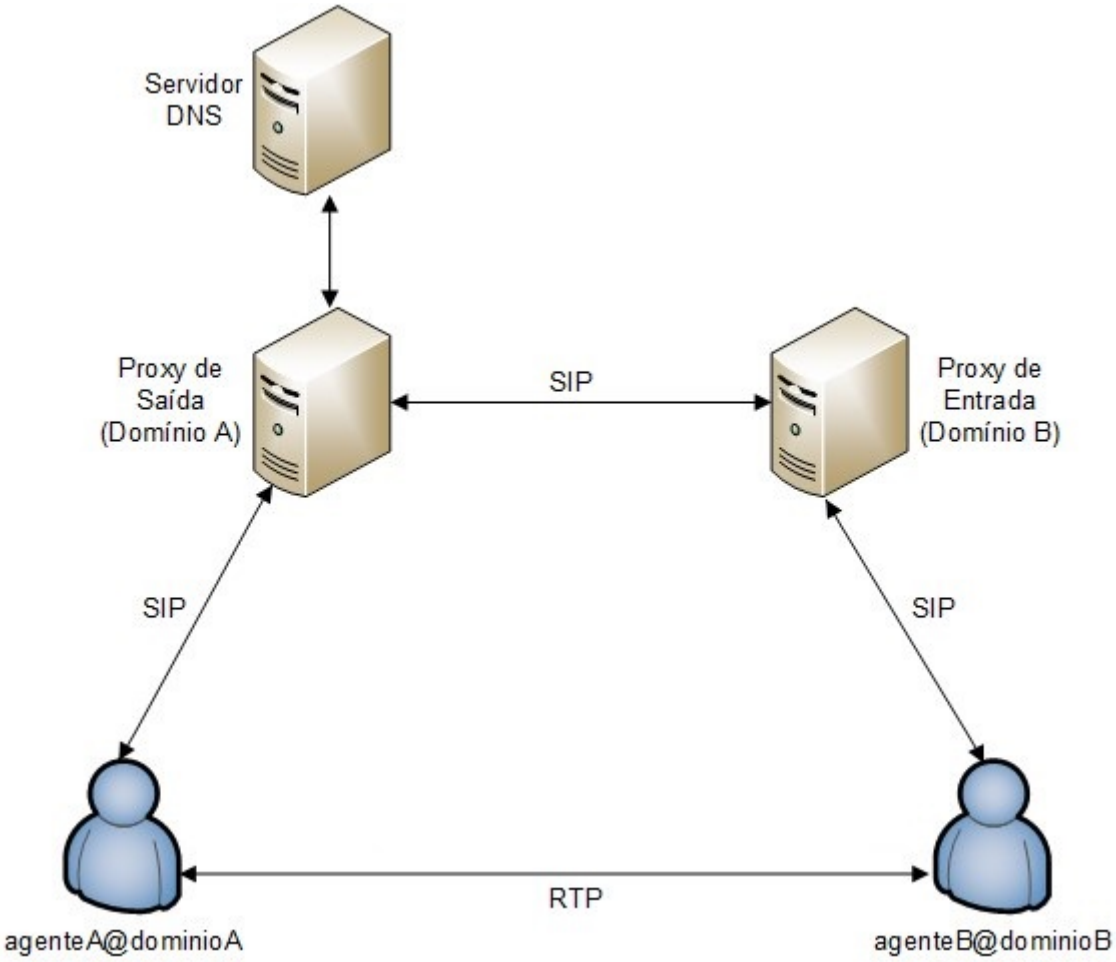

Figura 1 – Exemplo de cenário de telefonia IP em trapézio usando SIP. Fonte: elaborado pelo autor.

No modelo SIP, um agente de usuário - geralmente um telefone SIP - começará a se comunicar com seu proxy SIP (proxy de saída ou seu proxy local) usando uma mensagem conhecida como `INVITE` para enviar a chamada. O proxy de saída verá que a chamada é direcionada para um domínio externo. Ele procurará o servidor DNS para o endereço do domínio de destino e resolverá o endereço IP.

Em seguida, o proxy de saída encaminhará a chamada para o proxy SIP responsável pelo domínio B. O proxy de entrada verificará, na sua tabela de localização para o endereço

IP do agente B, se este endereço foi inserido na tabela de localização por um processo de registo anterior. Se o proxy de entrada pode localizar o endereço, ele encaminhará a chamada para o agente B. Depois de receber a mensagem SIP, o agente B terá todas as informações necessárias para estabelecer uma sessão RTP com o agente A (SCHULZRINNE et al., 2003). Usando uma mensagem com o método `BYE` terminará a sessão.

2.3 Componentes do SIP

Toda a sinalização SIP flui através do servidor proxy SIP. Por outro lado, a sinalização de mídia, transportada pelo protocolo RTP, flui diretamente de um ponto final para outro. Alguns dos componentes do SIP são:

- *User Agent* (UA): o terminal SIP (telefone IP, *Analog Telephone Adapter* (ATA), *softphone* e assim por diante).
- *User Agent Client* (UAC): o cliente ou terminal que inicia a sinalização SIP.
- *User Agent Server* (UAS): o servidor que responde à sinalização SIP proveniente de um UAC.
- Servidor Proxy: recebe solicitações de um UA e transfere para outro proxy SIP se este terminal específico não estiver sob seu domínio
- Servidor redirecionador: recebe solicitações e responde ao chamador com uma mensagem contendo dados sobre o destino.
- Servidor de localização: fornece os endereços de contato do destinatário aos servidores de proxy e redirecionamento.

2.4 Servidor de Registro

O protocolo SIP permite o emprego de um servidor de registro que aceita as solicitações `REGISTER` e salva as informações recebidas dentro desses pacotes no banco de dados de localização para seus domínios gerenciados. O servidor de registo pode aceitar outros tipos de informação como scripts *Call Processing Language* (CPL), e não apenas os endereços IP do cliente. Através do banco de dados de localização por domínio que amarzena todos os telefones associados aos seus respectivos endereços IP é possível o estabelecimento das sessões. Todas as atualizações sobre a localização corrente de cada usuário é recebida neste servidor o qual efetua a resolução dos nomes similar aos servidores *Domain Name System* (DNS) para uma rede web.

Em Johnston et al. (2003) são definidas boas práticas recomendadas para implementar um conjunto mínimo de funcionalidade para uma rede de comunicações SIP sobre IP. De acordo com o documento, existem cinco fluxos básicos associados com o processo de registro de um agente de usuário. Eles estão elencados a seguir.

2.4.1 Novo Registro bem-sucedido

Bob envia uma solicitação de registro (REGISTER) para o servidor SIP. O pedido inclui a lista de contatos do usuário. Devido a falta de proteção de integridade no HTTP Digest e o perigo de sequestro de registro o utiliza-se com transporte TLS para a autenticação. O servidor retorna a Bob uma resposta de não autorizado (`401 Unauthorized`). Bob criptografa as informações do usuário conforme o desafio recebido do servidor e envia um segundo registro (REGISTER) com a autorização e a senha criptografada. Servidor recebe o pedido de Bob e se a senha for correta o registra na sua base de dados e envia a reposta (200 OK) significando um registro bem-sucedido. Assume-se que Bob não tenha registrado anteriormente com este servidor. Diálogo abaixo na figura 2.

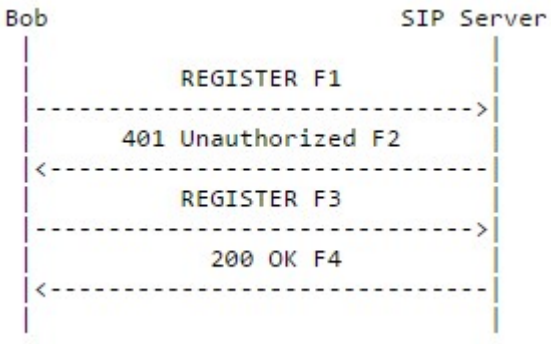

Figura 2 – Novo Registro bem-sucedido. Fonte: Johnston et al. (2003).

2.4.2 Atualização da base de contatos

Bob envia uma solicitação de registro (REGISTER) para o servidor SIP a fim de atualizar a lista de endereços onde o servidor SIP irá redirecionar ou encaminhar solicitações INVITE. O pedido inclui a lista de contatos atualizada. O servidor recebe a solicitação e não desafia Bob pois ele já se autenticou com o mesmo. As credencias do usuário são validadas e sua lista de contatos atualizada retornando uma reposta (200 OK) para Bob. A resposta inclui a lista de contatos atual do usuário em cabeçalhos de contato. Troca de mensagens exemplificadas na figura 3.

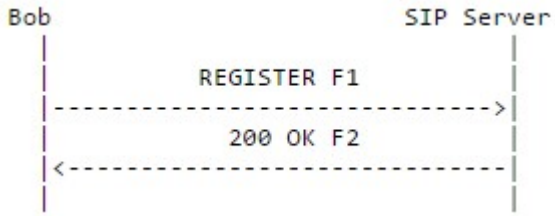

Figura 3 – Atualização da base de contatos. Fonte: Johnston et al. (2003).

2.4.3 Solicitação de base de contatos atuais

Bob envia uma solicitação de registro (REGISTER) para o servidor proxy desejando consultar a lista de contatos atual do usuário. O servidor recebe a solicitação e não desafia Bob pois ele já se autenticou com o mesmo. As credencias do usuário são validadas e sua lista de registro atual é enviada na reposta (200 OK) para Bob, como exemplificado na figura 4.

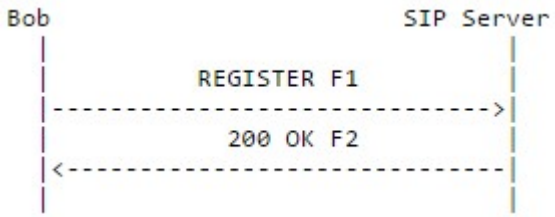

Figura 4 – Solicitação de base de contatos atuais. Fonte: Johnston et al. (2003).

2.4.4 Cancelamento do Registo

Bob envia uma solicitação de registro (`REGISTER`) para o servidor SIP afim de cancelar seu registro. O servidor recebe a solicitação e não desafia Bob pois ele já se autenticou com o mesmo. As credencias do usuário são validadas e a lista de de contatos do usuário é limpa e o servidor retorna uma reposta (`200 OK`) para Bob, conforme exemplificado na figura 5.

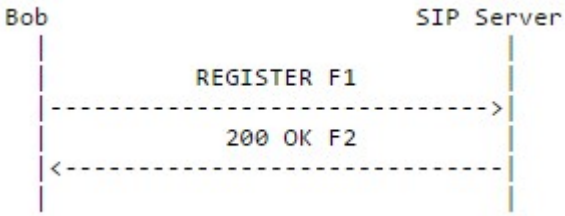

Figura 5 – Cancelamento do Registo. Fonte: Johnston et al. (2003).

2.4.5 Registro sem êxito

Bob envia uma solicitação de registro (REGISTER) para o servidor SIP. O servidor desafia Bob retornando uma resposta de não autorizado (401 Unauthorized). Bob

criptografa as informações do usuário conforme o desafio recebido do servidor e envia um segundo registro (REGISTER) com a autorização e as credenciais incorretas do usuário. O servidor retorna uma resposta (401 não autorizado) para o cliente SIP do Bob pois a senha do usuário está incorreta. Exemplo de diálogo abaixo na figura 6.

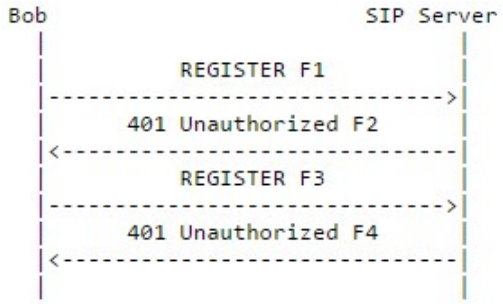

Figura 6 – Registro sem êxito. Fonte: Johnston et al. (2003).

2.5 Operando no modo SIP proxy

Opção mais comum utilizada, o modo SIP proxy é onde passa toda a sinalização SIP pelo servidor e facilita em processos como o de bilhetagem pois pode reter informações das sessões. As comunicações SIP durante o estabelecimento da sessão pode causar uma sobrecarga no servidor trazendo uma desvantagem para este modelo. Mesmo operando em modo SIP proxy os pacotes RTP sempre irão diretamente de um destino final para o outro.

