Makiphone: um Aplicativo em React Native com o uso de WebRTC e Asterisk

Matheo Rodrigues Bonucia[1], **Joyce Aline de Oliveira Marins**[2], **Fabrício B. Carvalho**[1]

[1]Faculdade de Engenharia – Universidade Federal de Mato Grosso (UFMT)
78.060-900 – Várzea Grande – MT – Brasil

[2]Faculdade de Ciência e Tecnologia – Universidade Federal de Mato Grosso (UFMT)
78.060-900 – Várzea Grande – MT – Brasil

mattewbonucia@gmail.com, {joyce.marins,fabricio.carvalho}@ufmt.br

Abstract. *WebRTC is an emerging technology that uses real-time peer-to-peer (P2P) connections to stream media. This article presents Makiphone, a mobile application developed in React Native, which integrates WebRTC with the SIP protocol for audio and video calls. The solution uses an Asterisk server to manage media, deployed on an Amazon Web Services (AWS) platform, which allows for a scalable, stable and intuitive infrastructure. The results obtained demonstrate the feasibility and efficiency of integrating these technologies into real-time communication environments.*

Resumo. *WebRTC é uma tecnologia emergente que utiliza conexões peer-to-peer (P2P) em tempo real para transmissão de mídia. Este artigo apresenta o Makiphone, um aplicativo móvel desenvolvido em React Native, que integra WebRTC com o protocolo SIP para chamadas de áudio e vídeo. A solução utiliza um servidor Asterisk para gerenciar as mídias, implantado em uma plataforma Amazon Web Services (AWS), o que permite uma infraestrutura escalável, estável e intuitiva. Os resultados obtidos demonstram a viabilidade e eficiência da integração dessas tecnologias em ambientes de comunicação em tempo real.*

1. Introdução

A comunicação sempre foi um fator imprescindível para acelerar processos que ocorrem no cotidiano, principalmente no âmbito empresarial, gerando consequentemente maior rendimento e agilidade no que se diz respeito a resultados. Assim, a recente tecnologia Web Real-Time Communication (WebRTC), permite a realização de conexões peer-to-peer (p2p) simultâneas para transmissão de áudio, vídeo e troca de mensagens, sem a necessidade de servidores intermediários [Loreto and Romano 2014].

O WebRTC destaca-se por manter uma conexão direta entre dois *endpoints*, utilizando apenas um servidor WebSocket para sinalizar e gerenciar as etapas necessárias para estabelecer a comunicação [Suciu et al. 2020]. Esse processo difere de outros protocolos de comunicação, como o SIP (Session Initiation Protocol), que tradicionalmente utiliza servidores intermediários. O WebRTC combina diversos protocolos e APIs que permitem a transmissão de diferentes tipos de dados em tempo real, como áudio e vídeo, além de possibilitar o controle de recursos durante a conexão, abrindo um vasto leque de aplicações em potencial.

Além do WebRTC, o SIP é amplamente utilizado em telecomunicações para estabelecer chamadas por meio da tecnologia Voice Over IP (VoIP). O SIP utiliza elementos do HTTP e SMTP para facilitar a comunicação de voz e dados através da Internet, sendo uma solução robusta para a transmissão de chamadas por VoIP e a comunicação em ramais que suportam essa tecnologia [Pilon et al. 2014]. Ambos os protocolos são essenciais para viabilizar a integração de diferentes dispositivos em redes modernas.

Neste artigo, a tecnologia do WebRTC é explorada em um ambiente real, utilizando React Native para desenvolver Makiphone, um aplicativo móvel que possibilite a comunicação entre dois dispositivos conectados ao mesmo servidor. Para isso, foi necessário configurar um servidor Asterisk com suporte a SIP sobre WebSocket, permitindo não apenas a comunicação via WebRTC, mas também a interação com dispositivos compatíveis com o protocolo SIP. O servidor Asterisk gerencia o fluxo de voz e vídeo entre os dispositivos, enquanto o aplicativo controla as chamadas de forma eficiente e intuitiva.

As principais contribuições deste artigo são:

- Criação de um servidor SIP sobre WebSocket utilizando Asterisk;
- Desenvolvimento de uma interface em React Native;
- Configuração dos ouvintes responsáveis pela interação com o servidor, garantindo que as chamadas fossem gerenciadas corretamente;
- Realização de testes de desempenho e experiência dos usuários; e
- Desenvolvimento de Makiphone, um aplicativo que facilita a comunicação entre diferentes dispositivos.

2. Metodologia

Este artigo desenvolve Makiphone, uma solução eficiente para comunicação de voz e vídeo em ambientes empresariais, integrando WebRTC [WebRTC 2023] e Asterisk [Project 2024] em um aplicativo móvel React Native [Native 2024]. A implementação garante uma comunicação rápida e compatível com outras plataformas, superando desafios para oferecer uma experiência de uso fluida para os usuários.

