UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Uma abordagem para o *live streaming* colaborativo de vídeo usando smartphones

VITÓRIA
2015

JUAN XABIER ESTEBAN DE AQUINO CALLES

Uma abordagem para o *live streaming* colaborativo de vídeo usando smartphones

Dissertação apresentada ao Mestrado de Informática do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Mestre em Informática.

Orientador: Prof. Dr. Celso Alberto Saibel Santos
Coorientadora: Profa. Dra. Roberta Lima Gomes

VITÓRIA

2015

TERMO DE APROVAÇÃO

JUAN XABIER ESTEBAN DE AQUINO CALLES

UMA ABORDAGEM PARA O LIVE STREAMING DE VÍDEO USANDO SMARTPHONES

Dissertação apresentada ao Mestrado de Informática do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Mestre em Informática.

_____

Prof. Dr. Celso Alberto Saibel Santos

_____

Profa. Dra. Roberta Lima Gomes

_____

Profa. Dra. Patrícia Dockhorn Costa

_____

Prof. Dr. Rodrigo Laiola Guimarães

AGRADECIMENTOS

Agradeço à minha família, aos meus orientadores e aos meus colegas de laboratório.
Agradeço à CAPES pelo apoio financeiro durante a realização deste trabalho e por possibilitar a participação no SBSC 2015.

RESUMO

Recentes acontecimentos ao redor do mundo, tais como a primavera árabe em 2010, o movimento 15-M na Espanha em 2011 e, no Brasil, as manifestações de junho em 2013, tornam evidente o descontentamento das populações com os modelos de governo vigentes. A adoção de mídias sociais como a principal ferramenta de divulgação, com destaque para o *live streaming* de vídeo a partir de smartphones tem caracterizado de maneira única esse tipo de movimento social. O *live streaming* traz um conjunto de desafios em seu processo de produção, os quais vão desde aspectos tecnológicos até a segurança dos indivíduos que realizam a captura de vídeo. Acreditamos que alguns desses desafios podem ser equacionados por meio de mecanismos que permitam a colaboração entre autores de transmissões (os chamados *streamers*). Por outro lado, percebemos que o atual modelo e plataformas de *live streaming* não tratam, de forma adequada, a produção colaborativa de vídeo. Dessa forma, neste trabalho propomos uma extensão para plataformas de *live streaming*, no qual, adicionando funcionalidades no processo de captura e transmissão, possibilita-se o *live streaming* colaborativo a partir de smartphones. A extensão tem como principais características o fornecimento e uso de informações de contexto, o agrupamento dinâmico e espontâneo de transmissões em um mesmo evento e, por último, a colaboração entre membros de um grupo a partir de um canal de comunicação em grupo e notificações em mapa. Resultados em experimentos realizados com um protótipo implementado mostram que as funcionalidades propostas geram baixo impacto em recursos de bateria e banda larga.

PALAVRAS-CHAVE

Live streaming de vídeo; produção colaborativa de vídeo; sistemas multimídia; sistemas colaborativos; mídias sociais.

ABSTRACT

Late events around the world, such as the Arab Spring in 2010, the 15-M movement in Spain (2011) and, in Brazil, the June 2013 protests, highlight the dissatisfaction from populations with present governments. The adoption of social media, particularly live video streaming from smartphone devices, has become a key feature for these social movements. Live video streaming provides a set of challenges during its production, which range from technology aspects to safety from individuals that are capturing video. We believe that some of these challenges can be addressed by features that allow the cooperation between different stream owners (called streamers). Moreover, we realize that present live video streaming models and platforms do not handle cooperative video production appropriately. Thus, in this dissertation we propose an extension for live video streaming platforms, in which by including features during the capture and streaming stages, cooperative live video streaming from smartphones is enabled. The extension features the usage of context data, dynamic and spontaneous grouping of streams within the same event and, at last, the cooperation between members of a same group through a common communication channel and map notifications. Results from conducted experiments, using an implemented prototype, suggest that the proposed features present low influence in battery and bandwidth consumption.

KEYWORDS

Live video streaming; cooperative video; multimedia systems; computer-supported cooperative work; social media.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

A evolução de tecnologias para smartphones e infraestrutura de banda larga têm promovido o crescimento do uso das chamadas mídias sociais, tais como redes sociais, microblogs, compartilhamento de fotos, compartilhamento de vídeos e transmissões ao vivo, ou *live streaming usando smartphones*. Neste trabalho, nosso interesse está exatamente nessa última forma de comunicação.

O chamado *live streaming* consiste em um processo que engloba as seguintes etapas: (i) captura de eventos com câmeras geralmente presentes em smartphones, *tablets* e *notebooks*; (ii) codificação e segmentação do conteúdo capturado em formatos de streaming; (iii) transmissão de vídeo para servidores de *streaming* responsáveis pela sua difusão; e finalmente, (iv) a reprodução de vídeo, realizada geralmente por meio de *plugins* de reprodução para *web browsers* [AUSTERBERRY 2005].

De forma geral, plataformas de *live streaming* fornecem todos os componentes de software necessários para a realização de transmissões ao vivo por meio de smartphones, incluindo um aplicativo mobile de captura, servidores de streaming e um *website* no qual espectadores podem navegar e encontrar *live streams* em andamento ou transmissões gravadas. As principais plataformas existentes no momento da escrita desta dissertação são Twitcasting [TWITCASTING 2014], Livestream [LIVESTREAM 2014], Ustream [USTREAM 2014] e Bambuser [BAMBUSER 2014].

O *live streaming* por meio das plataformas mencionadas tem sido amplamente usado com propósitos jornalísticos na cobertura de eventos de natureza social e política, o chamado *citizen journalism* [ALLAN 2007]. Alguns eventos de destaque com notada participação do público na divulgação de conteúdo foram a Primavera Árabe [BENGTSSON 2013], o Movimento 15-M [CAMUÑAS 2014] e os movimentos Occupy (como os protestos em Occupy Wall Street [COSTANZA-CHOCK 2012]).

No Brasil, durante as manifestações de Junho em 2013, destacamos a participação da chamada Mídia NINJA (Narrativas Independentes, Jornalismo e Ação) [NINJA 2015] na cobertura e transmissão ao vivo de protestos, manifestações e ações dos movimentos sociais e culturais em todo o país, equipados exclusivamente de smartphones com câmeras e acesso à Internet [ALMEIDA e EVANGELISTA 2013]. A Mídia NINJA é definida pelos seus membros como uma rede de comunicação descentralizada que produz e distribui conteúdo baseado em trabalho colaborativo e compartilhamento online.

O live streaming realizado a partir de smartphones apresenta uma série de desafios, incluindo limitações de recursos de hardware e infraestrutura, tais como pouca duração de bateria e variações na qualidade do sinal em redes móveis (como 3G e 4G) [HESSEL 2013]. Por outro lado, por tratarem-se de transmissões que ocorrem ao vivo, o processo de captura está sujeito a uma série de acontecimentos imprevisíveis e, muitas vezes, perigosos, tais como conflitos entre policiais e manifestantes [WERNECK e STURM 2013]. De forma oposta, ainda em relação à imprevisibilidade, transmissões ao vivo também podem sofrer com a falta de acontecimentos relevantes [JUHLIN et al. 2012].

Pode-se dizer que as atuais plataformas de *live streaming* ainda apresentam pouca preocupação com a produção colaborativa de vídeo, especialmente para transmissões ao vivo simultâneas. Autores de uma transmissão, aqui também chamados de *streamers*, mesmo sendo usuários de uma mesma plataforma de streaming, não são capazes de automaticamente identificar em tempo real outros autores que realizam uma transmissão simultânea sobre o mesmo evento. Além disso, apesar de geralmente ofertarem um meio de comunicação entre um *streamer* e os seus espectadores, comumente usando as redes sociais, não há forma direta de comunicação entre *streamers* que realizam a cobertura de um mesmo evento.

Acreditamos que, havendo a possibilidade de autores de transmissões se identificarem em um mesmo evento, seria possível, quando desejado, agrupar-se de forma voluntária e compartilhar informações de localização, condições de captura, entre outras informações de contexto [DEY e ABOWD 1999]. Principalmente, se

oferecidas funcionalidades que permitam a colaboração direta entre membros de um grupo, seria possível coordenar ações e cooperar em tempo de captura, promovendo assim o streaming colaborativo, cooperativo e coordenado, como visto em [JUHLIN et al. 2014].

Em uma transmissão colaborativa, *streamers* poderiam, por exemplo, informar aos demais membros de um grupo sobre problemas ocorridos, tais como o fim da bateria, brigas entre manifestantes e forças de ordem e/ou segurança, o surgimento de tropas de choque, etc. Principalmente, seria possível decidir estratégias conjuntas de cobertura por meio de um processo de comunicação contínua.

1.1 Justificativa

Dado o potencial do *live streaming* a partir de smartphones, esperamos que ao prover elementos que permitam a colaboração entre autores de transmissões (*streamers*) ampliem-se as possibilidades de uso dessa forma de mídia social. Em um cenário onde plataformas de *live streaming* ainda não abordam aspectos de colaboração entre *streamers* em um mesmo evento, a inclusão dessas funcionalidades oferece uma abordagem para solucionar problemas relacionados à limitação e indisponibilidade de recursos e à questão da imprevisibilidade durante transmissões ao vivo.