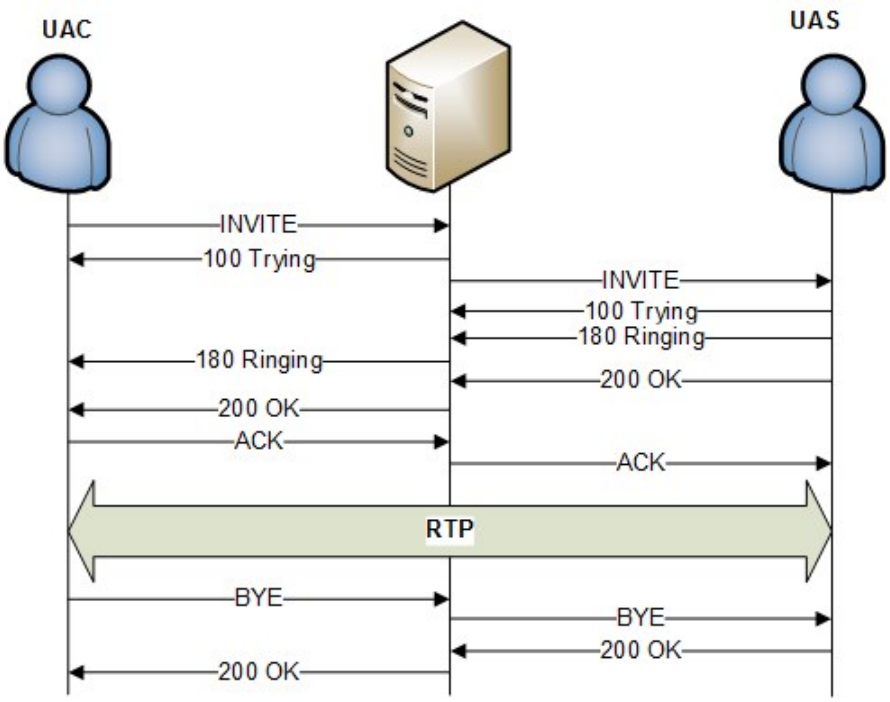

Figura 7 – Operando no modo SIP proxy. Fonte: elaborado pelo autor.

2.6 Operando como redirecionamento SIP

Modo de redirecionamento utilizado em casos onde precisa-se de alta escalabilidade e não necessita reter informações para fins contabilísticos ou de faturação. Possibilita a transmissão de milhões de chamadas por hora. Fornece resolução de nome e localização de usuário retornando o pedido ao UAC para que o mesmo contacte o UAS diretamente.

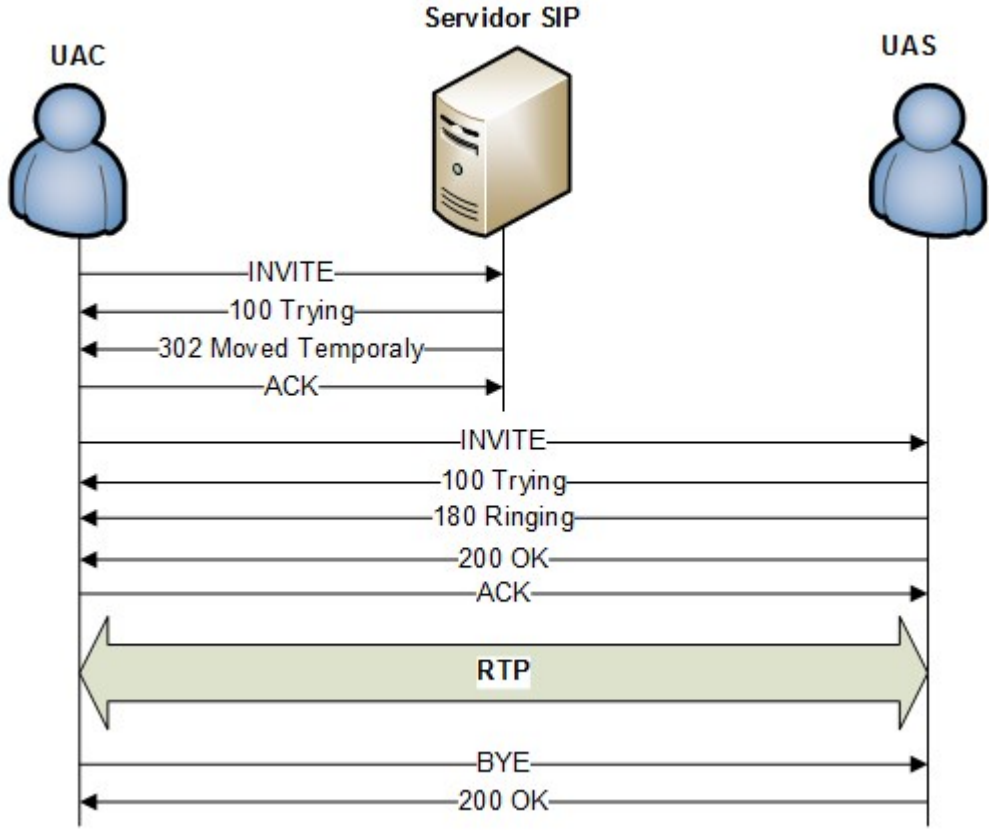

Figura 8 – Operando como redirecionamento SIP. Fonte: elaborado pelo autor

2.7 Mensagens básicas

As mensagens básicas enviadas em um ambiente SIP são:

Mensagem	Descrição	RFC
ACK	Reconhecer um INVITE	3261
BYE	Terminar uma sessão existente	3261
CANCEL	Cancelar um registo pendente	3261
INFO	Informação de sinalização de chamada intermediária	2976
INVITE	Estabelecimento da sessão	3261
MESSAGE	Transporte de mensagem instantânea	3428
NOTIFY	Enviar informações depois de subscrever	3265
PRACK	Reconhecer uma resposta provisória	3262
PUBLISH	Informações de estado de carregamento para o servidor	3903
REFER	Pedir outro UA para agir sobre URI	3515
REGISTER	Registrar o usuário e atualizar a tabela de localização	3261
SUBSCRIBE	Estabelecer uma sessão para receber futuras atualizações	3265
UPDATE	Atualizar informações de estado de uma sessão	3311

Tabela 1 – Mensagens SIP. Fonte: elaborado pelo autor

Na maioria das vezes, são usadas as mensagens REGISTER, INVITE, BYE e CANCEL. Algumas mensagens são usadas para outros recursos. Como exemplo, INFO é usado para informações de sinalização DTMF e de chamada intermediária. PUBLISH, NOTIFY e SUBSCRIBE dar suporte a sistemas de presença. REFER é usado para transferência de chamadas e MESSAGE para aplicativos de bate-papo. Mensagens mais recentes podem aparecer dependendo do processo de padronização do protocolo. As respostas a essas mensagens estão no formato de texto, como no protocolo HTTP.

2.8 Diálogo SIP

As mensagens SIP são rotuladas em sequência. Conforme a figura 9, userA utiliza um telefone IP para chamar pela rede outro telefone IP. Utiliza-se dois proxies SIP para completar a chamada. O userA utiliza sua identidade SIP (sip:userA@sipA.com), chamada SIP URI, para chamar o userB. Pode-se utilizar também um SIP URI seguro (sips:userA@sipA.com) utilizando TLS no transporte entre chamador e destinatário. UserA envia uma solicitação INVITE ao userB contendo um certo número de campos de cabeçalho que fornecem informações adicionais sobre a mensagem. Incluem identificador exclusivo, o destino e informações sobre a sessão:

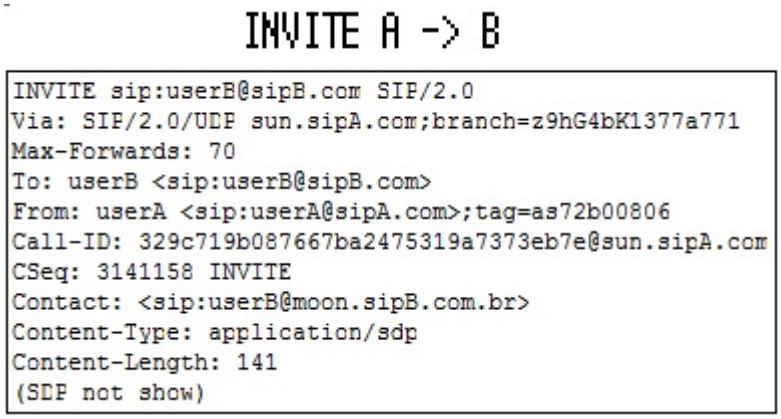

Figura 9 – Requisição SIP: Invite. Fonte: elaborado pelo autor.

O cabeçalho da mensagem SIP vai da primeira linha até o campo Content-Length. A primeira linha da mensagem de solicitação exibe o método (INVITE), a identidade SIP do destino (sip:userB@sipB.com) e a versão do protocolo (SIP/2.0).

A seguir vem o campo Via, que exibe o endereço e a porta para o qual o usuário userA espera receber a resposta de sua solicitação. O endereço utilizado é “sun.sipA.com” e a porta 5060 que é a porta padrão do SIP.

Ainda no campo Via, há o parâmetro branch que serve como um identificador de transação. Desta forma, respostas referentes a esta transação podem ser identificadas, pois deverão possuir o mesmo valor no parâmetro branch.

O campo Max-Forwards exibe a quantidade máxima de dispositivos SIP que a mensagem pode atravessar. A cada salto o campo é decrementado e quando chega a zero é descartado pelo dispositivo SIP que receber a mensagem.

O campo To carrega um nome de exibição (opcional) e a URI SIP que identifica o destinatário.

O campo From carrega um nome de exibição (opcional) e a URI SIP que identifica o chamador. Observa-se também a presença do parâmetro tag, que é uma sequência de caracteres gerada aleatoriamente pela origem.

O próximo campo é o Call-ID, outra sequência de caracteres aleatória gerada pela origem com o objetivo de identificar uma sessão SIP e, portanto, deve ser única no dispositivo SIP de origem. Os User Agents de origem e destino contribuem com a identificação da chamada cada um com mais uma sequência de caracteres aleatória, que são os parâmetros tag nos campos To e From. Esses três valores (Call-ID, tag do campo To e tag do campo From) identificarão completamente um diálogo aberto entre origem e destino. Nesta mensagem não há parâmetro tag no campo To porque ele será adicionado quando o destinatário responder a esta solicitação INVITE.

O campo seguinte é o CSeq (Command Sequence), que traz um valor inteiro e o

nome do método. A cada nova solicitação enviada dentro de um diálogo o valor inteiro deve ser incrementado.

O próximo campo na mensagem INVITE é o Contact, que contém o SIP URI do originador da chamada para o qual o User Agent de destino poderá encaminhar as transações seguintes diretamente.

Os dois campos seguintes, Content-Type e Content-Length, informam o tipo de mensagem que está sendo carregada no corpo do protocolo SIP e qual o tamanho desta mensagem, respectivamente.

2.8.1 Diálogo x Sessão x Transação

Diálogo: uma relação entre dois agentes que persiste por algum tempo, e identificada por um Call-ID. Os diálogos facilitam o sequenciamento e roteamento das mensagens entre UAS.

Transação: sequência de mensagens entre dois agentes iniciando com uma requisição e terminando com uma resposta final.