Neste artigo, foram utilizados métodos de estudo de caso para implementação do projeto na prática e também a pesquisa experimental com foco em capturar o desempenho do aplicativo nos mais diversos âmbitos. Assim, para o desenvolvimento de Makiphone, foi realizada uma configuração inicial do servidor Asterisk com suporte WebSocket, criando ramais SIP e WebRTC. A biblioteca JsSIP [JsSIP 2024] foi utilizada para conectar o aplicativo ao Asterisk, organizando os fluxos de chamadas e simplificando o processo de comunicação. Após o estabelecimento da conexão, funcionalidades como realizar e receber chamadas, controlar recursos e gerenciar eventos foram implementadas. Para tal, diversos estudos de casos, pesquisa bibliográficas e pesquisas experimentais foram realizadas a fim de desenvolver com eficiência todo o arcabouço do Makiphone [Pizzani et al. 2012, Gil 2002].

O aplicativo foi desenvolvido com React Native, utilizando contextos para gerenciar chamadas e eventos de login de agentes. Interfaces intuitivas permitem que os usuários registrem seus ramais, configurem servidores e realizem chamadas, garantindo a persistência das variáveis e a continuidade das operações em todas as telas. O sistema ainda oferece uma sala de conferência em tempo real com controles como alternar câmera, silenciar áudio e utilizar funções DTMF (Dual Tone Multi Frequency).

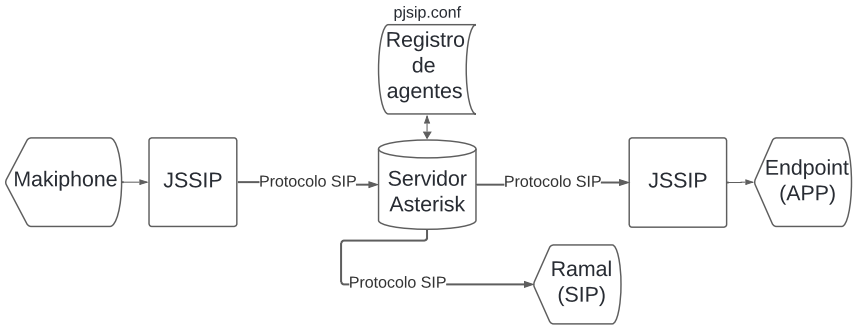

Figura 1. Visão geral de Makiphone.

A Figura 1 apresenta uma visão geral do Makiphone, explicitando o fluxo de dados em uma simples ligação. Assim, o servidor Asterisk armazena os registros de agentes, bem como gerencia as ligações para o aplicativo - um *endpoint* - por meio do protocolo SIP. Além disso, a interface do Makiphone envia constantemente eventos ao servidor, com o auxílio da biblioteca JSSIP, também via SIP.

3. Resultados

3.1. Interface e *Back-end*

O desenvolvimento do aplicativo Makiphone utiliza o React Native com a biblioteca React Navigation para facilitar a navegação e melhorar a experiência do usuário. A tela inicial permite realizar chamadas rapidamente, com o registro de ramal mantido no dispositivo de forma criptografada, eliminando a necessidade de novos registros após o primeiro acesso. A navegação entre a tela inicial e as configurações ocorre por meio de uma barra de navegação localizada na parte inferior do aplicativo, proporcionando fácil acesso às principais funcionalidades.

Ao iniciar uma ligação, o usuário é direcionado para a tela de saída da ligação, onde pode cancelar a chamada ou retornar à tela inicial. Caso a chamada seja aceita, ambos os usuários são direcionados para uma sala de conferência, onde ocorre a transmissão de mídia via WebRTC, possibilitando manipulação das chamadas em tempo real, por meio da API MediaStream. Além disso, o teclado DTMF também possibilita funções avançadas, como transferência de chamadas e navegação em Unidades de Resposta Audível (URA), que automatizam serviços por meio de comandos de voz ou dígitos numéricos [Corkrey and Parkinson 2002].

O *back-end* do aplicativo é dividido em três contextos: autenticação de usuários, gerenciamento de chamadas com WebRTC e um script principal que unifica essas funcionalidades. O gerenciamento de chamadas é feito pela biblioteca JsSIP, conectando o aplicativo ao servidor Asterisk via WebSocket e permitindo que o aplicativo funcione como um *endpoint* SIP. O contexto de autenticação utiliza dados criptografados armazenados no dispositivo e enviados ao servidor para registro via protocolo SIP.