Além disso, também há grande interesse em dar suporte a coberturas de vídeo realizadas por grupos ou indivíduos independentes, particularmente aos produtores de conteúdo do chamado *citizen journalism* [ALLAN 2007], ou seja, não vinculados aos grandes veículos tradicionais de comunicação. Nos casos citados anteriormente, é evidente o seu potencial em divulgar informações complementares à mídia tradicional, ou inclusive informações não conhecidas ou não divulgadas por esses meios, seja por limitações ou dificuldade de acesso aos fatos, ou mesmo por omissão deliberada ao contrariar seus próprios interesses. Uma característica do grupo Mídia NINJA, por exemplo, é a sua aceitação por parte de manifestantes em Junho de 2013, e inclusive a colaboração com a concessão de entrevistas, etc. Por

outro lado, na mesma manifestação, percebeu-se simultaneamente a hostilidade contra repórteres de emissoras tradicionais como a Rede Globo [MEDEIROS 2013].

1.2 Objetivos

O objetivo deste trabalho é propor uma extensão para plataformas de *live streaming*, hoje baseadas em canais individuais e sem informações compartilhadas, que permita, a partir de smartphones, a produção colaborativa de transmissões de vídeo. A proposta descreve os componentes arquiteturais que possibilitam a colaboração, isto é, tudo aquilo que deveria ser acrescentado a plataformas de *live streaming*, tanto no lado do cliente (aplicativos mobile de captura e transmissão), quanto no lado servidor. Descrevemos também as entidades lógicas representadas em nossa proposta (compartilhamentos de contexto e grupos) e os seus ciclos de vida. Finalmente, descrevemos também um conjunto inicial de mensagens trocadas entre os componentes da arquitetura, as quais são responsáveis pela transição do estado das entidades de nossa proposta.

Em seguida, desejamos implementar um protótipo para experimentação dos conceitos e da viabilidade da extensão para streaming colaborativo. O protótipo compreende um aplicativo mobile que se sobrepõe a aplicativos de captura existentes, fornecendo, ao autor de uma transmissão, maior percepção de contexto, identificação e agrupamento com outras transmissões e comunicação entre membros de um mesmo grupo. O protótipo deve incluir também os componentes arquiteturais que, estando no mesmo nível funcional que servidores de streaming, são necessários para permitir a colaboração entre *streamers*. Os principais componentes são o mapeador de informações de contexto e grupos e os serviços que fazem a intermediação para a comunicação entre membros de um grupo.

1.3 Metodologia

A partir do cenário descrito, a extensão proposta deve considerar a identificação de transmissões, realização de agrupamentos e inclusão de mecanismos de colaboração no momento de captura, permitindo o *live streaming*

colaborativo e servindo como uma abordagem para as dificuldades encontradas em coberturas ao vivo por um grupo de usuários usando smartphones.

Consideramos, para a identificação de outras transmissões de um mesmo evento, o aproveitamento de informações de contexto, principalmente a geolocalização. Algumas plataformas, como Bambuser, exploram a localização geográfica de transmissões, mas apenas para fins de visualização aos espectadores. As informações que representam o presente contexto de uma transmissão devem ser apresentadas também para outros *streamers* por meio de metáforas visuais simples e funcionalidades conhecidas. Cada *streamer* obterá também maior percepção de seu próprio contexto, principalmente explorando a sua localização geográfica por meio do uso de mapas, disponibilizando ruas e pontos de referência próximos.

Ao detectar outras transmissões, *streamers* podem então decidir juntar-se a outros *streamers* de maneira espontânea, formando grupos colaborativos. Ainda, ao encontrar novos grupos, é possível que um *streamer* migre entre eles dinamicamente. Após agrupamento, *streamers* poderiam visualizar em mapa, além de a si mesmos, a localização geográfica dos demais membros do seu grupo, adquirindo uma visão global sobre como o evento está sendo coberto.

Finalmente, após os grupos terem sido formados, mecanismos de colaboração são disponibilizados entre seus membros. Um dos mecanismos previstos é a tradicional troca contínua de mensagens de texto, isto é, um chat entre membros do grupo. O outro mecanismo corresponde ao envio de notificações em um mapa compartilhado, as quais possuem semântica bem definida e são acompanhadas por informações de coordenadas geográficas. Por meio de notificações em mapa é possível informar, por exemplo, o surgimento de confrontos em coordenadas específicas do mapa durante uma transmissão. As zonas de confronto serão marcadas com um sinal de perigo no mapa associado ao live streaming colaborativo.

O protótipo que implementa a extensão proposta inclui as funcionalidades de identificação e compartilhamento de informações de contexto, agrupamento de

transmissões e colaboração entre membros de um mesmo grupo. O protótipo foi desenvolvido, no lado do cliente (*streamer*), para a plataforma Android e, para oferecer a possibilidade de comunicação em tempo real, foi adotado o uso do protocolo XMPP [XMPP 2015].

Uma característica relevante do protótipo é a sua capacidade de se sobrepor (*piggyback*) a aplicativos de captura existentes, pertencentes às principais plataformas de *live streaming*. Assim, ele se comporta de fato como uma extensão de funcionalidades existentes e poupando-nos do esforço de construir toda a infraestrutura para permitir o *live streaming*.

Por fim, os experimentos realizados fazem uma avaliação exploratória das funcionalidades propostas, além de estimar os custos que resultam da adição desta funcionalidade à uma aplicação de live streaming, tais como o consumo de bateria e atrasos de comunicação envolvendo as mensagens entre elementos de um grupo. Os resultados sugerem que as funcionalidades propostas, além de permitirem a colaboração entre participantes de uma transmissão ao vivo, apresentam baixo custo adicional tanto em consumo de bateria quanto no *delay* de comunicação.

1.4 Desafios

Podemos separar os principais desafios em técnicos e conceituais. Os desafios técnicos estão relacionados às limitações das tecnologias usadas para a implementação de um protótipo. Por outro lado, desafios conceituais são a respeito das funcionalidades que devem permitir a colaboração entre *streamers* em uma cobertura de vídeo ao vivo.

Um dos principais desafios técnicos está relacionado ao uso das tecnologias de banda larga para smartphones, que são caracterizadas, em determinadas localizações e circunstâncias, por alta intermitência e áreas de cobertura limitadas. Por exemplo, picos de tráfego de dados em cenários como manifestações públicas, onde há um elevado número de usuários, resultam em baixa qualidade de serviços de banda larga, comumente apresentando lentidão ou

inclusive impossibilidade de comunicação. Evidentemente, a nossa proposta tende a aumentar a quantidade de dados enviados e recebidos por aplicativos de captura, assim, temos como desafio a utilização adequada dos recursos de rede para que a inclusão de funcionalidades de colaboração compense e justifique no aumento de custo de troca de dados.

Outro fator técnico, relacionado ao uso de smartphones, diz respeito a limitações de energia. Logo, outro grande desafio é o uso adequado de recursos energéticos para que as novas funcionalidades sejam justificadas e não impactem negativamente no tempo de vida de transmissões em *live streaming*. As dimensões reduzidas de suas telas também surgem como desafio, limitando as operações de entrada e visualização de dados (I/O). Logo, a nossa proposta também tem que considerar a simplicidade de interfaces, sem deixar de fornecer as funcionalidades propostas em nosso modelo colaborativo de *live streaming*.

Um último desafio técnico é a necessidade de infraestrutura mínima para permitir o *live streaming* de vídeo a partir de smartphones, que vai desde escolha e configuração de servidores de streaming até a implementação e adequação de protocolos de *live streaming* para ambientes de programação mobile, tais como no Android. Em nosso protótipo, buscamos utilizar plataformas de *live streaming* já existentes (possibilitado pelo uso da API – *Application Programming Interface*, de serviços da plataforma Android), dessa forma evitamos grande parte de custos em infraestrutura. Entretanto, consideramos futuros esforços em obter uma própria infraestrutura de *live streaming*, dando-nos maior autonomia sobre o conteúdo gerado (para obter, por exemplo, maiores informações de contexto).

O desafio conceitual mais relevante é o de determinar a melhor maneira de promover a colaboração entre *streamers* a partir da oferta de “novos” mecanismos de comunicação/coordenação. Isso inclui, por exemplo, a identificação automática de transmissões próximas e sugestão de formação de grupo, a definição e escolha de mensagens de notificação em mapa propostas neste trabalho, formas de avaliação dos níveis de colaboração, etc.

Finalmente, em nossa proposta, os aspectos de compartilhamento de

contexto, agrupamento de fontes e colaboração entre membros de grupo ocorre de forma livre, espontânea e dinâmica. Assim, o principal desafio é fazer com que autores de transmissões de vídeo sintam-se estimulados a colaborar uns com os outros, com a finalidade de minimizar as limitações técnicas e o impacto de acontecimentos imprevisíveis, ao mesmo tempo em que se geram conteúdos integrados e informações mais contextualizadas para os seus espectadores.

1.5 Organização do trabalho

A sequência desta dissertação está organizada da seguinte maneira em 4 outros capítulos. No capítulo 2 serão apresentados alguns trabalhos relacionados à nossa proposta, principalmente por abordarem os temas de contexto, agrupamentos e colaboração, que possam ser aplicados ao *live streaming*. No capítulo 3 descrevemos a nossa proposta de extensão de *live streaming* colaborativo, incluindo o seu funcionamento, arquitetura e mensagens. No capítulo 4 descrevemos os aspectos mais relevantes na implementação do nosso protótipo e, por último, relatamos os experimentos realizados e descrevemos os resultados obtidos. Finalmente, no capítulo 5 fazemos as nossas considerações finais e indicamos alguns trabalhos futuros para a prosseguimento desta pesquisa.