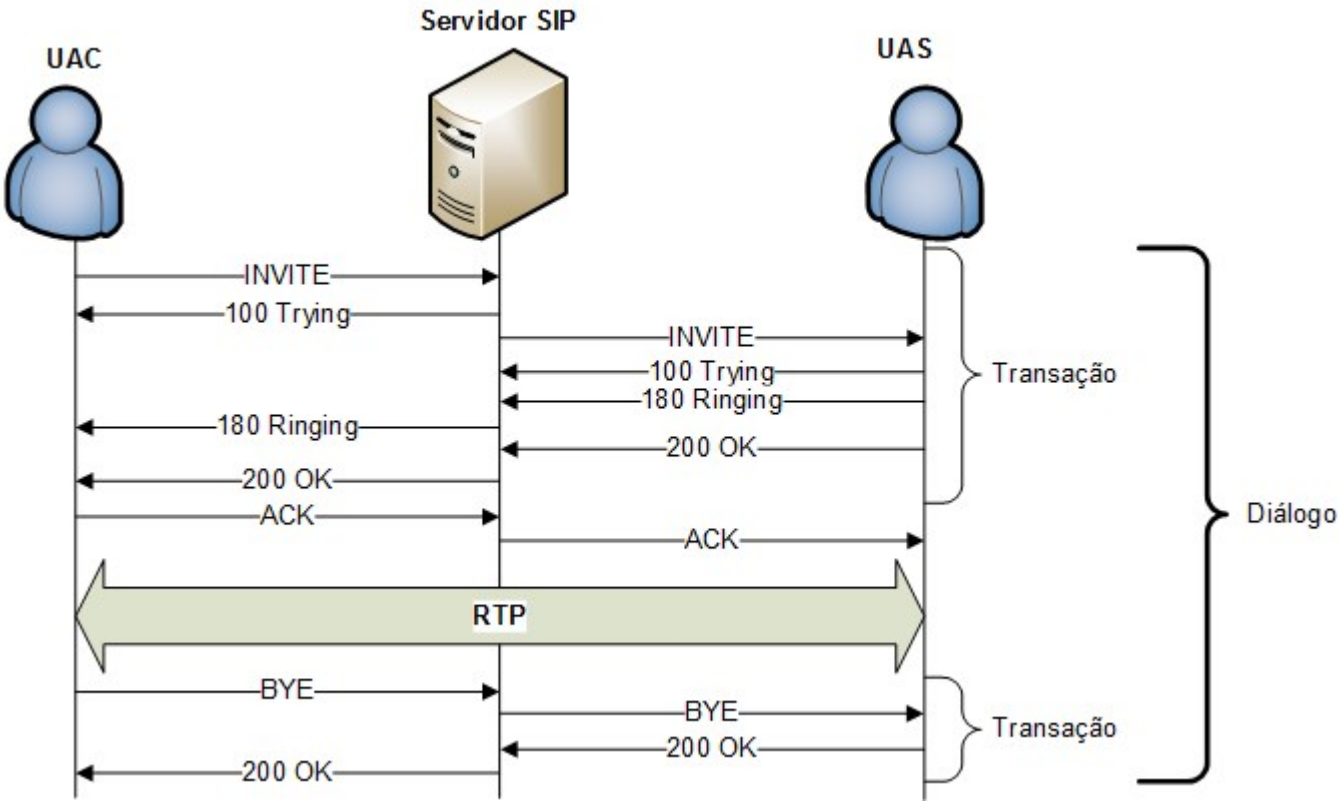

Figura 10 – Diálogo e transação SIP. Fonte: elaborado pelo autor.

Sessão: é a chamada em si, ou seja a troca de pacotes de áudio que corresponde a uma sessão RTP.

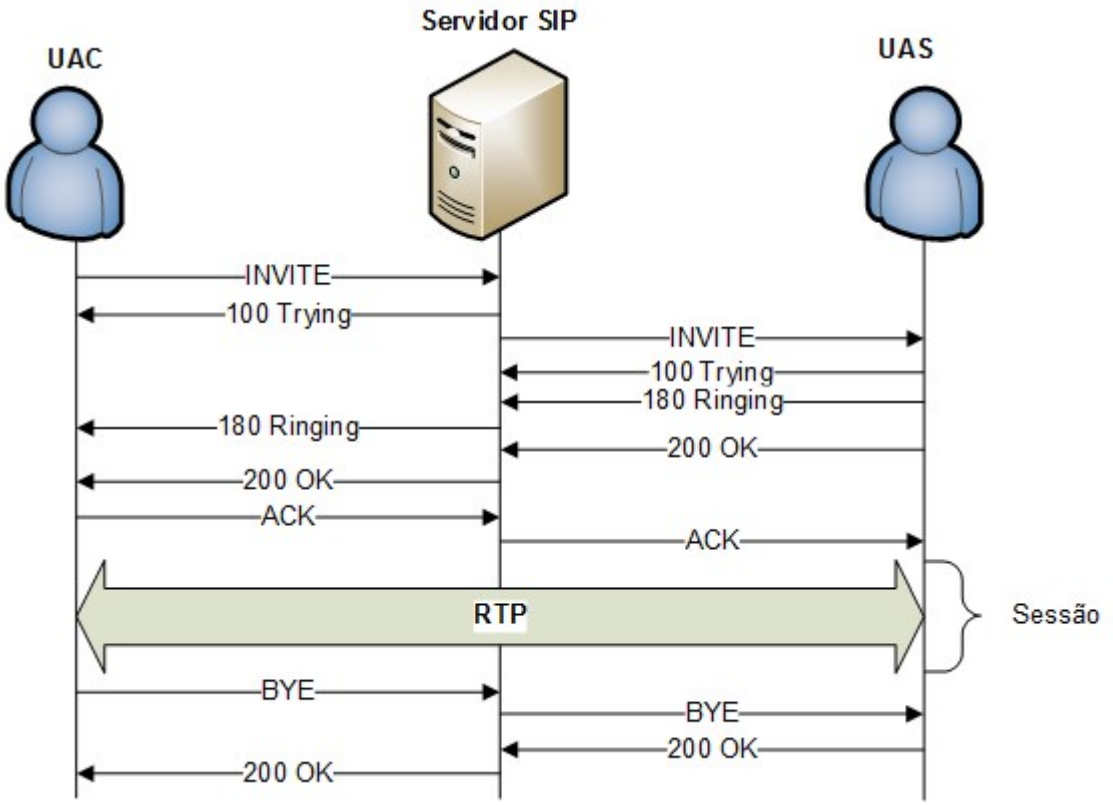

Figura 11 – Sessão SIP. Fonte: elaborado pelo autor.

2.9 OpenSIPS

Software livre de código aberto e documentação pública. OpenSIPS implementa um servidor SIP Proxy utilizado para o desenvolvimento de soluções em telefonia IP, como é o caso deste trabalho. Capaz de lidar com a comunicação via voz, texto e vídeo (GONÇALVES, 2010).

Destinado a demanda de processamento de milhares de chamadas simultâneas. Possui código aberto compatível com Rosenberg et al. (2002). OpenSIPS possui tamanho compacto capaz de lidar com extrema rapidez nas solicitações de encaminhamento e foi projetado para lidar com milhões de usuários com um único servidor (GONÇALVES, 2010).

OpenSIPS é baseado em SIP Express Router (SER), originalmente desenvolvido pelo Instituto de Pesquisa FhG Fokus, em Berlim, na Alemanha e liberado sob a licença GPL. OpenSERs foi a primeira bifurcação do projeto SER original. Em 2004, FhG Fokus iniciou uma derivação do projeto SER criando o iptel.org. Em 2005, a variante comercial de iptel foi vendida para a Tekelec. A equipe de desenvolvimento do núcleo foi dividida em dois, alguns deles foram para iptel.org e os outros deixaram FhG para começar uma empresa chamada Voice System, o principal mantenedor do projeto OpenSER que começou em 2005. O projeto OpenSER foi renomeado e bifurcado em 2008. Agora, existem duas variantes: Kamailio e OpenSIPS (GONÇALVES, 2010).

2.9.1 Características

2.9.1.1 Velocidade

Seu desenvolvimento usando ANSI C com algumas rotinas de assembly é o responsável pela velocidade. Mesmo em hardware de baixo custo pode-se lidar com dezenas de milhares de chamadas por segundo.

2.9.1.2 Flexibilidade

Com uma linguagem de programação flexível é possível atender até mesmo os cenários mais complicados, todo seu comportamento pode ser definido pelo seu administrador através da linguagem de programação contida no seu arquivo de configuração.

2.9.1.3 OpenSIPS é extensível

OpenSIPS possui inúmeros módulos diferentes que podem ser carregados e utilizados no arquivo de configuração/roteamento. Os novos códigos desenvolvidos em C independentes do núcleo OpenSIPS e iniciados no momento da execução proporcionam as novas funcionalidades. O conceito é semelhante ao dos módulos em servidores Web Apache.

Novas camadas de programação foram adicionadas ao longo das versões como a função de Processamento de Linguagem (CPL) para simplificar os scripts de roteamento. OpenSIPS já possui mais de 100 módulos diferentes capazes de agregar inúmeras funcionalidades.

2.9.1.4 Portabilidade

OpenSIPS é extremamente portátil devido seu desenvolvimento em ANSI C e está disponivel para sistemas UNIX-like, como Linux, Solaris e BSD.

2.9.1.5 Compacto

OpenSIPS possui um pequeno núcleo, e pode aumentar um pouco seu tamanho quando agregados alguns módulos. Por ser compacto seu uso é interessante para plataformas embarcadas.

2.9.2 Cenários de uso

O OpenSIPS é comumente utilizado como servidor de Registro e no modo SIP Proxy. Porém é possível utiliza-lo em outras aplicações como balanceador de carga, servidor de presença, mensagens instantâneas e NAT transversal. Possui suporte a IPv4 e IPv6. Através da inserção de novos módulos pode ser utilizado em cenários tais como os da tabela 2:

Módulos	**Funcionalidade**
DISPATCHER, PATH	Balanceador de Carga
MEDIAPROXY, RTPPROXY, NATHELPER	Nat Transversal
PRESENCE	Servidor de Presença
IMC, XMPP	Mensagem Instantânea

Tabela 2 – Módulos e funcionalidades. Fonte: elaborado pelo autor

2.9.3 Núcleo e módulos

OpenSIPS possui um núcleo responsável pelas funcionalidades básicas e manuseio das mensagens SIP. Boa parte das funções são controladas pelos módulos. No arquivo `opensips.cfg` é realizada a configuração do OpenSIPS, desde as definições globais aos módulos que devem ser carregados e seus respectivos parâmetros. Toda a parte de roteamento também é definida ao longo deste arquivo principal.

2.9.4 Seções do opensips.cfg

O arquivo `opensips.cfg` tem várias seções, que são as seguintes:

- Definições globais: Parâmetros que influenciam no núcleo OpenSIPS e seus módulos. A definição da porta de escuta e protocolo é realizada nesta seção.
- Módulos: O OpenSIPS possui inúmeros módulos que devem ser carregados nesta seção para possibilitar o uso dos mesmos. Devem ser carregados com `loadmodule`.
- Configuração de módulos: Para o correto funcionamento dos módulos é necessário que seus parâmetros sejam devidamente configurados. Estes parâmetros são configurados usando `modparam` (`modulename`, `parametername`, `parametervalue`).
- Bloco de roteamento principal: Responsável pelo processamento de cada pedido SIP recebido.
- Blocos de roteamento secundários: Através do comando `route()` podem ser definidos novos blocos de roteamento que funcionam como sub-rotinas no *script* OpenSIPS.
- Blocos de resposta de roteamento: Responsável pelo processamento das mensagens de resposta (provisórias/informação, respostas bem-sucedidas, ou respostas malsucedidas), comumente 200 OK.
- Blocos de falha de roteamento: Processamento das mensagens de resposta de falha/erros no cliente.
- Blocos de roteamento Branch: Responsável por cada branch (identificador de transação) de cada pedido SIP.
- Blocos de roteamento locais: Utilizado quando o OpenSIPS através do módulo de transações (`tm.so`) gera solicitações internas.
- Bloco de erro de roteamento: Ao detectar um erro ao analisar um pedido SIP é executado este bloco.

2.9.5 OpenSIPS e Banco de Dados

O OpenSIPS possui suporte a banco de dados e permite o armazenamento dos dados em um cluster de banco de dados separado. Este cluster pode fornecer uma solução de armazenamento flexível com disponibilidade e processamento de dados contínuos mesmo em caso de falhas individuais de servidores devido a replicação dos dados entre os servidores do cluster, criando assim um sistema de alta disponibilidade.