A integração desses contextos no aplicativo centraliza estados e métodos reutilizáveis, permitindo que toda a aplicação funcione de forma coesa. A implementação de um servidor Asterisk na AWS permite a superação de barreiras de NAT, habilitando conexões WebRTC e SIP de qualquer localidade. O servidor gerencia o fluxo de mídia utilizando os servidores STUN e TURN, essenciais para identificar o IP público do usuário

e o tipo de NAT associado, otimizando a transmissão de áudio e vídeo entre os *endpoints*, conforme as configurações detalhadas no repositório do projeto a ser disponibilizado na versão final deste artigo.

Figura 2. Pacotes envolvidos na comunicação do aplicativo Makiphone

O protocolo SIP é utilizado para registrar o aplicativo no servidor e realizar chamadas. Durante o processo de registro, o aplicativo envia credenciais criptografadas para o servidor, que as autentica e retorna um código de autorização. A comunicação ocorre por meio de pacotes SIP, como REGISTER e INVITE, os quais são gerenciados pelo servidor Asterisk. A Figura 2 ilustra as trocas de mensagens para a comunicação entre dois dispositivos com as respectivas latências em cada etapa. As medições de tempo entre os pacotes indicam que a maior latência observada ocorre na conexão com o servidor AWS, resultando em um pequeno atraso na troca de pacotes durante uma chamada de 16 segundos.

Por fim, a integração das bibliotecas e a infraestrutura do servidor resultam em uma experiência de comunicação eficaz no aplicativo Makiphone. O sistema completo foi encapsulado em uma instância AWS, o que facilita a manutenção remota e permite o uso do aplicativo por usuários externos à rede do servidor. As funcionalidades avançadas, como a manipulação de chamadas e a segurança dos dados, tornam o Makiphone uma solução robusta para chamadas SIP via WebRTC.

3.2. Avaliação de Desempenho

Após diversas configurações no servidor, superando barreiras como NAT e outras dificuldades típicas, foi possível monitorar a qualidade das chamadas em diferentes cenários de rede. Esses ajustes permitiram que o servidor Asterisk, hospedado na AWS, lidasse eficientemente com os desafios usuais de chamadas VoIP, independentemente da localização dos usuários.

A Tabela 1 ilustra os resultados obtidos das médias com o intervalo de confiança de 95%, em que foram realizadas ligações de aproximadamente 16 segundos, 11 vezes.

Por meio do módulo *pjsip*, os dados de desempenho das chamadas foram extraídos utilizando o comando *"pjsip show channelstats"*, executado diretamente na CLI do Asterisk ou externamente via *bash* com *asterisk -rx*. Esse recurso viabilizou a realização

Tabela 1. Latências de uma ligação entre dois *endpoints*.

Evento	Tempo Decorrido (s)
Início da ligação (SDP)	$0,020 \pm 0,023$
Resposta sem autorização	$0,125 \pm 0,056$
Pacote ACK (Reconhecimento)	$0,138 \pm 0,011$
Pacote INVITE	$0,205 \pm 0,008$
Pacote Trying (Recebido INVITE e realizando conexão com servidor)	$2,363 \pm 0,221$
Pacote Ringing (Esperando outro lado atender)	$6,015 \pm 0,675$
Resposta OK (Solicitação atendida)	$6,193 \pm 0,041$
Pacote ACK (Estabeleceu chamada)	$23,612 \pm 2,707$
Pacote BYE (Outra ponta desligou a chamada)	$23,669 \pm 0,034$
Resposta OK (Avisa ao servidor que recebeu o pacote BYE)	$23,669 \pm 0$

de testes de conectividade e qualidade das chamadas. A Tabela 2 apresenta os resultados médios de envio e recebimento de mensagens com os intervalos de confiança de 95% de um total de 11 execuções independentes.

Tabela 2. Resultados com intervalo de confiança de 95% da transmissão e recebimento por meio das redes 3G, 5G e Wireless.

Rede	Transmitidos			Recebidos			Latência (ms)
	Qtde.	Perda	Jitter (ms)	Qtde.	Perda	Jitter (ms)	
3G	1476 ± 152	0	13,1	1231 ± 117	0	16,5	$67,5 \pm 2,2$
5G	828 ± 132	0	14,8	648 ± 177	0	13,5	$83 \pm 5,2$
Wireless	1246 ± 132	0	59	1196 ± 132	0	10	$38 \pm 3,1$

Vale ressaltar que esses testes foram realizados com áudio e vídeo em funcionamento em ambos os aparelhos. Portanto, em diferentes cenários, ficou visível que a implantação do Makiphone atendeu às expectativas de estabilidade e apresentou nenhuma perda de pacotes, o que gerou uma comunicação fluida durante as ligações.