2 TRABALHOS RELACIONADOS

Neste capítulo apresentamos alguns dos principais trabalhos relacionados à nossa proposta de *live streaming* colaborativo. Como mencionado anteriormente, propomos uma extensão para plataformas de *live streaming* em que, informações de contexto serão exploradas para identificar e agrupar participantes em um mesmo evento e, em seguida, membros de um mesmo grupo tornam-se capazes de colaborar ao fornecer-lhes funcionalidades de comunicação. Dessa forma, dentro dos trabalhos relacionados, primeiro expomos aqueles que abordam a exploração de informações de contexto e agrupamento de transmissões e, em segundo lugar, trabalhos que lidam com a colaboração em tempo de captura.

2.1 Exploração de contexto e agrupamento

Dentre as atuais plataformas de live streaming, Bambuser [BAMBUSER 2014], e mais recentemente Periscope [PERISCOPE 2015], estão entre os que mais exploram dados de contexto da situação de captura. Ambos apontam a localização geográfica de dispositivos smartphones realizando transmissões (fontes) e apresentam as localizações dessas fontes em um mapa. Particularmente, Bambuser oferece um mapa no qual é possível visualizar as transmissões mais recentes de acordo com filtros temporais, tais como transmissões ao vivo naquele instante, transmissões que estiveram ao vivo 24 horas atrás, dois dias atrás e assim por diante. A Figura 1 mostra a visualização geográfica e temporal oferecida por Bambuser, a cor dos ícones muda de acordo com a data de cada transmissão. Além disso, dados de localização geográfica também são usados para recomendar transmissões geograficamente próximas, tanto ao vivo quanto conteúdo já salvo, embora a distancia que define a proximidade não esteja especificada, de fato retornando todas as transmissões do mesmo país ou região.

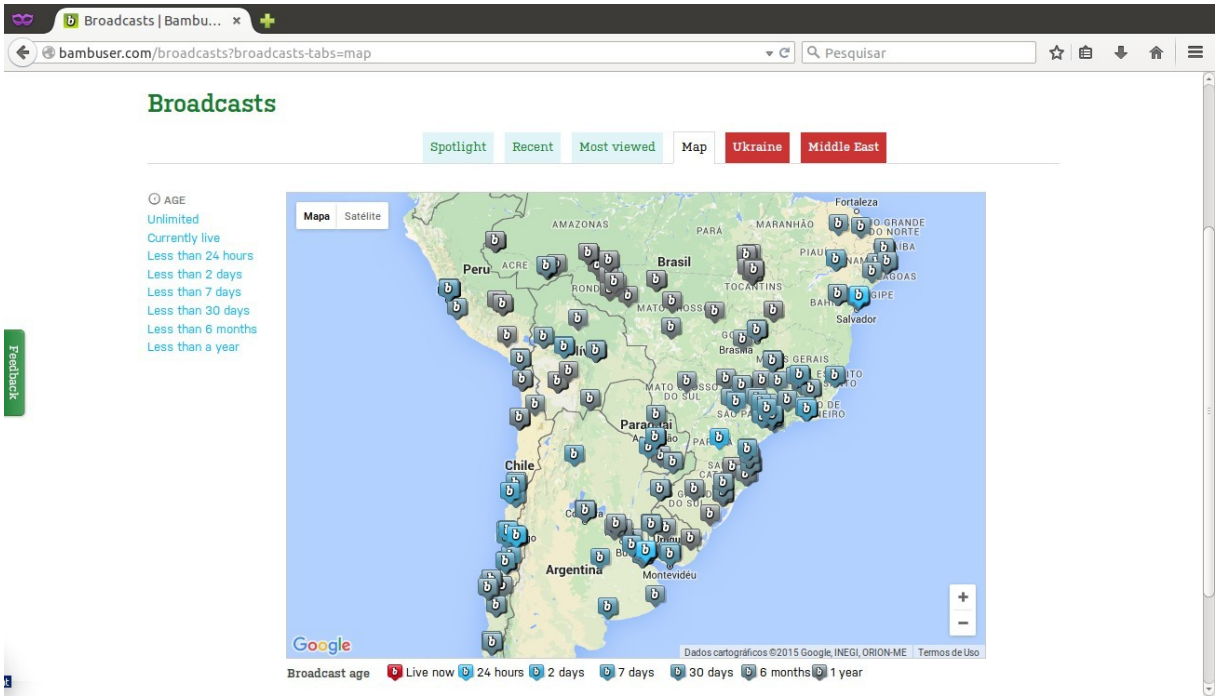

Figura 1: Visualização geográfica e temporal de streams

Apesar dessas abordagens explorarem os dados de localização por meio de mapas, essas funcionalidades não são estendidas para aplicativos mobile de captura. Isto permitiria, por exemplo, que *streamers* ganhassem maior percepção da sua localização exata, pontos de referência próximos (ruas, parques e prédios) e, principalmente, identificar outras transmissões nas redondezas.

O projeto IStream [BODDHU et al. 2012] também propõe o compartilhamento da localização geográfica enquanto se mantêm uma transmissão de vídeo ao vivo, principalmente para aspectos de segurança, em que *streamers* são militares ou policiais munidos de um aparelho iPhone. Nesse caso particular, o conteúdo de vídeo e as coordenadas GPS são enviados a centros de comando, com o objetivo de obter maior percepção do contexto no qual está sendo realizada a transmissão e acompanhar o pessoal em campo, mostrando a sua localização exata. Entretanto, esse trabalho não especifica como ocorrem as interações subsequentes, entre o centro de comando e as instâncias do IStream, ou ainda se *streamers* são capazes de interagir uns com outros diretamente.

FOCUS [JAIN et al. 2013] é um sistema destinado, de fato, a agrupar transmissões de vídeo ao vivo, baseado em dados de geolocalização e localização relativa e técnicas de visão computacional. Ao usar técnicas de visão computacional, FOCUS é capaz de deduzir similaridade de conteúdo mesmo quando vídeos são capturados em diferentes perspectivas, inferindo sobre a localização e orientação relativas de câmeras de dois ou mais *streamers*. Dados de sensores enviados junto ao conteúdo de vídeo incluem coordenadas geográficas de GPS, compasso, acelerômetro e giroscópio. Traçando a linha de visão de cada streamer, e identificando interseções geométricas, FOCUS é capaz de inferir similaridade de conteúdo mesmo quando o próprio vídeo contém pouca ou nenhuma similaridade visual.

FOCUS usa uma técnica de visão computacional chamada de *Structure From Motion* (SFM), ou Estrutura a partir de Movimento, e, para cada transmissão, desenvolve um modelo 3D estimado a partir da linha de visão de cada usuário. Em seguida, compara pares de transmissões, *frame* a *frame*, buscando por similaridade de linha de visão. Finalmente, comparando múltiplas transmissões e múltiplos *frames*, uma matriz espaço-temporal de similaridade de conteúdo é construída e, aplicando técnicas de agrupamento (*clustering*), retorna grupos com objetos de cena em comum. Assim, FOCUS vai além da identificação de transmissões em um mesmo evento, sendo capaz de determinar subconjuntos de transmissões que capturaram o mesmo objeto durante o evento. Entretanto, não são apresentados aspectos de colaboração entre *streamers* ou inclusive percepção de contexto, uma vez que grupos são detectados.

A formação de grupos também é abordada em [BOIX et al. 2011] tendo como base um conceito chamado de Flocks. Os Flocks são grupos dinâmicos de usuários que são formados seguindo descrições de alto nível, tais como “está cursando ensino superior” e “está perto”, que podem ser combinados, por exemplo, para retornar estudantes universitários que se encontrem nas proximidades.

Inicialmente, participantes devem providenciar as informações dos seus perfis para então definirem indeterminados conjuntos de descrições em alto nível,

cada um correspondendo a um Flock. O *matching* é realizado ao comparar a descrição de um Flock com as informações de perfis de usuários identificados. Usuários são identificados por meio de tecnologia *Wi-Fi direct*. Flocks são gerenciados pelo framework Urbiflock [BOIX et al. 2011].

O *Group Context Framework* (GCF) [DE FREITAS e DEY 2015] também permite que dispositivos mobile formem grupos, em que a identificação de participantes e comunicação ocorre por meio de tecnologia Bluetooth. O principal objetivo do framework é o compartilhamento de informações de contexto. Sendo assim, cada aplicação que usa o framework faz subscrições a determinados tipos de informações de contexto que possam ser disponibilizadas, tais como localização geográfica. Para cada instância GCF, os dados de sensores que serão disponibilizados são gerenciados por chamados *context providers*, correspondentes a cada tipo de dado (localização geográfica, acelerômetro, giroscópio, etc.). As informações de contexto, recebidas de outros dispositivos, são gerenciadas por chamados *arbiters*.

Ambos Flocks e grupos GCF são de tipo *ad hoc* e não são representados globalmente, portanto cada participante vê e gerencia os seus próprios grupos. Apesar de apresentarem uma abordagem para o agrupamento de *streamers*, uma representação global e centralizada simplificaria o gerenciamento e a visualização de grupos, além de facilitar a execução de rotinas no *server-side*, tais como o monitoramento de recursos.