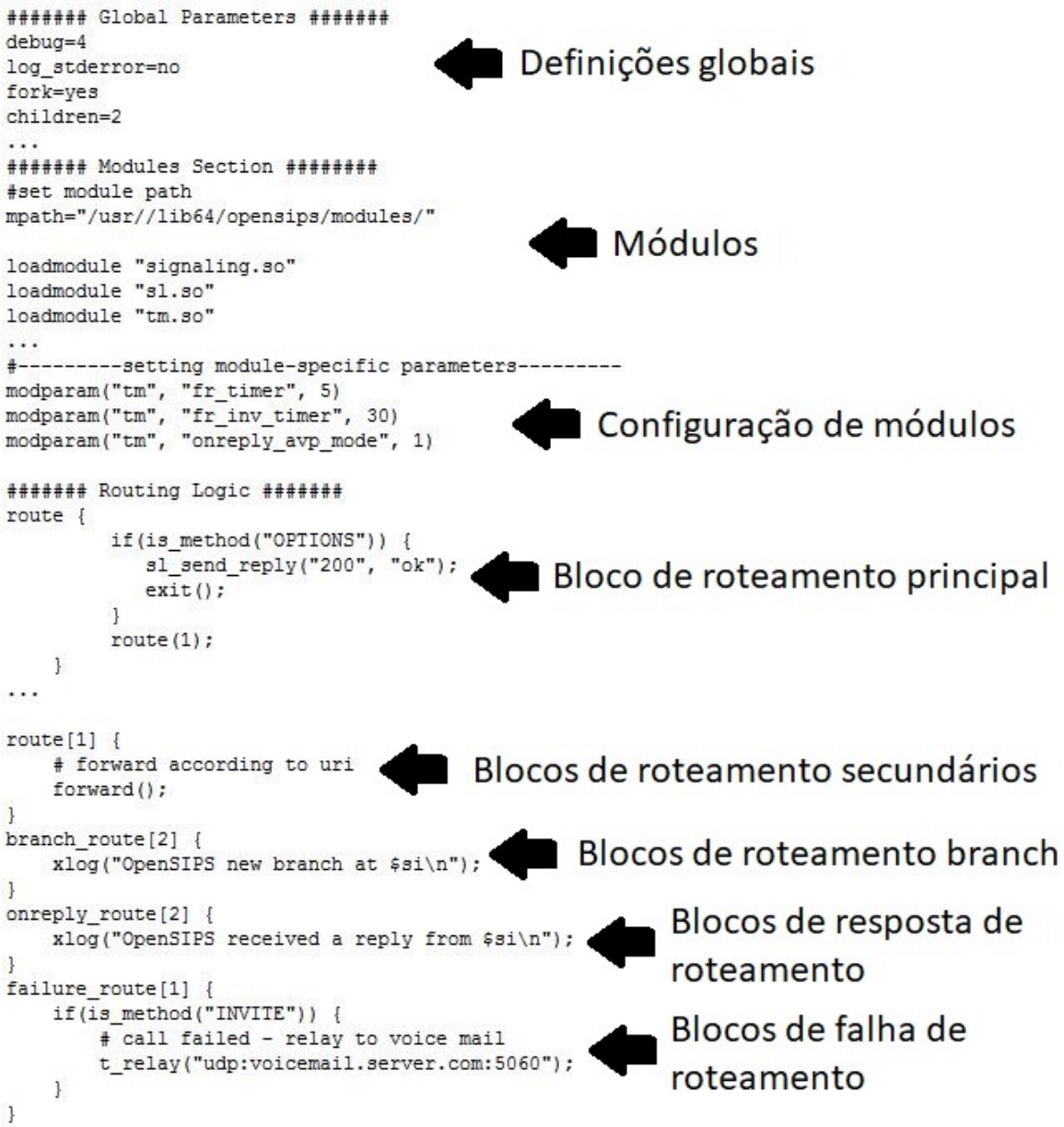

Figura 12 – Arquivo principal de configuração OpenSIPS, `opensips.cfg`. Fonte: elaborado pelo autor

Esta característica torna possível assim o armazenamento dos registros SIP, portanto, vários servidores REGISTRAR podem atender vários agentes se comportando como uma nuvem de servidores - a única coisa que não pode ser compartilhada é o diálogo (ele é atômico). Dessa forma, pode-se fazer uso de contêineres para as “n” instâncias da aplicação.

2.10 O Contêiner

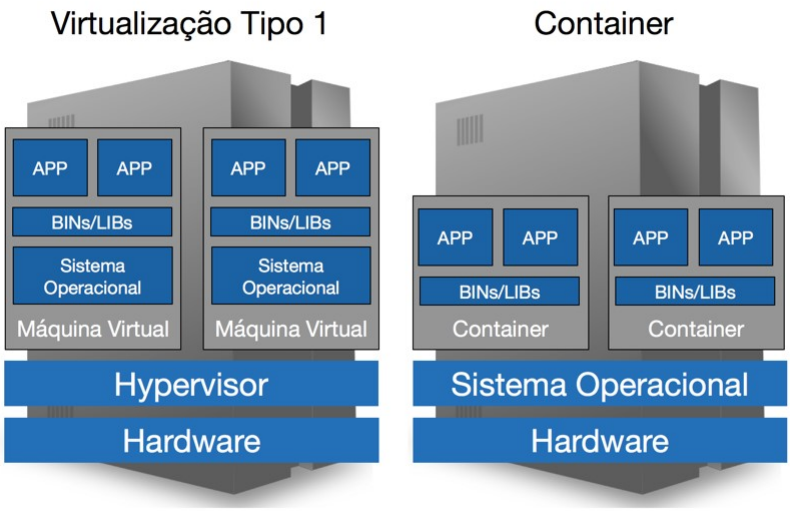

Figura 13 – Virtualização x Contêiner. Fonte: TLC-BR

Conforme a figura 13 o contêiner é uma maneira de virtualizar aplicações dentro de um servidor Linux. É possível dentro de um único sistema hospedeiro criarmos múltiplas instâncias de um determinado sistema operacional, sendo cada instância isoladas uma das outras.

Tudo começou com o comando `chroot` lançado em 1979 pelo Unix V7 que é utilizado na separação de acessos a diretórios afim de evitar acesso livre a todo o ambiente, como exemplo arquivos do usuário root. Com o lançamento do comando jails do FreeBSD 4 além do isolamento do sistema de arquivos permitiu o isolamento de processos, acompanhados pela empresa Sun a mesma desenvolveu uma solução baseada em conteineres com o nome de Solaris Zones, compatível apenas em Solaris. Em 2006 os engenheiros do Google iniciaram o desenvolvimento de um recurso de kernel do Linux chamado cgroups. Através deste é possível limitar, contabilizar e isolar o uso de recursos (CPU, memória, disco, rede, etc.) de uma coleção de processos. Através do projeto LXC iniciado em 2008 começaram a buscar uma solução completa e estável para a criação e gerenciamento de conteineres. Este projeto utilizou-se de `cgroups`, `namespaces` e `chroot` (ALMEIDA, 2015).

O serviço em contêiner permite o empacotamento do código e suas depêndencias a

serem executados em outro ambiente. Devido seu tamanho compacto é possível em um único computador rodar diversos contêineres ao invés do uso único de um sistema operacional e do software em um computador. O uso desta tecnologia facilita o desenvolvimento das aplicações e testes pois na medida que o software passa pelos estágios de desenvolvimento, ele pode sair da maquina do desenvolvedor para um outro ambiente e depois aplicado em ambiente de produção possuindo a velocidade de implementação variando de mili a poucos segundos (ALMEIDA, 2015).

Neste trabalho será adotada a implementação de contêiner utilizando-se Docker devido sua melhor compatibilidade com a ferramenta de orquestração de contêineres, conforme Oliveira (2016).

2.10.1 Dockerfile

A criação de imagens com sistema operacional e a aplicação desejada ocorre a partir do arquivo de definição chamado Dockerfile. Pode-se usar uma imagem pronta dos vários repositórios na Internet, como da própria Docker ou criar o seu.

O Dockerfile é um documento de texto que contém todos os comandos que um usuário pode chamar na linha de comando para montar uma imagem. Usando o comando `docker build` o usuário executará várias instruções de linha de comando em sucessão criando a imagem desejada.

Em resumo, o Dockerfile é um arquivo texto com instruções, comandos e passos que seriam executados manualmente, basicamente o Docker executa uma receita de bolo.

3 Desenvolvimento

Sabendo-se da nova infraestrutura de servidores e serviços do câmpus São José, que é baseada nos trabalhos de conclusão de curso de Oliveira (2017) e de Reis (2017), este trabalho implementará telefonia IP escalável. A aplicação foi pensada para um ambiente distribuído (armazenamento centralizado com MySQL, processamento distribuído com contêineres e rede virtual distribuída com SDN), orquestração de contêineres e serviços (mapeamento de portas/serviços por DNS e rede).

Foi utilizado o software livre OpenSIPS[1] pois atende as necessidades de grande carga de processamento de chamadas, escalabilidade e adaptabilidade.

3.1 Arquitetura da Aplicação

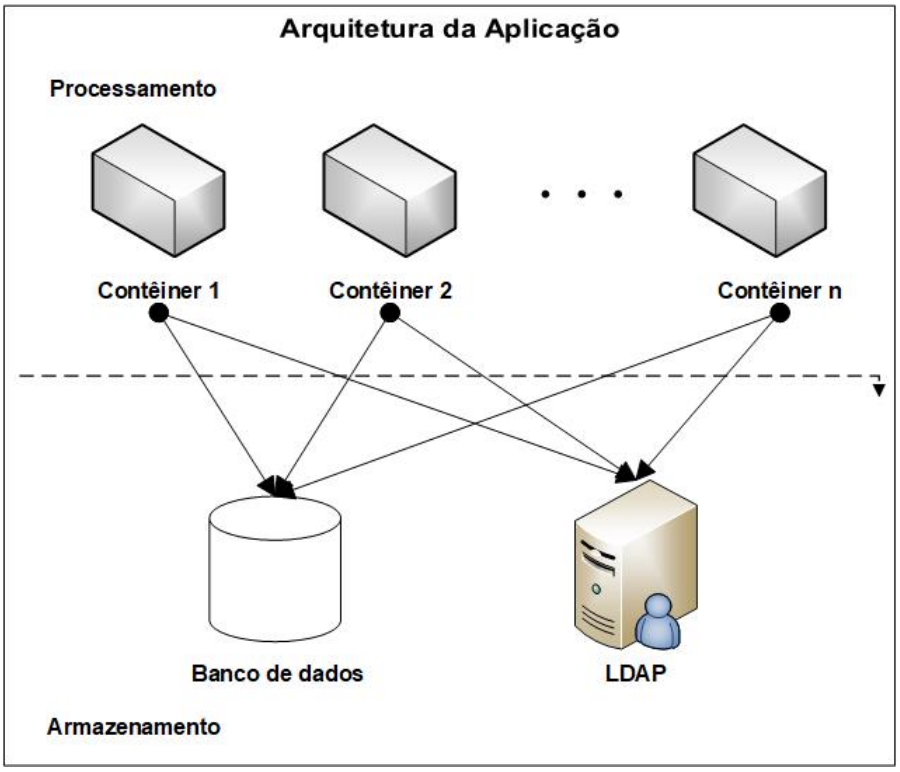

Figura 14 – Arquitetura da Aplicação. Fonte: elaborado pelo autor.

O processamento ocorrerá nos contêineres OpenSIPS que se conectarão ao mesmo banco de dados MySQL independente utilizado para gerenciamento das sessões dos usuários. Através do módulo *Lightweight Directory Access Protocol* (LDAP) os usuários conseguirão registro para realização de chamadas. A figura 14 ilustra a arquitetura da solução.

[1] <https://opensips.org>

3.2 Infraestrutura da Aplicação

A estrutura implantada conta com três estações de trabalho, contando com processadores Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz, ambas com 2GB de memória RAM e 32GB de armazenamento configuradas como um cluster, conforme a figura 15.

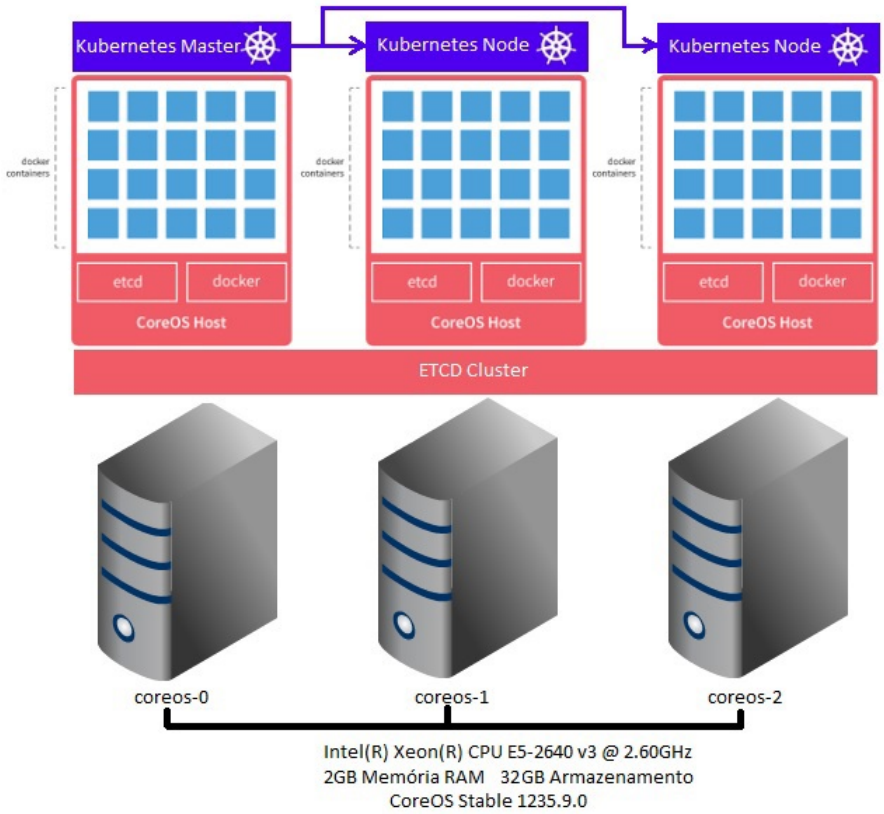

Figura 15 – Infraestrutura da Aplicação. Fonte: Oliveira (2017)

3.3 Orquestração e Serviço distribuído

Para o usuário final é apresentado um sistema único para comunicação via voz e videoconferência. Todas as requisições são gerenciadas pelo Kubernetes que as distribui para os contêineres.