3.3. Segurança

A habilitação de WebSocket seguro no servidor Asterisk depende da implementação de certificados SSL/TLS, que estabelecem uma conexão segura e criptografada entre o servidor e o *endpoint*. Para facilitar a geração desses certificados, ferramentas como o *Let's Encrypt* são amplamente utilizadas. No entanto, a obtenção de um certificado SSL/TLS requer a posse de um domínio registrado, o que representa um obstáculo financeiro para o projeto devido ao custo de aquisição do domínio.

Como alternativa temporária, o Makiphone adota o uso de WebSocket não seguro na porta 8088 do servidor Asterisk. Embora essa solução não ofereça o mesmo nível de segurança que o WebSocket seguro, ela permite a continuidade do desenvolvimento e o funcionamento da aplicação sem interrupções significativas. Entretanto, durante o desenvolvimento, foram implementadas todas as configurações necessárias para que o Makiphone se conecte a servidores com conexão criptografada, possibilitando que, com uma futura aquisição de domínio, o aplicativo já possa ser utilizado sem problemas.

Essa solução temporária, no entanto, não elimina a importância de garantir uma conexão segura. A implementação futura de WebSocket seguro permanece no planejamento do projeto, sendo vista como uma etapa crucial para fortalecer a segurança e a confiança no sistema.

Enquanto isso, o Makiphone continua a explorar opções de baixo custo e soluções alternativas para atender aos padrões de segurança exigidos pelas melhores práticas, visto que a não implementação de uma conexão criptografada pode se tornar um empecilho

para dispositivos mais restritivos em permissões e também expor a aplicação a possíveis ataques de Cross-Site WebSocket Hijacking (CSWSH), em que o invasor rouba a conexão WebSocket do usuário e consegue acessar suas informações. Assim, a transição para um ambiente mais seguro torna-se uma prioridade a ser explorada em trabalhos futuros, juntamente com novas funcionalidades, como agenda e histórico de chamadas.

4. Conclusão

Este artigo apresenta o Makiphone, um aplicativo desenvolvido para facilitar a comunicação entre celulares e ramais em ambientes corporativos. Embora a tecnologia WebRTC seja adequada para essa finalidade, sua principal limitação é o uso restrito a navegadores. Para superar esse problema, Makiphone envolve o uso de React Native em conjunto com WebRTC e JsSIP, transformando o celular em um *endpoint* capaz de se comunicar com o Asterisk via WebSocket. O principal desafio foi criar um fluxo unificado de controle de chamadas, mitigando a gestão de sessões entre dispositivos. No entanto, com o uso de CODECs modernos, o desenvolvimento de Makiphone supera essas dificuldades, permitindo a realização de chamadas com qualidade, segurança e eficiência.

O Makiphone atinge seus objetivos ao conectar dispositivos móveis e ramais por meio do Asterisk, configurado para garantir a estabilidade das chamadas. Android Package Kits (APKs) são gerados e distribuídos na versão *release* para os testes, permitindo a coleta de *feedback* durante todo o processo. A experiência adquirida inclui desde a configuração de servidores até a superação de barreiras de rede, como o NAT. Além disso, o Makiphone se mostra escalável, com o servidor Asterisk preparado para incorporar novas funcionalidades, estabelecendo uma base sólida para futuras expansões.

Referências

Corkrey, R. and Parkinson, L. (2002). Interactive voice response: Review of studies 1989–2000. *Behavior Research Methods, Instruments, & Computers*, 34(3):342–353.

Gil, A. C. (2002). *Como elaborar projetos de pesquisa*. Editora Atlas SA.

JsSIP (2024). JsSIP: The JavaScript Library. `https://jssip.net/`. Acessado em 06/10/24.

Loreto, S. and Romano, S. P. (2014). *Real-time communication with WebRTC: peer-to-peer in the browser*. "O'Reilly Media, Inc.".

Native, R. (2024). React native. `https://reactnative.dev/`. Acessado em 06/10/24.

Pilon, G. C. et al. (2014). *SIP: O protocolo para convergência nas redes de nova geração.* PhD thesis, Universidade de São Paulo.

Pizzani, L. et al. (2012). A arte da pesquisa bibliográfica na busca do conhecimento. *RDBCI: Revista Digital de Biblioteconomia e Ciência da Informação*.

Project, A. (2024). A Brief History of the Asterisk Project. `https://tinyurl.com/8dvefry7`. Acessado em 23/05/24.

Suciu, G. et al. (2020). Webrtc role in real-time communication and video conferencing. In *2020 Global Internet of Things Summit (GIoTS)*.

WebRTC (2023). Peer Connections. `https://webrtc.org/getting-started/peer-connections?hl=pt-br`. Acessado em 19/05/24.