Outras abordagens para agrupar vídeo ao vivo vêm de *websites* que explicitamente agregam *feeds* provenientes das principais plataformas de *live streaming*. Um deles é o OccupyStreams [OCCUPYSTREAMS 2014], que agrega canais dedicados à transmissão de conteúdo social e/ou político, mas principalmente àqueles relacionados a transmissões do chamado *Occupy movement* [COSTANZA-CHOCK 2012], vide o seu nome. A iniciativa apenas aceita a inscrição de canais provenientes das plataformas Livestream e Ustream.

Outro *website* similar é o Web Realidade [WEBREALIDADE 2014], dedicado especificamente a canais brasileiros que usam a plataforma Twitcasting. Em ambos

casos, OccupyStreams e Web Realidade, cada *streamer* faz o seu cadastro no *website* de forma manual. Em Web Realidade canais são organizados por estado, informação fornecida quando um usuário se inscreve. Assim, para cada estado, as transmissões são apresentadas em uma *grid* e, ao selecionar uma em particular, surge um *player* de vídeo reproduzindo a transmissão escolhida.

2.2 Colaboração em tempo de captura

Uma funcionalidade que todas as plataformas de *live streaming* têm em comum é a possibilidade de *streamers* interagirem, em tempo real, com os seus espectadores por meio de troca de mensagens de texto, as quais são geralmente replicadas em redes sociais (principalmente Twitter). Dessa forma espectadores podem colaborar com uma transmissão, ao fornecer sugestões ou inclusive informações relevantes ao autor de uma transmissão. Ustream [USTREAM 2014] inclusive permite que um *streamer* faça perguntas do tipo “sim” ou “não” aos seus espectadores. Apesar disso, não há qualquer funcionalidade própria que permita *streamers* de se comunicar entre si.

O Twitcasting [TWITCASTING 2014] oferece uma funcionalidade para a realização de uma transmissão em um grupo, de até cinco membros, chamada de Collabo. Entretanto, o principal objetivo do Collabo é aumentar a capacidade de participação dos espectadores. De fato, a sua forma parte de um *streamer* que habilita a participação de seus espectadores e, uma vez habilitado, até cinco espectadores podem iniciar o Collabo, compartilhando eles também a imagem de suas câmeras. Assim, Collabo mostra-se limitado no agrupamento e realização de *live streaming* colaborativo entre vários *streamers* em um mesmo evento (mais ainda quando os *streamers* não se conhecem).

Ainda, o aplicativo de captura Twitcasting permite que um *streamer* navegue por outras transmissões, e inclusive deixe comentários enquanto mantêm a sua própria transmissão. Porém, essa busca e visualização é realizada da mesma forma que com espectadores, na qual o acesso e a interação ocorre por meio da mesma interface web que é disponibilizada para espectadores. Assim, um autor de uma

transmissão não é capaz de diferenciar se quem interage com ele é mais um espectador ou outro streamer. Dessa forma, a interação se mantém como um canal de comunicação apenas entre um *streamers* e os seus espectadores.

O sistema Caleido, descrito por [SA et al. 2014], tem como principal objetivo obter a maior quantidade de mídia de alta qualidade durante eventos. Ele permite, de fato, que usuários realizando transmissões de vídeo sejam capazes de se perceberem, possibilitando que um certo grau de coordenação seja implementado. A principal funcionalidade do Caleido é a capacidade de perceber outras coberturas de vídeo sendo realizadas no mesmo evento, por meio do uso de dados de geolocalização e posicionamento relativo, que são transmitidos junto ao conteúdo multimídia. Ele usa uma simples arquitetura cliente-servidor, na qual o servidor armazena os dados transmitidos e automaticamente agrupa dispositivos mobile transmitindo próximos entre si.

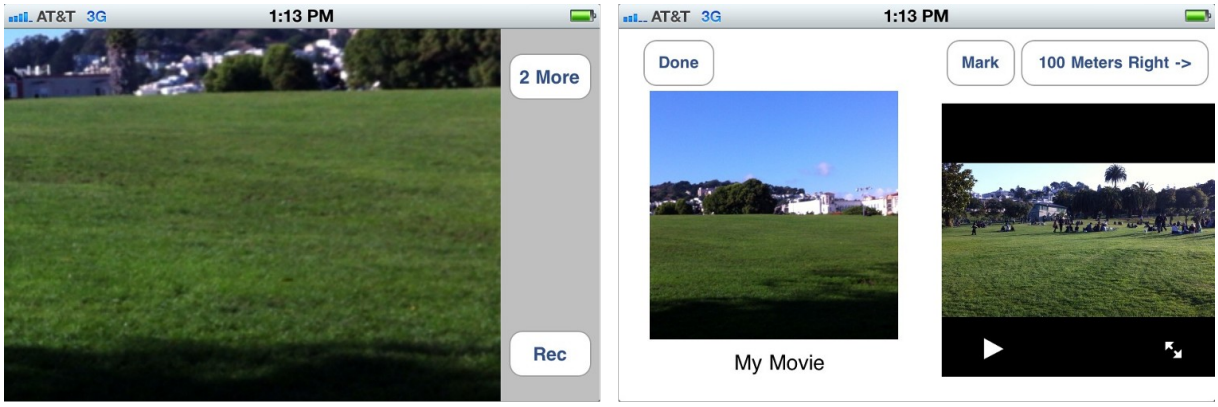

Figura 2: Protótipo do sistema Caleido

O aplicativo mobile oferece, em sua tela principal (Figura 2.a), uma *preview* típica do que é capturado pela câmera e uma barra lateral que inicialmente mostra o número de dispositivos de captura próximos. Uma tela (Figura 2.b) secundária mostra a transmissão de vídeo de um participante próximo e um botão que permite navegar pelas transmissões de todos os outros participantes. O botão de navegação é etiquetado com a posição relativa do próximo *streamer* (por exemplo,

“100 metros à direita”).

Ao visualizar transmissões de outros participantes em tempo real, o Caleido fornece uma visão mais ampla de como está ocorrendo a cobertura de um evento. Isto é útil, por exemplo, para reduzir a quantidade de conteúdo redundante, pois uma pessoa pode detectar que vídeo de maior qualidade já está sendo capturado a partir da mesma localização ou ângulo de visão. Outra funcionalidade oferecida é a possibilidade de que os vídeos de outros *streamers* sejam avaliados. Ao finalizar a transmissão de um evento, o servidor automaticamente toma os vídeos mais populares, isto é, os que possuem maior número de sinalizações positivas, e gera um resumo do evento a partir deles. Dessa forma, Caleido promove o *live streaming* colaborativo, ao aumentar a percepção de contexto, embora limitada a descrever textualmente o posicionamento relativo de outros participantes. Isso faz com que a colaboração seja indireta, pois além de permitir uma forma de *feedback* ao sinalizar outros *streamers*, participantes não se comunicam e interagem diretamente.

[PROCYK et al. 2014] propõem o compartilhamento de vídeo para cumprir tarefas colaborativas, especificamente o *geocaching*, uma atividade ao ar livre parecida com o pique-esconde usando o GPS. A colaboração ocorre apenas entre dois indivíduos, cada um equipado com um protótipo construído a partir de *hardware* e *software* comerciais, consistindo basicamente de uma câmera anexada a um par de óculos de sol e um smartphone anexado a uma munhequeira, ambos conectados por meio de Bluetooth.

A câmera anexada aos óculos de sol tem o objetivo de capturar o ponto de vista de um dos participantes, o qual é transmitido para o seu smartphone (através de Bluetooth) e, em seguida, enviado diretamente para o smartphone no outro extremo da aplicação (pertencente ao outro participante). Dessa forma, cada um assiste o ponto de visão do outro no seu smartphone.

Esta abordagem usa o vídeo como um meio para completar uma tarefa (*geocaching*), ao invés de ser ele próprio o objetivo, embora, como visto em Caleido [SA et al. 2014], o compartilhamento de *previews* de vídeo permita a produção colaborativa de vídeo, mesmo que indiretamente. Áudio também é capturado e

transmitido, assim um par de usuários pode conversar entre si durante toda a experiência, efetivamente permitindo uma maneira direta de comunicação, que vai além da sinalização positiva de vídeos em Caleido. Pares de participantes na atividade de *geocaching* são predefinidos e a sua correspondência ou formação não é mencionada.

Outra contribuição relacionada ao *live streaming* colaborativo está ligada a ferramentas de edição e mixagem ao vivo. O *Instant Broadcasting System* (IBS) [ENGSTROM et al. 2012] é um sistema que suporta até cinco *feeds* de vídeo ao vívo, provenientes de aplicativos de captura mobile. O usuário, chamado de diretor, pode assistir simultaneamente todas as cinco transmissões ao vivo e, entre elas, escolher uma para entrar “no ar”.

Figura 3: Visualização de um diretor no Instant Broadcasting System

Quando um *feed* em particular é selecionado, o seu autor é notificado por meio de um canal de comunicação de retorno. Mais importante ainda, esse canal de retorno permite que o diretor envie mensagens de texto para os *streamers*, de fato permitindo a coordenação ao vivo, apesar de centralizada e unidirecional (de diretor para *streamers*).

Um ponto interessante a respeito dos experimentos realizados por

[ENGSTROM et al. 2012], é o fato de que mensagens de texto tenderam a desviar a atenção de *streamers* no momento de captura, percebido na qualidade do conteúdo de vídeo. Entretanto, mostrou-se útil no planejamento e coordenação de ações antes do instante de captura, ou durante intervalos. Outro ponto em destaque é, novamente, o fato de que ao aumentar a percepção de contexto pode-se reduzir a redundância de conteúdos, pois o diretor pode informar a dois *streamers* que ambos estão capturando o mesmo objeto.