A ferramenta Kubernetes realiza a automatização, distribuição de carga, monitoramento e orquestração conforme a figura 16.

Através da função de controle de replicação, os Pods, contêineres ou grupos de contêineres de um mesmo tipo de aplicação ou serviço, são criados e excluídos conforme a demanda de processamento. Este é um serviço interno do Kubernetes que permite quantificar ou escalar cada tipo de contêiner replicando e recriando unidades de contêineres conforme definido, de acordo com a demanda ou mediante à falha de alguma unidade de contêiner.

Para acesso externo aos contêineres, é atrelada determinada faixa de IP para um controlador de replicação através da função Services, que na prática define endereços IP para as instâncias de contêineres.

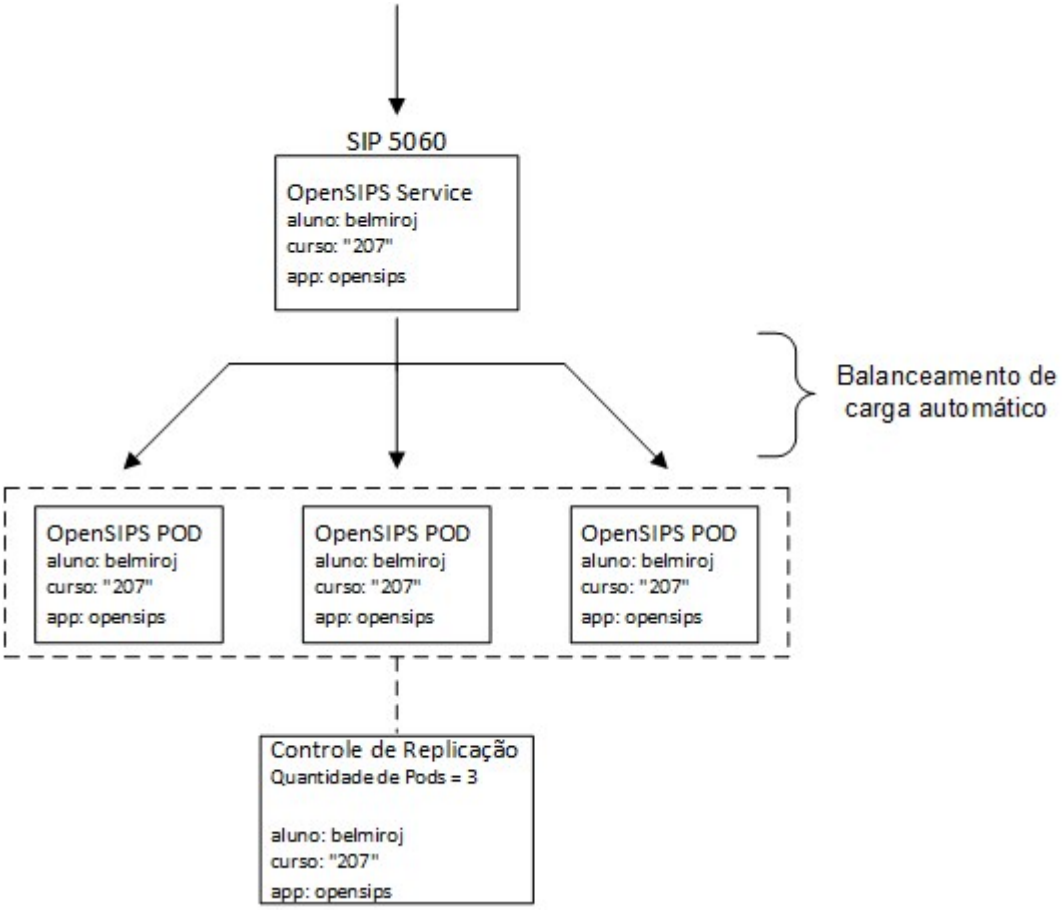

Figura 16 – Service, Controle de Replicação e Pod. Fonte: elaborado pelo autor

4 Implementação e testes

Etapas executadas para implementação da proposta.

4.1 Criando o Dockerfile

A criação do contêiner ocorreu a partir da criação de um Dockerfile que carrega o OpenSIPS com os módulos de banco de dados MySQL e de autenticação LDAP deixando as informações persistentes fora do contêiner. O sistema operacional no contêiner é o CentOS, recomendado pela própria equipe do OpenSIPS.

O primeiro passo na criação do Dockerfile é a escolha do sistema operacional, aproveitou-se uma imagem já existente no Docker Hub, repositório de imagens Docker, do sistema CentOS versão 6.9 que é compatível com o OpenSIPS.

É utilizado o comando `FROM` informando a partir de qual imagem será gerada a nova imagem.

```
1 FROM centos:centos6.9
```

Após descarregar a imagem do CentOS é realizada a instalação de pacotes necessários para a compilação do código fonte do OpenSIPS e dos seus módulos nativos que serão utilizados para este trabalho.

O comando `RUN` realiza a execução de um comando.

```
1 RUN yum -y install git gcc gcc-c++ bison flex zlib-devel openssl-
    devel mysql mysql-server mysql-devel subversion pcre-devel ncurses
    -devel ncurses openldap openldap-devel openssl openssl-devel
```

No repositório oficial Docker Hub[1] existe a imagem do OpenSIPS. Porém, a mesma é a versão básica e não possui todos os módulos necessários para o IFSC, como exemplo o suporte a banco de dados MySQL. Sendo assim, utilizou-se os arquivos encontrados no repositório oficial do OpenSIPS[2] e antes de realizar a compilação da versão foram adicionados os módulos não nativos a serem utilizados.

```
1 RUN git clone https://github.com/OpenSIPS/opensips.git -b 2.2 ~/
    opensips_2_2 && \
```

[1] <https://hub.docker.com>
[2] <https://github.com/opensips>

```
sed -i 's/db_http db_mysql db_oracle/db_http db_oracle/g' ~/
    opensips_2_2/Makefile.conf.template && \
sed -i 's/json ldap lua/json lua/g' ~/opensips_2_2/Makefile.conf.
    template && \
sed -i 's/snmpstats tls_mgm xcap/snmpstats xcap/g' ~/opensips_2_2/
    Makefile.conf.template && \
sed -i 's/proto_sctp proto_tls proto_wss pua/proto_sctp pua/g' ~/
    opensips_2_2/Makefile.conf.template && \
cd ~/opensips_2_2 && \
make all && make install && \
cd .. && rm -rf ~/opensips_2_2
```

4.2 Criando o Banco de Dados

O banco de dados roda em um contêiner separado, criou-se a base principal denominada `opensips` e as tabelas foram criadas a partir dos scripts existentes no repositório oficial do OpenSIPS no GitHub.

Configurado na tabela `domain` o domínio a ser utilizado.

```
git clone https://github.com/OpenSIPS/opensips.git
mysqladmin -u root password 'SUASENHA'
mysql -u root -p <<EOF
delete from mysql.user where not (host="localhost" and user="root");
grant all privileges on *.* to root identified by 'SUASENHA';
flush privileges;
create database opensips;
create user 'opensips'@'localhost' identified by 'SUASENHA';
grant all on opensips.* to 'opensips';
EOF
mysql -u opensips --password=SUASENHA -h localhost -e 'source /tmp/
    opensips/database/mysql/standard-create.sql' opensips
for i in *; do mysql -u opensips --password=SUASENHA -h localhost -e
    "source /tmp/opensips/database/mysql/$i" opensips ;done
```

4.3 Configurando o opensips.cfg

No arquivo `opensips.cfg` foi realizada a configuração para o pleno funcionamento do sistema.

4.3.1 Parâmetros globais

Configurados os protocolos e portas com apontamento para o endereço `0.0.0.0` e é responsável por toda comunicação externa.

```
1 listen=udp:0.0.0.0:5060
2 listen=tcp:0.0.0.0:5060
```

4.3.2 Seção de Módulos

Realizada configuração do diretório onde se encontrarão os módulos e o carregamento dos mesmos e seus parâmetros. O banco de dados foi configurado nos módulos que o utilizam.

Módulos alterados e/ou adicionados e configuração do diretório.

```
1  ####### Modules Section ########
2
3  #set module path
4  mpath="/usr/local/lib64/opensips/modules/"
5
6  #### URI module
7  loadmodule "uri.so"
8  modparam("uri", "use_uri_table", 0)
9  modparam("uri", "db_url", "mysql://opensips:SUASENHA@mysql/opensips")
10
11 #### LDAP Module
12 loadmodule "ldap.so"
13 modparam("ldap", "config_file", "/usr/local/etc/opensips/ldap.cfg")
14
15 ### MySQL Module
16 loadmodule "db_mysql.so"
17
18 #### USeR LOCation module
19 loadmodule "usrloc.so"
20 modparam("usrloc", "nat_bflag", "NAT")
21 modparam("usrloc", "db_mode",   2)
22 #modparam("usrloc", "accept_replicated_contacts", 1)
23 #modparam("usrloc", "replicate_contacts_to", 1)
24 modparam("usrloc", "db_url", "mysql://opensips:SUASENHA@mysql/
       opensips")
25
26 #### ACCounting module
27 loadmodule "acc.so"
28 modparam("acc", "early_media", 0)
29 modparam("acc", "report_cancels", 0)
```

```
30 modparam("acc", "detect_direction", 0)
31 modparam("acc", "db_url", "mysql://opensips:SUASENHA@mysql/opensips")
32
33 #### Auth
34 loadmodule "auth.so"
35 modparam("auth", "nonce_expire",  30)
36 modparam("auth", "disable_nonce_check", 0)
37 modparam("auth","username_spec","$avp(54)")
38 modparam("auth","password_spec","$avp(55)")
39 modparam("auth","calculate_ha1",1)
40
41 #### DOMAIN module
42 loadmodule "domain.so"
43 modparam("domain", "db_url", "mysql://opensips:SUASENHA@mysql/
      opensips")
44 modparam("domain", "db_mode", 1)   # Use caching
45 modparam("auth_db|usrloc|uri", "use_domain", 1)
46
47 #### Group module
48 loadmodule "group.so"
49 modparam("group", "db_url", "mysql://opensips:SUASENHA@mysql/opensips
      ")
```

4.3.3 Lógica de Roteamento

Configuração dos blocos de roteamento responsáveis por tratar as requisições SIP.

```
1  ####### Routing Logic ########
2
3  # main request routing logic
4
5  route{
6
7  ...
8
9     # Busca e validacao do usuario na base LDAP
10   if (!ldap_search("ldap://sipaccounts/ou=Users,dc=cefetsc,dc=edu,dc=
       br??sub?(uid=$tU)"))
11   {
12         switch ($retcode)
13         {
14         case -1:
15             # Usuario nao encontrado
16             sl_send_reply("404", "User Not Found");
17             exit;
18         case -2:
```

```
19              # Erro de comunicacao
20              sl_send_reply("500", "Internal server error");
21              exit;
22          default:
23              exit;
24          }
25    }
26
27    # Salvando a senha do usuario na AVP 55
28    ldap_result("userPassword/$avp(55)");
29
30    # Invocando o bloco de roteamento secundario
31    if (method=="REGISTER") {
32      route(2);
33      exit;
34    }
35
36 #Bloco de roteamento secundario
37 route[2] {
38
39          if (!is_present_hf("Authorization")) {
40            www_challenge("", "0");
41            exit;
42          }
43
44    $avp(54) = $tU;
45
46          if (!pv_www_authorize("")) {
47                  # authentication failed -> do challenge
48                  www_challenge("", "0");
49                  exit;
50          };
51          #Inserindo usuario na tabela de localizacao
52          save("location");
53          exit;
54
55 }
```

4.4 Autenticação de usuários

O IFSC possui uma base de usuários LDAP e se aproveitando deste recurso já implementado utilizaremos a autenticação dos usuários através do protocolo LDAP, onde o OpenSIPS utilizará o parâmetro `uid` como usuário e o parâmetro `userPassword` como senha do usuário existentes no OpenLDAP do IFSC.