O Mobile Vision Mixer (MVM) [ENGSTROM et al. 2012B] é outra ferramenta de mixagem voltada exclusivamente para dispositivos mobile, tanto os dispositivos de captura quanto a ferramenta do diretor. Uma característica importante dela é o aproveitamento da plataforma Bambuser, usando o seu aplicativo de captura e os servidores de streaming. A ferramenta de mixagem obtém até quatro *feeds* nos servidores do Bambuser e exibe-os ao diretor que, em seguida, pode selecionar um entre eles para ser transmitido ao público, novamente usando a infraestrutura Bambuser. Aspectos de colaboração direta não são abordados.

Jogos do tipo *massive multiplayer online* (MMO) também podem contribuir para o *live streaming* colaborativo, com destaque a jogos que apresentam cenários competitivos. Ambos MMOs e *live streaming* colaborativo tem que lidar com protagonistas remotos, ou *players*, que estão agrupados logicamente e precisam, juntos, atingir um objetivo coletivo. Para jogos *multiplayer*, o objetivo é geralmentereconhecimento vencer outros grupos. No *live streaming* colaborativo, o objetivo pode ser interpretado como a realização de uma cobertura de vídeo mais ampla e duradoura sobre um evento.

O trabalho de [MONTANE-JIMENEZ 2014] traz um estudo sobre as funcionalidades de colaboração mais comuns em jogos *multiplayer*, que incluem comunicação em tempo real por meio de texto e voz. Além disso, propõe um *framework* para automaticamente disparar uma interação de acordo com o contexto de um *player*.

Alguns gêneros de jogos *multiplayer*, tais como os *first person shooter* (FPS) e *multiplayer online battle arena* (MOBA), fornecem um mapa ou radar onde é

possível localizar a si mesmo, colegas de time e inimigos. Particularmente, jogos do tipo MOBA permitem interações por meio do mapa [JORVID 2014], tais como notificar aliados (membros do mesmo grupo) sobre o desaparecimento de inimigos ou a localização dos próximos alvos. Estes tipos de funcionalidades servem de base para a colaboração em nossa extensão de *live streaming* colaborativo.

2.3 Considerações

Os trabalhos citados no decorrer deste capítulo contribuem em diferentes aspectos do *live streaming* colaborativo. Dentre estes aspectos podem ser incluídos o fornecimento e percepção de contexto, o agrupamento de transmissões baseada no seu contexto e, finalmente, a colaboração entre membros de um grupo. Sendo assim, os trabalhos apresentam elementos semelhantes aos da nossa proposta de extensão para o *live streaming* colaborativo.

Destacamos, dentre os trabalhos relacionados, as principais contribuições para cada um dos três aspectos explorados em nossa proposta:

1. **Fornecimento e percepção de contexto e agrupamento de autores de transmissões:** O uso de mapas em Bambuser e IStream [BODDHU et al. 2012], em que espectadores identificam a localização de autores de transmissões, será aplicado para que *streamers* sejam capazes de identificar uns aos outros. O mapa também auxiliará usuários a identificar pontos de referência próximos (ruas, avenidas, parques, etc.), fornecendo-lhes detalhes de sua atual localização geográfica.

   Grupos serão formados a partir da identificação de participantes em um mesmo contexto, principalmente a sua localização geográfica, tal como em Caleido [SA et al. 2014] e FOCUS [JAIN et al. 2013]. Em ambos, as informações de contexto são enviadas pelos aplicativos de captura para um servidor que, em FOCUS, automaticamente forma os grupos de acordo com as coordenadas geográficas e técnicas de visão computacional, e em Caleido, aponta a um autor de transmissão a

posição relativa dos demais participantes. Como será visto à frente, em nossa proposta grupos não são formados automaticamente, mas o agrupamento é sugerido aos participantes.

O agrupamento por geolocalização também é usado em Flocks [BOIX et al. 2011] e no *framework* GCF [DE FREITAS e DEY 2015], entretanto, como mencionado anteriormente, nenhum deles fornece uma representação global sobre grupos. Dessa forma, cada *streamer* lida com uma representação própria sobre os membros de um grupo. Isto faz com que, por exemplo, a execução de rotinas de monitoramento não seja tão intuitiva, já que não há uma representação centralizada sobre a composição de um grupo.

Uma vez identificados os demais participantes, o Caleido permite que os membros de um grupo sejam capazes de perceber-se entre si, descrevendo a sua posição relativa ao informar direção (por exemplo, direita ou esquerda) e distância (por exemplo, 10 metros). Em nossa proposta, membros de um grupo percebem-se entre si usando mapas, apontando precisamente a localização geográfica de cada um deles.

2. **Streaming colaborativo:** Em termos efetivamente de colaboração temos que as principais plataformas de *live streaming* permitem que espectadores forneçam algum tipo de apoio a autores de transmissões por meio de um canal de comunicação para *feedback*. Entretanto, as ferramentas atuais permitem apenas a comunicação entre um *streamer* e os seus espectadores.

   Embora o Twitcasting permita, no aplicativo de captura, navegar no seu website, visualizar uma transmissão ao vivo e interagir com o seu autor, esta interação faz-se de mesma maneira que um típico espectador. Dessa forma, para um streamer, todos os indivíduos que interagem com ele são considerados espectadores.

   O Instant Broadcasting System (IBS) [ENGSTROM et al. 2012] permite

a comunicação entre o chamado “diretor” (quem recebe vários *feeds* de vídeo e seleciona um para entrar “ao ar”) e autores de transmissões, entretanto, ela ocorre apenas no sentido diretor-streamer. Isto faz com que a coordenação, ou a tomada de decisões em relação à cobertura de um evento, seja centralizada. Em nossa proposta, todos os autores de transmissões pertencentes ao mesmo grupo serão capazes de se comunicar entre si, permitindo que decisões (ex. o quê capturar, evitar situações perigosas, compartilhar recursos, etc.) sejam tomadas em conjunto.

Notificações em mapa similares às usadas atualmente jogos digitais, apresentadas no estudo de [JORVID 2014], serão incluídas em nossa extensão de streaming colaborativo. Por meio delas, *streamers* serão capazes de, em tempo real, notificar aos colegas de grupo sobre objetos ou acontecimentos relevantes e a sua localização exata.

O uso da comunicação por meio de texto e de mapas compartilhados se deve à sua simplicidade de implementação e utilização. Apesar disso, alguns aspectos devem ser ressaltados, como o fato de que ao interagir usando mensagens de texto, autores de transmissões tendem a distrair-se, como relatado em [ENGSTROM et al. 2012], o que influência no conteúdo de vídeo produzido. Dessa forma, as funcionalidades que permitem a troca de mensagens instantâneas e a interação com o mapa devem ser intuitivas e breves.

3 LIVE STREAMING COLABORATIVO BASEADO NA FORMAÇÃO ESPONTÂNEA E DINÂMICA DE GRUPOS

Este capítulo é dedicado a descrever as principais ideias para o *live streaming* em grupo que propomos neste trabalho. Nele apresentamos: (i) uma breve descrição do problema que estamos abordando, essencialmente a ausência de formas de colaboração entre *streamers* no *live streaming*; (ii) os requisitos que devem ser atendidos pela nossa proposta; (iii) o comportamento ou sequencia das funcionalidades que promovem o *live streaming* colaborativo; (iv) os componentes arquiteturais que permitem a execução da nossa proposta; e (v) a relação entre cada uma das etapas e os componentes de arquitetura.

3.1 Descrição do problema

As atuais plataformas de *live streaming* a partir de smartphones desfrutam de popularidade e ampla utilização, especialmente na cobertura de eventos de natureza social e política, tal como ocorre com a Mídia NINJA. Um dos principais fatores de popularidade é a capacidade de imersão de seus espectadores, ao transmitir eventos precisamente no momento e local em que eles estão ocorrendo, possibilitando ainda que ocorra uma interação em tempo real com os seus autores [JUHLIN et al. 2012].

A capacidade de imersão no *live streaming* a partir de smartphones está diretamente ligada à evolução simultânea de tecnologias de banda larga, dos dispositivos móveis e do processo de convergência digital. Apesar disso, ao mesmo tempo as próprias tecnologias também determinam limitações e desafios. Por exemplo, problemas comuns encontrados ao realizar transmissões ao vivo de manifestações são a intermitência no acesso à banda larga e o esgotamento de bateria. É comum, inclusive, autores de *live streaming* participarem de transmissões levando consigo baterias de reserva, ou ainda, notebooks em mochilas para aproveitamento de suas baterias [HESSEL 2013].

Outras dificuldades encontradas por quem realiza o *live streaming* estão relacionadas ao próprio fato delas ocorrerem em tempo real. Nesse caso, um dos principais desafios é a imprevisibilidade durante a captura [JUHLIN et al. 2012]. Os imprevistos vão além de problemas técnicos já mencionados, mas incluem também a possibilidade e risco de ocorrência de situações que coloquem em risco a integridade física de autores de transmissões, como por exemplo, atos de violência ou conflitos. De forma oposta, também é possível que em transmissões de longa duração ocorram poucos fatos de interesse, resultando em frequentes situações tediosas para quem assiste às transmissões.

Por outro lado, plataformas de *live streaming* não contemplam, atualmente, elementos para a produção colaborativa de vídeo. Apesar de fornecerem um canal de comunicação entre fontes e seus espectadores, exclusivo para *feedback* e demais interação, o *live streaming* atual não apresenta formas de comunicação entre fontes. Algumas plataformas, como o Bambuser, ainda exploram informações de geolocalização, inclusive com o uso de mapas, porém é voltado apenas para aspectos de visualização por parte dos espectadores. Mesmo sendo possível que os espectadores identifiquem visualmente duas fontes transmitindo um mesmo acontecimento, essas fontes são incapazes de se perceberem mutuamente num mesmo mapa.

A proposta elaborada no decorrer deste capítulo trata de uma extensão que, a partir de plataformas de *live streaming* existentes, explora informações de contexto fornecidas por fontes e a partir delas permite o a realização de agrupamentos de fontes e a produção colaborativa de vídeo. Ao introduzir a colaboração no *live streaming* buscamos uma abordagem para enfrentar alguns dos desafios hoje encontrados, tanto em limitação de recursos como na imprevisibilidade de coberturas ao vivo, como visto em [JUHLIN et al. 2014].

3.2 Requisitos

Tendo em vista o nosso objetivo de possibilitar a colaboração entre fontes na realização do *live streaming* de um mesmo evento a partir de smartphones, em

essência a nossa proposta de extensão deve ser capaz de:

- Explorar o uso de informações de contexto junto à transmissão de conteúdo de vídeo, e por meio de uso de mapas, permitir que autores de transmissões tenham maior percepção de si mesmos;
- permitir, quando desejado, o agrupamento de fontes que se encontrem em um mesmo contexto, e por meio do uso de mapas, permitir a percepção de outros membros do mesmo grupo; e
- permitir a colaboração entre membros de grupos de fontes a partir de mecanismos de colaboração, também por meio do uso de mapas e troca de mensagens instantâneas.

Além desses três requisitos funcionais, como requisito não funcional, o nosso modelo deve ser capaz de desacoplar o *streaming* de vídeo da troca de informações de contexto. As principais razões para isto são: (1) aproveitar a infraestrutura gratuita oferecida por plataformas de *live streaming* e (2) não limitar-se ao uso de uma plataforma de *live streaming* exclusiva, permitindo que o compartilhamento de informações de contexto ocorra com qualquer uma delas, ou inclusive uma nova implementação.

3.3 Comportamento das funcionalidades de colaboração

A extensão que permite aos *streamers* em um mesmo evento colaborar ativamente é baseada em um processo de três etapas como representado na Figura 4. O primeiro passo visa tornar os participantes mais a par (ou *aware*) dos elementos nas redondezas, tais como objetos de referência e, principalmente, outros participantes. Para isso, mais adiante apresentaremos um conceito chamado de *compartilhamento de contexto*, representando uma sessão na qual os dados de contexto são compartilhados publicamente por um *streamer*. Assim, chamamos essa etapa de *Fornecimento e percepção de contexto*.

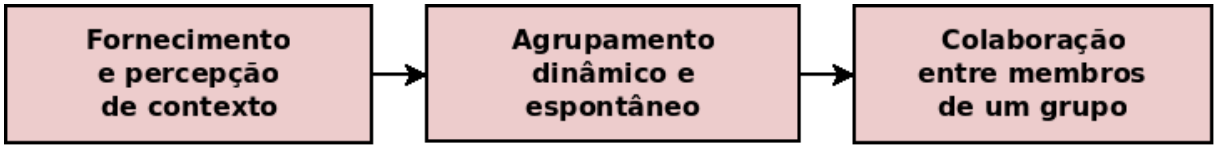

Figura 4: Comportamento do live streaming colaborativo

Usando dados de contexto (por exemplo, a localização geográfica), a etapa seguinte consiste em permitir que *streamers* formem grupos no chamado *Agrupamento dinâmico e espontâneo*. Nela, grupos não são formados automaticamente, mas de maneira ativa pelos próprios autores de transmissões, de acordo com as informações de contexto a eles apresentadas. Outros dados de contexto poderiam ser usados para refinar a formação de grupos, quando a geolocalização não for suficiente, tais como *hashtags* ou informações extraídas do próprio conteúdo multimídia gerado (por exemplo, identificação de objetos ou pessoas na cena).

Uma vez formado um grupo, a terceira etapa consiste na oferta de possibilidades de *colaboração entre membros do grupo*. Especificamente, serão oferecidas funcionalidades que permitam a comunicação por meio de troca de mensagens de texto (um *chat* de grupo) e notificações especiais com o uso de mapas. Ao se comunicarem, participantes adquirem uma forma de coordenar ações e efetivamente colaborar.

3.4 Arquitetura

De forma resumida, o processo de *streaming* de um vídeo engloba tipicamente captura, codificação, *streaming* e playback [AUSTERBERRY 2005]. Para o *live streaming* a partir de smartphones, plataformas tais como Ustream e Livestream fornecem um aplicativo mobile onde conteúdo é capturado, codificado e transmitido sob a forma de uma *stream* de dados para servidores, também fornecidos pelas mesmas plataformas.

Além disso, as plataformas de *live streaming* também fornecem serviços adicionais, tais como autenticação, interação social e um *website* onde é possível

consultar e navegar pelo conteúdo disponível. Quando um espectador, ao navegar no *website*, seleciona uma transmissão em particular, os servidores de mídia iniciam o *streaming* de vídeo para o *player* de vídeo incorporado no *web browser*, onde finalmente o ocorre *playback* [JUHLIN et al. 2012].

Tratando-se de uma extensão para plataformas de *live streaming*, propomos a inclusão das novas funcionalidades a partir um conjunto de serviços adicionais, sem a necessidade de interferir nos elementos de arquitetura das plataformas existentes. A Figura 5 mostra, de forma resumida, os serviços mais comuns em plataformas de *live streaming* (ex. servidores de streaming e *website*) e, abaixo deles (em fonte negrito), um novo conjunto de serviços para o *live streaming* colaborativo, além de seus clientes.

Figura 5: Visão geral da arquitetura de extensão para o live streaming colaborativo

Os novos serviços são independentes daqueles já existentes (especialmente de servidores de streaming), embora eventualmente seja conveniente a integração entre eles. Por exemplo, o *correlacionador* (componente apresentado adiante na Figura 3) poderia acessar serviços de *streaming* e extrair características no conteúdo de vídeos disponíveis no momento de identificar possíveis grupos, incluindo essas novas informações como parte do contexto a ser considerado para o agrupamento.

Mais adiante, ao apresentar o protótipo desenvolvido, mostraremos que

mantendo serviços independentes nos permite o *piggyback* (aproveitamento) das principais plataformas de *live streaming*, implementando os novos conceitos de *compartilhamento de contexto*, *grupo* e *serviço de colaboração*, sem preocupação com a codificação e o *streaming* de conteúdo.

A comunicação entre esses elementos adicionais ocorrerá por meio da troca de mensagens em um canal de dados paralelo ao fluxo de conteúdo multimídia (representadas nas setas da Figura 6), de maneira similar ao tratamento das interações sociais e do *feedback* de espectadores.

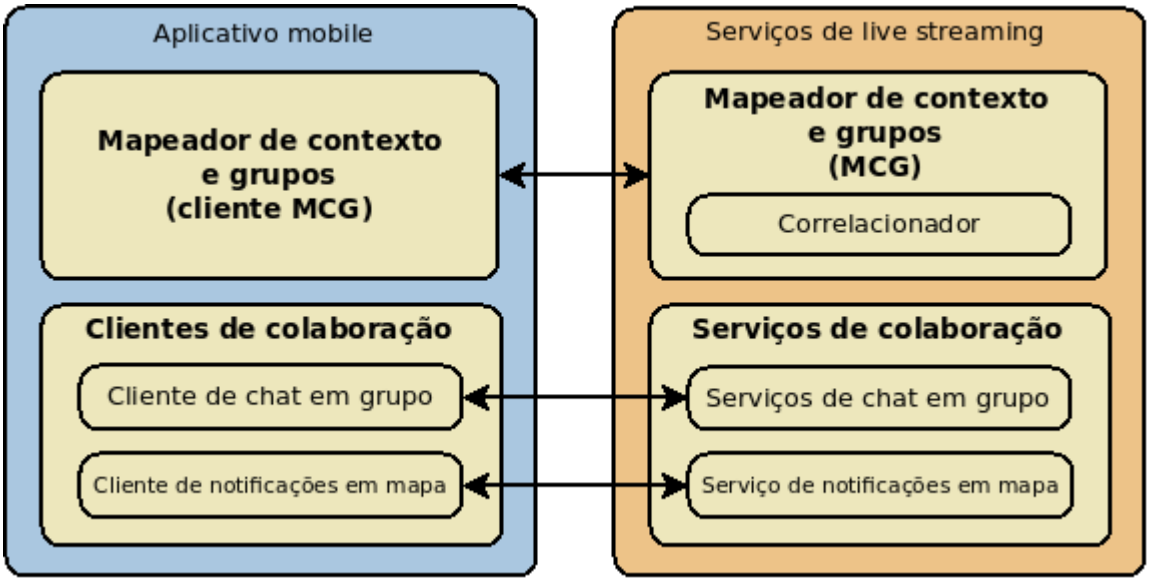

Figura 6: Arquitetura da extensão para o live streaming colaborativo

De forma concreta, propomos adicionar os serviços apresentados na Figura 6 (expandindo os novos serviços da Figura 5), incluindo os seus respectivos clientes. Primeiramente, um serviço chamado de *Mapeador de contexto e grupos* (MCG) será responsável por lidar com a etapa de *fornecimento e percepção de contexto* e, em seguida, será usado também para lidar com a etapa de formação de grupos.