Para realizar a consulta a base LDAP do IFSC através do OpenSIPS além do carregamento do módulo LDAP e da lógica de roteamento é necessário o carregamento do arquivo `ldap.cfg` para permitir o acesso aos dados.

```
[sipaccounts]
ldap_server_url             = "ldap://ldap.ifsc.edu.br"
ldap_bind_dn                = "cn=ldap,dc=cefetsc,dc=edu,dc=br"
ldap_bind_password          = "SUASENHA"
ldap_network_timeout        = 500
ldap_client_bind_timeout    = 500
```

Utilizando-se dos AVPs[3], foi realizado o armazenamento do usuário e senha utilizados pelo módulo `AUTH` para autenticação e registro dos usuários.

```
#Parametro para ativar os AVPs

modparam("tm", "onreply_avp_mode", 1)

#Parametros indicando que o usuario e senha utilizados na
    autenticacao serao os armazenados nos avps 54 e 55

modparam("auth","username_spec","$avp(54)")
modparam("auth","password_spec","$avp(55)")
```

Na seção da lógica de roteamento foi implementada a busca do usuário no LDAP. Primeiramente é realizada a validação do `uid`, se não encontrado é retornado o código `404 - User Not found` ou em caso de falha na comunicação com o LDAP o código `500 - Internal server error`. Caso e usuário existir continua e salva o parâmetro de senha na variável AVP 55 e logo vai para o bloco de roteamento secundário criado para realizar a autenticação e registro do usuário. Tendo uma resposta positiva da senha o usuário é inserido no banco de dados na tabela de localização de usuários denominada `location` e assim já encontra-se disponível para realizar e receber chamadas.

```

    # Busca e validacao do usuario na base LDAP
  if (!ldap_search("ldap://sipaccounts/ou=Users,dc=cefetsc,dc=edu,dc=
    br??sub?(uid=$tU)"))
  {
        switch ($retcode)
        {
        case -1:
            # Usuario nao encontrado
```

[3] <https://www.opensips.org/Documentation/Script-CoreVar-2-2#varavps>

```
9               sl_send_reply("404", "User Not Found");
10              exit;
11          case -2:
12              # Erro de comunicacao
13              sl_send_reply("500", "Internal server error");
14              exit;
15          default:
16              exit;
17          }
18    }
19
20    # Salvando a senha do usuario na AVP 55
21    ldap_result("userPassword/$avp(55)");
22
23    # Invocando o bloco de roteamento secundario
24    if (method=="REGISTER") {
25      route(2);
26      exit;
27    }
28
29 #Bloco de roteamento secundario
30 route[2] {
31
32          if (!is_present_hf("Authorization")) {
33            www_challenge("", "0");
34            exit;
35          }
36
37    $avp(54) = $tU;
38
39          if (!pv_www_authorize("")) {
40                  # authentication failed -> do challenge
41                  www_challenge("", "0");
42                  exit;
43          };
44          #Inserindo usuario na tabela de localizacao
45          save("location");
46          exit;
47
48 }
```

Outros tipos de autenticação como por exemplo através de banco de dados serão fechadas permitindo assim acesso somente a usuários do OpenLDAP.

Não foi possível a utilização das senhas já existentes na base LDAP do IFSC pois estão em formato SHA e SSHA e o módulo de autenticação do OpenSIPS não possui

suporte a este formato de criptografia. Para contornar esta limitação recomenda-se o uso de um atributo conforme Franks et al. (1999).

As senhas dos usuários localizadas no OpenLDAP serão consultadas e calculadas diretamente no OpenSIPS através do parâmetro calculate ha1 do módulo nativo do OpenSIPS, o `AUTH`, este módulo possui suporte as senhas.

4.5 Testes de registro

Foram realizados testes de registro e capturados os pacotes de sinalização SIP.

4.5.1 Registro de usuário

Diálogo quando é enviado usuário e senha corretos.

Figura 17 – Diálogo SIP de usuário e senhas corretos. Fonte: elaborado pelo autor.

É enviado o pedido de registro conforme figura 18 solicitando o registro no servidor.

```
REGISTER sip:172.30.6.99 SIP/2.0
Via: SIP/2.0/UDP 172.30.3.45:65390;branch=z9hG4bK-d8754z-266936115752cf7d-1
-d8754z-;rport
Max-Forwards: 70
Contact: <sip:antonioj@172.30.3.45:65390;rinstance=6098688189134ed5>
To: "antonioj"<sip:antonioj@172.30.6.99>
From: "antonioj"<sip:antonioj@172.30.6.99>;tag=74766707
Call-ID: NTg4YmIzNjA2Y2EwMjM1NGFlNzlkODc2ZDlkM2NiNTM.
CSeq: 1 REGISTER
Expires: 3600
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE, SUBSCRIBE
INFO
Supported: replaces
User-Agent: Bria Professional release 2.4 stamp 49381
Content-Length: 0
```

Figura 18 – Pedido de registro inicial. Fonte: elaborado pelo autor.

O servidor envia uma resposta não autorizado conforme 19 desafiando o usuário.

```
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP 172.30.3.45:65390;received=172.30.3.45;branch=z9hG4bK-d875
-266936115752cf7d-1---d8754z-;rport=65390
To: "antonioj"<sip:antonioj@172.30.6.99>;tag=1cdc36d543f1efe3ae3a68455465ad
.e130
From: "antonioj"<sip:antonioj@172.30.6.99>;tag=74766707
Call-ID: NTg4YmIzNjA2Y2EwMjM1NGFlNzlkODc2ZDlkM2NiNTM.
CSeq: 1 REGISTER
WWW-Authenticate: Digest realm="172.30.6.99", nonce="5a2628ba000000072dcb49
aac113d86e266bb42287293e"
Server: OpenSIPS (2.2.5 (x86_64/linux))
Content-Length: 0
```

Figura 19 – Resposta não autorizado. Fonte: elaborado pelo autor.

Um segundo pedido de registro conforme figura 20 é enviado com a autorização e os dados do usuário.

```
REGISTER sip:172.30.6.99 SIP/2.0
Via: SIP/2.0/UDP 172.30.3.45:65390;branch=z9hG4bK-d8754z-c96162259961af55-1
-d8754z-;rport
Max-Forwards: 70
Contact: <sip:antonioj@172.30.3.45:65390;rinstance=6098688189134ed5>
To: "antonioj"<sip:antonioj@172.30.6.99>
From: "antonioj"<sip:antonioj@172.30.6.99>;tag=74766707
Call-ID: NTg4YmIzNjA2Y2EwMjM1NGFlNzlkODc2ZDlkM2NiNTM.
CSeq: 2 REGISTER
Expires: 3600
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE, SUBSCRIBE
INFO
Supported: replaces
User-Agent: Bria Professional release 2.4 stamp 49381
Authorization: Digest username="antonioj",realm="172.30.6.99",nonce="5a2628
000000072dcb49aeaac113d86e266bb42287293e",uri="sip:172.30.6.99",response="5
7418d58c2d9cb896a89c0d930fb6c",algorithm=MD5
Content-Length: 0
```

Figura 20 – Pedido de registro com autorização e dados do usuário. Fonte: elaborado pelo autor.

Servidor valida o segundo pedido de registro e retorna com a resposta `200 - OK` conforme figura 21.

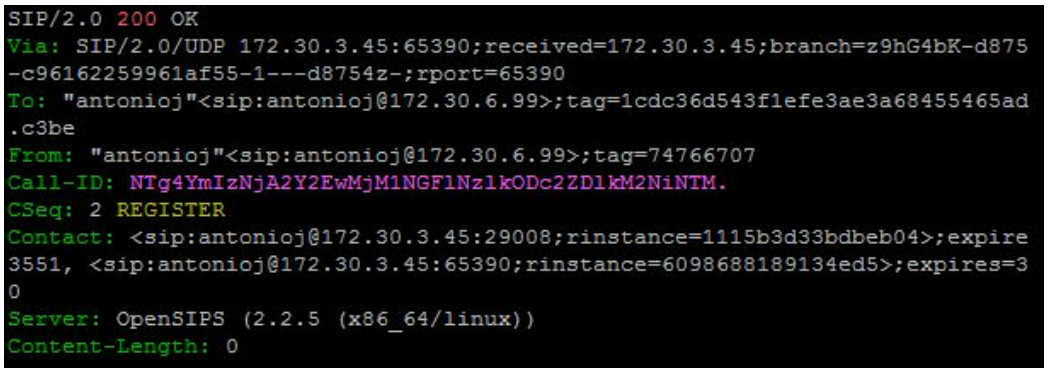

Figura 21 – Resposta `200 - OK`. Fonte: elaborado pelo autor.

No servidor de banco de dados o usuário é inserido na tabela de localização de usuário denominada `location` conforme tabela 3.

CAMPO	**DADO**
contact id	3828130052009099587
username	antonioj
domain	172.30.6.99
contact	sip:antonioj@172.30.3.45:65390;rinstance=6098688189134ed5
received	NULL
path	NULL
expires	2017-12-05 06:03:24
q	-1.00
callid	NTg4YmIzNjA2Y2EwMjM1NGF1Nz1k0Dc2ZD1kM2NiNTM.
cseq	2
last modified	2017-12-05 05:03:24
flags	0
cflags	
user agent	Bria Professional release 2.4 stamp 49381
socket	udp:172.17.0.10:5060
methods	5951
sip instance	NULL
attr	NULL

Tabela 3 – Tabela de localização de usuário. Fonte: elaborado pelo autor

4.5.2 Registro não autorizado

Ao tentar logar com um usuário existente na base LDAP porém enviado a senha incorreta recebemos o código SIP `401 - Unauthorized`.

Figura 22 – Diálogo SIP enviando senha incorreta. Fonte: elaborado pelo autor.

É enviado o pedido de registro conforme figura 23 solicitando o registro no servidor.

```
REGISTER sip:172.30.6.99 SIP/2.0
Via: SIP/2.0/UDP 172.30.3.45:60316;branch=z9hG4bK-d8754z-9702df4a982e964f-1
-d8754z-;rport
Max-Forwards: 70
Contact: <sip:antonioj@172.30.3.45:60316;rinstance=c379d6fe262ca359>
To: "antonioj"<sip:antonioj@172.30.6.99>
From: "antonioj"<sip:antonioj@172.30.6.99>;tag=ca05cc2c
Call-ID: OTEwNzI1MjlmZDY1MjdkMjVlMzBhMjg2M2M5ZjA2ZWI.
CSeq: 1 REGISTER
Expires: 3600
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE, SUBSCRIBE
INFO
Supported: replaces
User-Agent: Bria Professional release 2.4 stamp 49381
Content-Length: 0
```

Figura 23 – Requisição de registro de usuário. Fonte: elaborado pelo autor.

O servidor envia uma resposta não autorizado conforme figura 24 desafiando o usuário.

```
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP 172.30.3.45:60316;received=172.30.3.45;branch=z9hG4bK-d875
-9702df4a982e964f-1---d8754z-;rport=60316
To: "antonioj"<sip:antonioj@172.30.6.99>;tag=1cdc36d543f1efe3ae3a68455465ad
.060f
From: "antonioj"<sip:antonioj@172.30.6.99>;tag=ca05cc2c
Call-ID: OTEwNzI1MjlmZDY1MjdkMjVlMzBhMjg2M2M5ZjA2ZWI.
CSeq: 1 REGISTER
WWW-Authenticate: Digest realm="172.30.6.99", nonce="5a2628b0000000043c3222
605f59b8520d5ae971755657"
Server: OpenSIPS (2.2.5 (x86_64/linux))
Content-Length: 0
```

Figura 24 – Resposta não autorizado desafiando o usuário. Fonte: elaborado pelo autor.

É enviado o pedido de registro ao servidor conforme figura 25 com o nome de usuário correto porém a senha incorreta.

```
REGISTER sip:172.30.6.99 SIP/2.0
Via: SIP/2.0/UDP 172.30.3.45:60316;branch=z9hG4bK-d8754z-0e7d46188b237007-1
-d8754z-;rport
Max-Forwards: 70
Contact: <sip:antonioj@172.30.3.45:60316;rinstance=c379d6fe262ca359>
To: "antonioj"<sip:antonioj@172.30.6.99>
From: "antonioj"<sip:antonioj@172.30.6.99>;tag=ca05cc2c
Call-ID: OTEwNzI1MjlmZDY1MjdkMjVlMzBhMjg2M2M5ZjA2ZWI.
CSeq: 2 REGISTER
Expires: 3600
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE, SUBSCRIBE
INFO
Supported: replaces
User-Agent: Bria Professional release 2.4 stamp 49381
Authorization: Digest username="antonioj",realm="172.30.6.99",nonce="5a2628
000000043c32226f605f59b8520d5ae971755657",uri="sip:172.30.6.99",response="a
0c17dd64e70e267ea5b6217499724",algorithm=MD5
Content-Length: 0
```

Figura 25 – Pedido de registro com autorização e com senha incorreta. Fonte: elaborado pelo autor.