O MCG serve como um repositório que armazena dados de contexto e grupos associados a um *live stream*. Os dados de contexto são obtidos pelo *cliente*

MCG (no dispositivo mobile) e enviados para o MCG enquanto houver um *compartilhamento de contexto* ativo, apresentados em breve. Os dados obtidos pelo cliente MCG são fornecidos diretamente por autores de transmissões (por exemplo, *hashtags*) ou, em sua grande maioria, obtidos automaticamente dos recursos disponíveis em smartphones, como GPS e diversos sensores.

O MCG possui um componente especial chamado de *correlacionador* que, ao usar as informações de contexto disponíveis no próprio MCG, identifica *live streams* de contextos similares e, portanto, têm o potencial de pertencer a um mesmo grupo. Como já mencionado, grupos não são formados automaticamente. Ao invés disso, ao identificar transmissões em um mesmo contexto, o correlacionador notifica cada *streamer* sobre a presença de outros participantes e, em seguida, *streamers* decidem individualmente pertencer (ou não) a um grupo.

Formados os grupos, um conjunto de serviços chamados de *Serviços de colaboração* fornecem de fato as funcionalidades de colaboração para *streamers* dentro de um mesmo grupo. Propomos inicialmente dois tipos de serviços de colaboração: (i) um *Serviço de chat de grupo*, para a comunicação baseada em troca de mensagens de texto entre membros de um grupo e; (ii) o *Serviço de notificações em mapa*, para o envio de mensagens marcadas com informações de geolocalização, chamadas de *notificações em mapa*. A arquitetura da extensão é modular e, portanto, novos *serviços de colaboração* podem ser especificados futuramente, como por exemplo, um *Serviço de mensagens de áudio*.

Seguindo o exemplo de arquiteturas de protocolos de comunicação em tempo real, como o XMPP [XMPP 2015], os serviços de colaboração estão associados a componentes no *server-side* (ex. *queues* ou *brokers*) e os clientes de colaboração a seus respectivos clientes. Os serviços de colaboração desta proposta foram baseados em funcionalidades de colaboração presentes em jogos digitais multijogador, detalhados em [MONTANE-JIMENEZ 2014] e [JORVID 2014].

3.5 Fornecimento e percepção de contexto

Informações de contexto são fornecidas pelas fontes por meio do que chamamos de um *Compartilhamento de contexto*. Este pode ser visto como um tipo de sessão finita durante a qual dados de contexto são enviados do aplicativo de captura, a partir do *cliente MCG*, para o serviço chamado de *mapeador de contexto e grupos* (MCG). Durante o seu ciclo de vida, um compartilhamento de contexto pode ser interrompido quando desejado ou automaticamente quando for detectada a ocorrência de erros (ex. perda de conexão). Nesse último caso, ele deve ser reiniciado instantes mais tarde. Quando um compartilhamento de contexto é encerrado, um novo compartilhamento deve ser feito para que os dados de contexto voltem a ser fornecidos.

Definimos que o compartilhamento de contexto está desacoplado do *streaming* de vídeo que ele representa. Assim, é possível interromper um compartilhamento de contexto enquanto mantém-se a transmissão de conteúdo de vídeo e vice-versa, dependendo de requisitos específicos, como a economia de energia. Entretanto, em termos da qualidade do conteúdo apresentado aos espectadores, o ideal é que o *streaming* de vídeo e o seu compartilhamento de contexto sejam mantidos simultaneamente.

Os dados iniciais que representam o contexto de um *streamer* são: coordenadas geográficas, orientação relativa da câmera, *hashtags*, *timestamps*, níveis de bateria, estado do dispositivo mobile (ex. bateria fraca ou sinal de banda larga ruim), estado do *streamer* (ex. machucado, detido, correndo), entre outros. Outras informações que podem ser incluídas a um compartilhamento de contexto são: qualidade da câmera, quantidade de espectadores, *ratings* de espectadores, *ratings* de membros do grupo (relacionado à colaboração).

O fornecimento de informações de contexto, representadas por um compartilhamento de contexto, é tratado pelo MCG (mapeador de contexto e grupos) e o cliente MCG, nos *server-side* e aplicativo mobile respectivamente. A captura de contexto é realizada pelo cliente MCG e consiste em obter os dados de contexto mencionados acima. Os dados são obtidos de duas formas: (i) fornecidas pelo autor da transmissão (ex. *hashtags* ou estado do *streamer*) e (ii) obtidas

diretamente dos seus dispositivos móveis (coordenadas geográficas ou estado do próprio dispositivo) a partir de API.

Depois de capturados, os dados de contexto são enviados para o MCG, onde ficam armazenados. Esses dados podem ser acessados pelos demais componentes da arquitetura para implementar funcionalidades adicionais ao sistema. O correlacionador, por exemplo, usa as informações de contexto para identificar *streamers* que possivelmente participam de um mesmo evento, agrupando-os por proximidade geográfica.

O MCG e o cliente MCG, além de armazenar e capturar informações de contexto respectivamente, ainda podem monitorar informações de contexto e acionar interações conforme requisitos de aplicação. Por exemplo, o cliente MCG poderia monitorar o nível de bateria e, ao atingir um limite estabelecido, enviaria uma atualização no estado do dispositivo mobile para “bateria baixa” e inclusive emitir uma *notificação de ajuda* (um tipo de notificação em mapa que será apresentada adiante). Além disso, o MCG poderia monitorar aspectos de colaboração, por exemplo, analisando o deslocamento de *streamers*.

O cliente MCG também usa informações para fornecer ao seu *streamer* maior percepção sobre o seu contexto. Por exemplo, ao capturar as suas novas coordenadas geográficas, além de enviá-las para o MCG, o cliente MCG disponibiliza-as para serem acessadas pelo aplicativo mobile e renderizadas em um mapa. Usando o mapa, *streamers* podem identificar pontos de referência próximos, tais como ruas ou praças.

Quando participando de um grupo, o cliente MCG é também responsável por obter informações do próprio grupo, disponíveis no MCG. Assim, o aplicativo *mobile* também renderiza em mapa a localização dos membros do grupo. Exibir a localização de colegas de grupo, junto a funcionalidades de colaboração descritas à frente, permite um nível inicial de planejamento e coordenação de ações durante o *live streaming*, tais como menor redundância de conteúdo ou definição de alvos (sabendo locais que já estão sendo cobertos).

Dados de contexto são trocados por meio de mensagens enviadas do cliente MCG, no aplicativo mobile, para o MCG. Dois tipos de mensagens foram estabelecidas inicialmente: *mensagens de ciclo de vida* e *mensagens de atualização de contexto*.

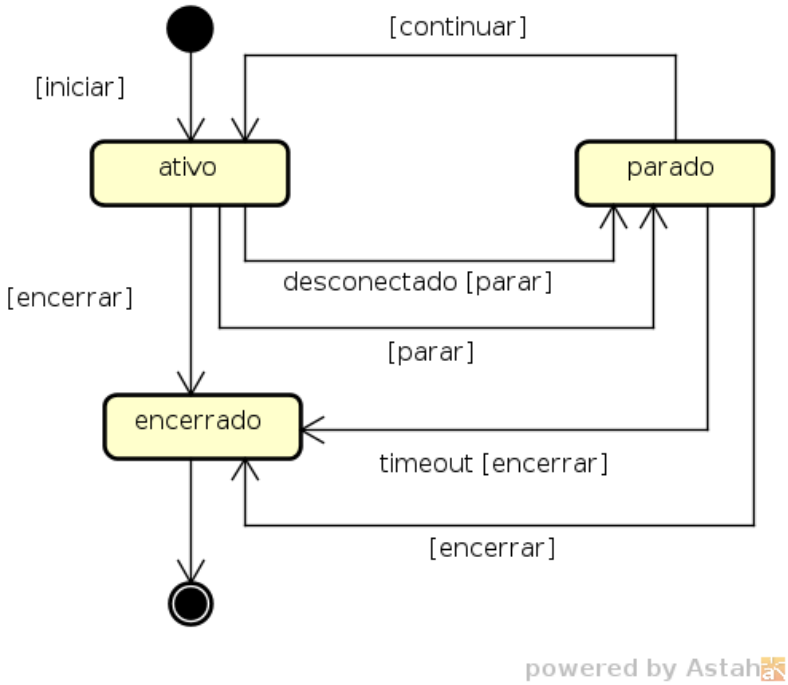

Figura 7: Ciclo de vida de um compartilhamento de contexto

Mensagens de ciclo de vida, representadas na Figura 7, estão associadas às mudanças de estado de compartilhamentos de contexto. Seguem as atribuições de cada uma delas:

- A mensagem *iniciar* é usada ativamente quando o streamer deseja criar um novo compartilhamento, definindo o seu estado como *ativo* e iniciando o envio periódico de mensagens de *atualização de contexto* (vistas adiante).
- A mensagem *parar* também é usada ativamente por *streamers*, ou é chamada automaticamente pelo cliente MCG quando ocorrem erros (ex. perda de conexão banda larga), alterando o seu estado para *parado*. Mensagens de atualização de contexto são temporariamente interrompidas enquanto o

compartilhamento de contexto mantém-se no estado de *parado*.