O servidor valida as credenciais e verifica que a senha está incorreta não autorizando o registro do usuário conforme figura 26.

```
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP 172.30.3.45:60316;received=172.30.3.45;branch=z9hG4bK-d875
-0e7d46188b237007-1---d8754z-;rport=60316
To: "antonioj"<sip:antonioj@172.30.6.99>;tag=1cdc36d543f1efe3ae3a68455465ad
.d4d4
From: "antonioj"<sip:antonioj@172.30.6.99>;tag=ca05cc2c
Call-ID: OTEwNzI1MjlmZDY1MjdkMjVlMzBhMjg2M2M5ZjA2ZWI.
CSeq: 2 REGISTER
WWW-Authenticate: Digest realm="172.30.6.99", nonce="5a2628b000000005992d09
ecbf80d4070f17d4e9ecba60"
Server: OpenSIPS (2.2.5 (x86_64/linux))
Content-Length: 0
```

Figura 26 – Não autorizado devido a senha estar incorreta. Fonte: elaborado pelo autor.

Usuário realiza mais um pedido de registro conforme figura 27 enviando novamente ao servidor o nome de usuário correto porém a senha incorreta.

```
REGISTER sip:172.30.6.99 SIP/2.0
Via: SIP/2.0/UDP 172.30.3.45:60316;branch=z9hG4bK-d8754z-7207282da1766120-1
-d8754z-;rport
Max-Forwards: 70
Contact: <sip:antonioj@172.30.3.45:60316;rinstance=c379d6fe262ca359>
To: "antonioj"<sip:antonioj@172.30.6.99>
From: "antonioj"<sip:antonioj@172.30.6.99>;tag=ca05cc2c
Call-ID: OTEwNzI1MjlmZDY1MjdkMjVlMzBhMjg2M2M5ZjA2ZWI.
CSeq: 3 REGISTER
Expires: 3600
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE, SUBSCRIBE
INFO
Supported: replaces
User-Agent: Bria Professional release 2.4 stamp 49381
Authorization: Digest username="antonioj",realm="172.30.6.99",nonce="5a2628
00000005992d0908ecbf80d4070f17d4e9ecba60",uri="sip:172.30.6.99",response="0
99435636e2fc2058ad5d6ce32060c",algorithm=MD5
Content-Length: 0
```

Figura 27 – Usuário envia novamente o pedido de registro com senha incorreta. Fonte: elaborado pelo autor.

O servidor valida as credenciais novamente e verifica que a senha está incorreta não autorizando o registro do usuário conforme figura 28.

```
SIP/2.0 401 Unauthorized
Via: SIP/2.0/UDP 172.30.3.45:60316;received=172.30.3.45;branch=z9hG4bK-d875
-7207282da1766120-1---d8754z-;rport=60316
To: "antonioj"<sip:antonioj@172.30.6.99>;tag=1cdc36d543f1efe3ae3a68455465ad
.1f6e
From: "antonioj"<sip:antonioj@172.30.6.99>;tag=ca05cc2c
Call-ID: OTEwNzI1MjlmZDY1MjdkMjVlMzBhMjg2M2M5ZjA2ZWI.
CSeq: 3 REGISTER
WWW-Authenticate: Digest realm="172.30.6.99", nonce="5a2628b000000006a234c4
cfb793001ea609059fb8ef96"
Server: OpenSIPS (2.2.5 (x86_64/linux))
Content-Length: 0
```

Figura 28 – Resposta não autorizado devido a senha continuar incorreta. Fonte: elaborado pelo autor.

4.5.3 Usuário não encontrado

Tentativa de registro com um usuário que não existe na base LDAP.

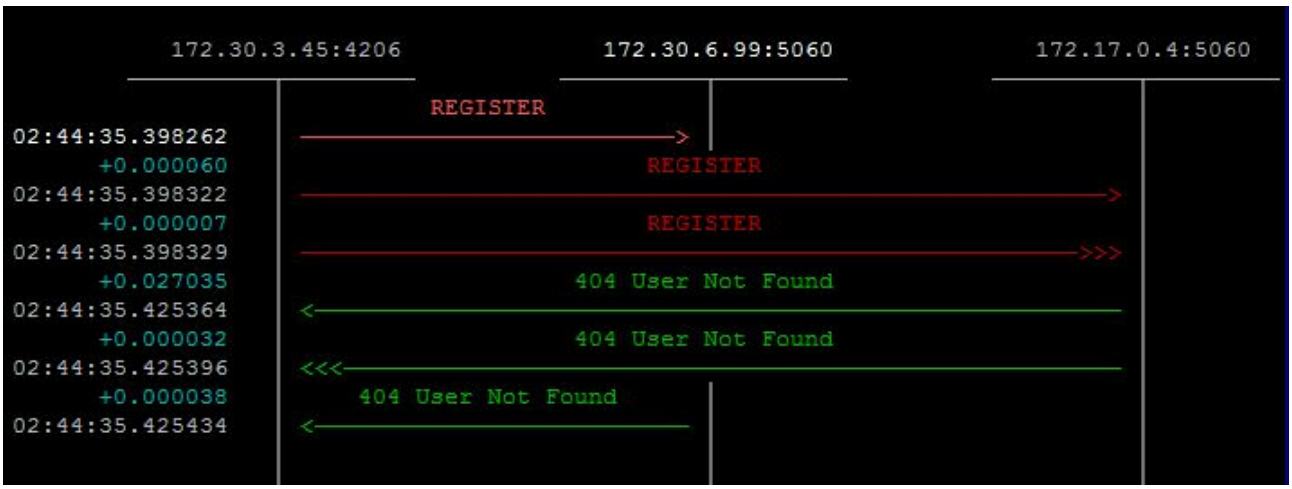

Figura 29 – Diálogo SIP de usuário não encontrado. Fonte: elaborado pelo autor.

É enviado o pedido de registro conforme figura 30 com o nome de usuário não existente na base LDAP para registro no servidor.

```
REGISTER sip:172.30.6.99 SIP/2.0
Via: SIP/2.0/UDP 172.30.3.45:4206;branch=z9hG4bK-d8754z-7706dd44eb15483c-1-
d8754z-;rport
Max-Forwards: 70
Contact: <sip:belmiro@172.30.3.45:4206;rinstance=e35b50fa94626353>
To: "belmiro"<sip:belmiro@172.30.6.99>
From: "belmiro"<sip:belmiro@172.30.6.99>;tag=6a6be853
Call-ID: MWVjYmU4MDY0ODZkNmQ1ZjRmZGRhMjZjZDdjNTA3Yjc.
CSeq: 1 REGISTER
Expires: 3600
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE, SUBSCRIBE
INFO
Supported: replaces
User-Agent: Bria Professional release 2.4 stamp 49381
Content-Length: 0
```

Figura 30 – Requisição de registro de usuário. Fonte: elaborado pelo autor.

O servidor verifica que o nome de usuário não existe na base LDAP retornando com a resposta de usuário não encontrado conforme figura 31.

```
SIP/2.0 404 User Not Found
Via: SIP/2.0/UDP 172.30.3.45:4206;received=172.30.3.45;branch=z9hG4bK-d8754
7706dd44eb15483c-1---d8754z-;rport=4206
To: "belmiro"<sip:belmiro@172.30.6.99>;tag=542ba1a617ac37899eeb925cca4f1639
ec2
From: "belmiro"<sip:belmiro@172.30.6.99>;tag=6a6be853
Call-ID: MWVjYmU4MDY0ODZkNmQ1ZjRmZGRhMjZjZDdjNTA3Yjc.
CSeq: 1 REGISTER
Server: OpenSIPS (2.2.5 (x86_64/linux))
Content-Length: 0
```

Figura 31 – Resposta do servidor de usuário não encontrado. Fonte: elaborado pelo autor.

5 Conclusões

Devido a falta de fóruns e documentação limitada do OpenSIPS foram necessários inúmeros testes para validar as funcionalidades até que estivessem prontas para o uso sem retorno de erro. O uso em contêineres facilitou bastante todos os testes pois é um processo rápido e prático de carregar a imagem criada junto dos arquivos de configuração do OpenSIPS.

A autenticação LDAP funcionou corretamente utilizando uma base de teste, porém ao utilizar a base LDAP do IFSC foi encontrado um problema ao consultar a senha e validar no módulo de autenticação devido a criptografia em `SHA` ou `SSHA` das senhas. Devido a falta de tempo hábil e complexidade não foi possível inferir no código do módulo a fim de suportar senhas `SHA` e `SSHA`. De qualquer forma, como haveria a necessidade de manutenção do código para versões futuras, sugere-se como solução de longo prazo a criação de um novo atributo para cada usuário a autenticar no servidor registrador.

Por se tratar de contêineres é comum cenários de oscilação onde são criados e recriados novos contêineres conforme a demanda. Foi obtido sucesso com os usuários de teste nestes cenários de oscilação. Os testes de carga não foram efetuados pois a nuvem do IFSC ainda não está homologada, no entanto está garantida a alta disponibilidade.

Como trabalhos futuros, sugere-se o contato com o mantenedor do OpenSIPS para o fork do módulo AUTH para o suporte ao registro de usuários de uma base LDAP que utiliza senhas criptografadas garantindo assim maior segurança. Esperam-se implementações para fazer uso do SIP como protocolo de sinalização para telefonia IP ou mesmo áudio e videoconferências, a exemplo de WebRTC que pode disponibilizar áudio e/ou vídeo para o ensino a distância em ambientes como moodle.

Referências

ALMEIDA, J. M. B. d. Container, o novo passo para a virtualização. *Technology Leadership Council Brazil*, n. 234, 2015. Citado 2 vezes nas páginas 38 e 39.

BERNERS-LEE, T.; FIELDING, R. T.; MASINTER, L. *Uniform Resource Identifiers (URI): Generic Syntax*. [S.l.], 1998. <http://www.rfc-editor.org/rfc/rfc2396.txt>. Disponível em: <http://www.rfc-editor.org/rfc/rfc2396.txt>. Citado na página 23.

BERNERS-LEE, T.; FIELDING, R. T.; MASINTER, L. *Uniform Resource Identifier (URI): Generic Syntax*. [S.l.], 2005. <http://www.rfc-editor.org/rfc/rfc3986.txt>. Disponível em: <http://www.rfc-editor.org/rfc/rfc3986.txt>. Citado na página 23.

FRANKS, J. et al. *HTTP Authentication: Basic and Digest Access Authentication*. [S.l.], 1999. <http://www.rfc-editor.org/rfc/rfc2617.txt>. Disponível em: <http://www.rfc-editor.org/rfc/rfc2617.txt>. Citado na página 52.

GONÇALVES, F. E. *Building Telephony Systems with OpenSIPS 1.6*: Build scalable and robust telephony systems using sip. [S.l.]: Packt Publishing Ltd., 2010. Citado na página 34.

HANDLEY, M. et al. *SIP: Session Initiation Protocol*. [S.l.], 1999. <http://www.rfc-editor.org/rfc/rfc2543.txt>. Disponível em: <http://www.rfc-editor.org/rfc/rfc2543.txt>. Citado na página 23.

JOHNSTON, A. et al. *Session Initiation Protocol (SIP) Basic Call Flow Examples*. [S.l.], 2003. Citado 4 vezes nas páginas 15, 26, 27 e 28.

OLIVEIRA, N. B. de. *Aplicações em Contêineres*. São José: Instituto Federal de Santa Catarina, 2017. Citado 4 vezes nas páginas 15, 21, 41 e 42.

REIS, R. B. *Teste de desempenho de sistemas de arquivos distribuídos em nuvem computacional privada no IFSC câmpus São José*. São José: Instituto Federal de Santa Catarina, 2017. Citado 2 vezes nas páginas 21 e 41.

ROSENBERG, J. et al. *SIP: Session Initiation Protocol*. [S.l.], 2002. <http://www.rfc-editor.org/rfc/rfc3261.txt>. Disponível em: <http://www.rfc-editor.org/rfc/rfc3261.txt>. Citado 2 vezes nas páginas 23 e 34.

ROSS, J. *VoIP*: Voz sobre ip. [S.l.]: Antenna Edições Técnicas Ltda., 2007. Citado na página 21.

SCHULZRINNE, H. et al. *RTP: A Transport Protocol for Real-Time Applications*. [S.l.], 2003. <http://www.rfc-editor.org/rfc/rfc3550.txt>. Disponível em: <http://www.rfc-editor.org/rfc/rfc3550.txt>. Citado na página 25.

Apêndices

APÊNDICE A – Repositórios da aplicação

A imagem criada utilizada pela aplicação foi criada e depositada no repositório da Docker Hub: <https://hub.docker.com/r/belmiroj/opensips/>.

Os arquivos de configuração `opensips.cfg`, `ldap.cfg` e de criação do banco de dados MySQL foram depositados no repositório do GitHub : <https://github.com/belmiroj/opensips>.

E, por fim, o repositório com os arquivos da aplicação rodando no Kubernetes do câmpus São José: <https://github.com/ctic-sje-ifsc/servicos_kubernetes/tree/2017.2/srv/ensino/TCC20706/belmiroj>.

APÊNDICE B – Arquivos de configuração

Dockerfile

```
FROM centos:centos6.9
MAINTAINER belmiro1991@gmail.com

RUN yum -y install git gcc gcc-c++ bison flex zlib-devel openssl-devel mysql mysql-server mysql-devel subversion pcre-devel ncurses-devel ncurses openldap openldap-devel

RUN git clone https://github.com/OpenSIPS/opensips.git -b 2.2 ~/opensips_2_2 && \
    sed -i 's/db_http db_mysql db_oracle/db_http db_oracle/g' ~/opensips_2_2/Makefile.conf.template && \
    sed -i 's/json ldap lua/json lua/g' ~/opensips_2_2/Makefile.conf.template && \
    cd ~/opensips_2_2 && \
    make all && make install && \
    cd .. && rm -rf ~/opensips_2_2

EXPOSE 5060/TCP
EXPOSE 5060/UDP

ENTRYPOINT ["/usr/local/sbin/opensips", "-D"]
```

Arquivo opensips.cfg

```
####### Global Parameters #########

log_level=6
log_stderror=no
log_facility=LOG_LOCAL0

children=4

/* uncomment the following line to enable debugging */
#debug_mode=yes

/* uncomment the next line to enable the auto temporary blacklisting of
   not available destinations (default disabled) */
#disable_dns_blacklist=no

```

```
16 /* uncomment the next line to enable IPv6 lookup after IPv4 dns
17    lookup failures (default disabled) */
18 #dns_try_ipv6=yes
19
20 /* comment the next line to enable the auto discovery of local
      aliases
21    based on revers DNS on IPs */
22 auto_aliases=no
23
24 listen=udp:eth0:5060   # CUSTOMIZE ME
25
26 ######## Modules Section ########
27
28 #set module path
29 mpath="/usr/local/lib64/opensips/modules/"
30
31 #### SIGNALING module
32 loadmodule "signaling.so"
33
34 #### StateLess module
35 loadmodule "sl.so"
36
37 #### Transaction Module
38 loadmodule "tm.so"
39 modparam("tm", "fr_timeout", 5)
40 modparam("tm", "fr_inv_timeout", 30)
41 modparam("tm", "restart_fr_on_each_reply", 0)
42 modparam("tm", "onreply_avp_mode", 1)
43
44 #### Record Route Module
45 loadmodule "rr.so"
46 /* do not append from tag to the RR (no need for this script) */
47 modparam("rr", "append_fromtag", 0)
48
49 #### MAX ForWarD module
50 loadmodule "maxfwd.so"
51
52 #### SIP MSG OPerationS module
53 loadmodule "sipmsgops.so"
54
55 #### FIFO Management Interface
56 loadmodule "mi_fifo.so"
57 modparam("mi_fifo", "fifo_name", "/tmp/opensips_fifo")
58 modparam("mi_fifo", "fifo_mode", 0666)
59
60 #### URI module
61 loadmodule "uri.so"
```

```
62 modparam("uri", "use_uri_table", 0)
63 modparam("uri", "db_url", "mysql://opensips:opensips@172.30.6.95/
       opensips")
64
65 #### LDAP Module
66 loadmodule "ldap.so"
67 modparam("ldap", "config_file", "/usr/local/etc/opensips/ldap.cfg")
68
69 ### MySQL Module
70 loadmodule "db_mysql.so"
71
72 #### USeR LOCation module
73 loadmodule "usrloc.so"
74 modparam("usrloc", "nat_bflag", "NAT")
75 modparam("usrloc", "db_mode",   2)
76 modparam("usrloc", "db_url", "mysql://opensips:opensips@172.30.6.95/
       opensips")
77
78 #### REGISTRAR module
79 loadmodule "registrar.so"
80
81 #### ACCounting module
82 loadmodule "acc.so"
83 /* what special events should be accounted ? */
84 modparam("acc", "early_media", 0)
85 modparam("acc", "report_cancels", 0)
86 /* by default we do not adjust the direct of the sequential requests.
87    if you enable this parameter, be sure the enable "append_fromtag"
88    in "rr" module */
89 modparam("acc", "detect_direction", 0)
90 modparam("acc", "db_url", "mysql://opensips:opensips@172.30.6.95/
       opensips")
91
92 #### UDP protocol
93 loadmodule "proto_udp.so"
94
95 #### Auth
96 loadmodule "auth.so"
97 modparam("auth", "nonce_expire",  30)
98 modparam("auth", "disable_nonce_check", 0)
99 modparam("auth","username_spec","$avp(54)")
100 modparam("auth","password_spec","$avp(55)")
101 modparam("auth","calculate_ha1",1)
102
103 ### Module Cachedb Local
104 loadmodule "cachedb_local.so"
105 modparam("cachedb_local", "cache_table_size", 20)
```

```
106 modparam("cachedb_local", "exec_threshold", 100000)
107 modparam("cachedb_local", "cache_clean_period", 1800)
108
109 #### DOMAIN module
110 loadmodule "domain.so"
111 modparam("domain", "db_url", "mysql://opensips:opensips@172.30.6.95/
    opensips")
112 modparam("domain", "db_mode", 1)   # Use caching
113 modparam("auth_db|usrloc|uri", "use_domain", 1)
114
115 ####  NAT modules
116 loadmodule "nathelper.so"
117 modparam("nathelper", "natping_interval", 10)
118 modparam("nathelper", "ping_nated_only", 1)
119 modparam("nathelper", "received_avp", "$avp(received_nh)")
120
121 #loadmodule "rtpproxy.so"
122 #modparam("rtpproxy", "rtpproxy_sock", "udp:localhost:12221")
123
124 #### Group module
125 loadmodule "group.so"
126 modparam("group", "db_url", "mysql://opensips:opensips@172.30.6.95/
    opensips")
127
128
129 ####### Routing Logic ########
130
131 # main request routing logic
132
133 route{
134         if (!mf_process_maxfwd_header("10")) {
135                 sl_send_reply("483","Too Many Hops");
136                 exit;
137         }
138
139         if (has_totag()) {
140                 # sequential requests within a dialog should
141                 # take the path determined by record-routing
142                 if (loose_route()) {
143
144                         if (is_method("BYE")) {
145                                 # do accunting, even if the
    transaction fails
146                                 do_accounting("log","failed");
147                         } else if (is_method("INVITE")) {
148                                 # even if in most of the cases is
    useless, do RR for
```

```
149                                 # re-INVITEs alos, as some buggy
    clients do change route set
150                                 # during the dialog.
151                                 record_route();
152                         }
153
154                         # route it out to whatever destination was
    set by loose_route()
155                         # in $du (destination URI).
156                         route(relay);
157                 } else {
158
159                         if ( is_method("ACK") ) {
160                                 if ( t_check_trans() ) {
161                                         # non loose-route, but
    stateful ACK; must be an ACK after
162                                         # a 487 or e.g. 404 from
    upstream server
163                                         t_relay();
164                                         exit;
165                                 } else {
166                                         # ACK without matching
    transaction ->
167                                         # ignore and discard
168                                         exit;
169                                 }
170                         }
171                         sl_send_reply("404","Not here");
172                 }
173                 exit;
174         }
175
176         # CANCEL processing
177         if (is_method("CANCEL"))
178         {
179                 if (t_check_trans())
180                         t_relay();
181                 exit;
182         }
183
184         t_check_trans();
185
186         if ( !(is_method("REGISTER")  ) ) {
187                 if (from_uri==myself)
188                 {
189                 } else {
190                         # if caller is not local, then called number
```

```
    must be local
191                         if (!uri==myself) {
192                                 send_reply("403","Rely forbidden");
193                                 exit;
194                         }
195                 }
196         }
197
198         # preloaded route checking
199         if (loose_route()) {
200                 xlog("L_ERR",
201                 "Attempt to route with preloaded Route's [$fu/$tu/$ru
    /$ci]");
202                 if (!is_method("ACK"))
203                         sl_send_reply("403","Preload Route denied");
204                 exit;
205         }
206
207         # record routing
208         if (!is_method("REGISTER|MESSAGE"))
209                 record_route();
210
211         # account only INVITEs
212         if (is_method("INVITE")) {
213                 do_accounting("log");
214         }
215
216         if (!uri==myself) {
217                 append_hf("P-hint: outbound\r\n");
218                 route(relay);
219         }
220
221         # requests for my domain
222         if (is_method("PUBLISH|SUBSCRIBE"))
223         {
224                 sl_send_reply("503", "Service Unavailable");
225                 exit;
226         }
227
228   # ldap search
229   if (!ldap_search("ldap://sipaccounts/ou=Users,dc=belmiro,dc=com??
     sub?(uid=$tU)"))
230   {
231         switch ($retcode)
232         {
233         case -1:
234             # no LDAP entry found
```

```
235             sl_send_reply("404", "User Not Found");
236             exit;
237         case -2:
238             # internal error
239             sl_send_reply("500", "Internal server error");
240             exit;
241         default:
242             exit;
243         }
244   }
245   xlog("L_INFO", "ldap_search: found [$retcode] entries for (uid=$tU)
       \n");
246
247
248   # save telephone number in $avp(tel_number)
249   ldap_result("telephoneNumber/$avp(tel_number)");
250   xlog("L_INFO", "Numero de telefone consulta LDAP: [$avp(tel_number)
       ]\n");
251
252   # save password in $avp(password)
253   ldap_result("userPassword/$avp(55)");
254   xlog("L_INFO", "Password consulta LDAP: [$avp(55)]\n");
255
256   if (method=="REGISTER") {
257     route(2);
258     exit;
259   }
260
261         if ($rU==NULL) {
262                 # request with no Username in RURI
263                 sl_send_reply("484","Address Incomplete");
264                 exit;
265         }
266
267         # do lookup with method filtering
268         if (!lookup("location","m")) {
269                 t_newtran();
270                 t_reply("404", "Not Found");
271                 exit;
272         }
273
274         # when routing via usrloc, log the missed calls also
275         do_accounting("log","missed");
276         route(relay);
277 }
278
279 route[2] {
```

```
280
281 #if (!(method=="REGISTER") && from_uri==myself) { /*no multidomain
    version*/
282         # are any credentials available in the request ?
283         if (!is_present_hf("Authorization")) {
284           www_challenge("", "0");
285           exit;
286         }
287
288   $avp(54) = $tU;
289         xlog("SCRIPT: stored user is $avp(54)\n");
290
291   #$avp(55) = $avp(password);
292         xlog("SCRIPT: stored password is $avp(55)\n");
293
294         if (!pv_www_authorize("")) {
295                 # authentication failed -> do challenge
296                 www_challenge("", "0");
297                 exit;
298         };
299         save("location");
300         exit;
301
302 }
303
304 route[relay] {
305         # for INVITEs enable some additional helper routes
306         if (is_method("INVITE")) {
307                 t_on_branch("per_branch_ops");
308                 t_on_reply("handle_nat");
309                 t_on_failure("missed_call");
310         }
311
312         if (!t_relay()) {
313                 send_reply("500","Internal Error");
314         };
315         exit;
316 }
317
318
319
320
321 branch_route[per_branch_ops] {
322         xlog("new branch at $ru\n");
323 }
324
325
```

```
326 onreply_route[handle_nat] {
327
328         xlog("incoming reply\n");
329 }
330
331
332 failure_route[missed_call] {
333         if (t_was_cancelled()) {
334                 exit;
335         }
336
337         # uncomment the following lines if you want to block client
338         # redirect based on 3xx replies.
339         ##if (t_check_status("3[0-9][0-9]")) {
340         ##t_reply("404","Not found");
341         ##      exit;
342         ##}
343
344 }
```

Arquivo ldap.cfg

```
1 [sipaccounts]
2 ldap_server_url            = "ldap://172.30.6.95"
3 ldap_bind_dn               = "cn=Manager,dc=belmiro,dc=com"
4 ldap_bind_password         = "PASSWD"
5 ldap_network_timeout       = 500
6 ldap_client_bind_timeout   = 500
```