- Um compartilhamento de contexto é reativado por meio da mensagem *continuar*, que é usada ativamente pelo autor da transmissão, redefinindo o seu estado para *ativo* e retomando o envio periódico de mensagens de atualização de contexto.

- A mensagem *encerrar* termina permanentemente um compartilhamento de contexto junto com o envio de mensagens de atualização associadas a ele, modificando o seu estado para *encerrado*. Essa mensagem é chamada manualmente, quando desejado pelo autor, ou, o MCG pode ser configurado para automaticamente terminar um compartilhamento de contexto quando mantido por muito tempo no estado de *parado* (ex. *timeout*).

Quando um compartilhamento de contexto chega ao estado *encerrado*, para voltar a fornecer informações de contexto um novo compartilhamento deve ser iniciado, estabelecendo um novo ciclo de vida para um novo compartilhamento. A principal razão disso é tentar associar um compartilhamento de contexto à um determinado evento, por exemplo, permitindo representar “manifestação de 17 de Junho de 2013” e “manifestação de 20 de Junho de 2013”.

Outra mensagem dentro do conjunto de *mensagens de ciclo de vida* é a mensagem *estado*. Ela se destina a obter do MCG o estado do atual compartilhamento de contexto. Assim, o cliente MCG pode prevenir o envio de mensagens de maneira equivocada, por exemplo, pode prevenir a continuação de um compartilhamento de contexto que tenha sido encerrado automaticamente devido a *timeout*.

Mensagens de atualização de contexto contêm efetivamente os dados de contexto que são capturados pelo cliente MCG e serão armazenados no MCG. Elas apenas são enviadas quando um compartilhamento de contexto está ativo e o seu envio é feito periodicamente, em intervalos variáveis de acordo com restrições de aplicação. Por exemplo, os envios podem ser feitos em intervalos longos, quando a bateria estiver próxima do esgotamento ou em intervalos mais curtos, se ela estiver

com carga acima de certo valor percentual.

3.6 Grupos espontâneos e dinâmicos

O próximo passo é agrupar autores de *streams* presentes no mesmo evento. Grupos são sugeridos de acordo com compartilhamentos de contexto, cujos dados são disponibilizados no MCG. Em nossa proposta as possibilidades de agrupamento são identificadas e sugeridas de forma individual a cada *streamer* (com base em características associadas a ele). Cada *streamer* decide se aceita ou não participar do grupo após a sugestão. Sendo assim, o ingresso de membros em um grupo ocorre de forma espontânea (a qualquer instante desejado) e dinâmica (é possível a entrada e saída livremente). Por fim, um streamer só pode pertencer a um grupo por vez.

Optamos por esse modelo de agrupamento principalmente para contornar desafios em rearranjo constante de membros, ocasionados pelas características inerentes aos aplicativos para dispositivos mobile, em particular, mobilidade e disponibilidade variável. Por exemplo, partindo do princípio de que um grupo é formado porque todos seus membros decidiram colaborar com outros *streamers* em um mesmo evento, é possível lidar com problemas de perda de conexão ou afastamento do raio de atuação de maneira mais flexível, já que espera-se que o *streamer* se recupere ou que ele mantenha o seu objetivo (mesmo que critérios de identificação de agrupamentos não tenham sido fortemente atendidos).

Por outro lado, trata-se também de uma abordagem para lidar com falsos agrupamentos (ou agrupamentos indesejados), uma vez que os membros de um grupo decidiram participar dele por vontade própria. A partir do instante que grupos foram identificados e notificados, com as informações que descrevem cada um deles, *streamers* são capazes de distinguir, por exemplo, usando *hashtags*, diferentes eventos que ocorrem em uma mesma localização geográfica ou ainda grupos com objetivos distintos (ex. orientação política).

Neste trabalho não abordamos questões de restrição de acesso ou

moderação dentro de grupos. A nossa proposta baseia-se inicialmente em interações livres, espontâneas e dinâmicas ocorrendo dentro do grupo.

Um grupo surge no momento em que um novo compartilhamento de contexto é iniciado, sendo este o seu membro inicial e único. Nesse primeiro instante, o estado do grupo é definido como *ativo* (ver Figura 8). O número de participantes varia de acordo com o ingresso e a saída de *streamers* num grupo. Quando um grupo perde todos os seus membros o seu estado altera-se permanentemente para o estado de *inativo*.

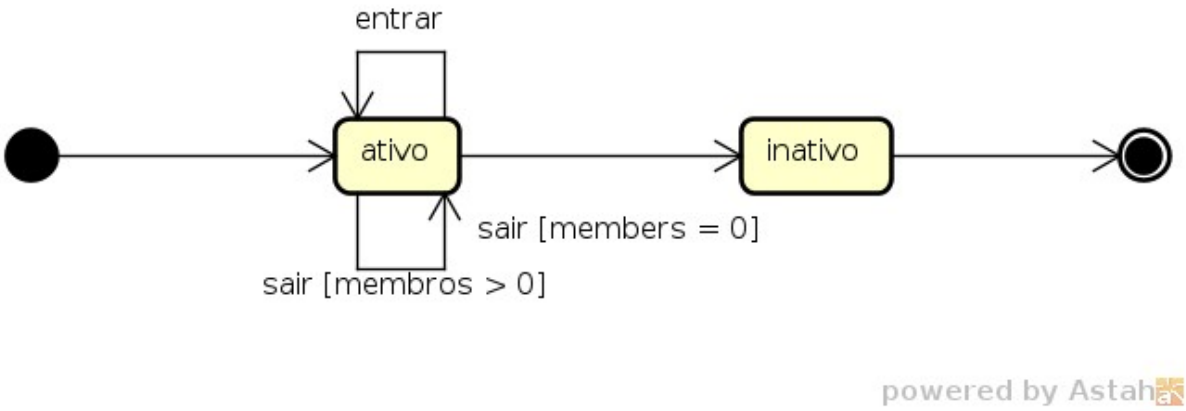

Figura 8: Estados de um grupo

A entrada e saída de *streamers* em um grupo é realizada por meio mensagens enviadas também ao MCG, chamadas de *mensagens de agrupamento*, algumas delas visíveis na Figura 8 e descritas à seguir.

Para que um *streamer* ingresse em um grupo com contexto similar ao seu, ele necessita inicialmente obter conhecimento sobre os grupos candidatos. Uma das formas de fazer isto é solicitando ao MCG que obtenha grupos ativos dentro do mesmo contexto, em especial, os que estiverem em localização próxima. A forma de solicitar essas informações é usando a mensagem *encontrar grupos*. Ela retorna, para cada grupo encontrado, um identificador, a quantidade de membros e um conjunto agregado de suas *hashtags*. O conjunto de grupos candidatos é apresentado para o autor da transmissão no aplicativo de captura.

Em seguida, o *streamer* pode se juntar a um dos grupos sugeridos pelo MCG por meio da mensagem *entrar em grupo*, informando o seu identificador. Quando uma fonte de transmissão passa a fazer parte de um grupo, ela automaticamente abandona o seu grupo anterior. Além disso, ao ingressar no grupo, recebe-se do MCG informações sobre os seus atuais membros, incluindo localização geográfica, estado de bateria, etc. Essas informações também podem ser solicitadas ativamente em seguida, usando a mensagem *obter membros*.

Um *streamer* pode retirar-se de um grupo quando desejado, chamando a mensagem *sair de grupo*. Ao sair do grupo, um novo grupo é criado automaticamente, contendo apenas o *streamer*. A partir desse momento, novos participantes podem também ingressar no grupo recém-criado. Grupos não possuem hierarquia ou moderação, portanto, não é possível banir membros ou impedir a entrada de um usuário específico.

Algumas operações simples relacionadas aos dados de seus membros podem ser realizadas pelo MCG. Dentre elas, podem ser citadas agregações e sumarizações, descrevendo características gerais do grupo e obtidas a partir de características de seus membros.

O GCM possui um elemento especial chamado de *correlacionador*. Ele é um componente que periodicamente varre os dados de contexto e grupos e, com eles, busca por *clusters* de grupos em contextos parecidos. Em um cenário inicial *clusters* são obtidos por geolocalização (centroide), mas outras informações podem ser usadas também para filtrar/refinar os resultados obtidos. Por exemplo, um *cluster* poderia ser dividido em dois de acordo com o estudo das *hashtags* de seus grupos membros, por exemplo, quando não possuírem *hashtags* relacionadas.

Depois de que o *correlacionador* identifica um *cluster*, seleciona-se um de seus membros, um grupo, e define-se como grupo pivô. A seleção pode ser feita a partir de diferentes atributos, tais como quantidade de membros, maior tempo agregado de transmissão, maior número de espectadores, etc. O grupo pivô é tomado como referência e em seguida é sugerido para todos os *streamers* dentro dos demais grupos do *cluster.* Essa sugestão é realizada usando uma mensagem

chamada de *sugestão de grupo*. Quando um *streamer* recebe uma *mensagem de sugestão de grupo*, ele pode livremente escolher ingressar no grupo sugerido ou simplesmente ignorá-la.

O MCG também pode realizar o monitoramento de informações de compartilhamentos de contextos e grupos. Por um lado, isso permite acompanhar o estado tanto de recursos (bateria, conectividade, perfil), quanto dos próprios *streamers*, e ao identificar cenários de interesse acionar rotinas de tratamento. Por exemplo, a identificação de um participante com níveis baixos de bateria e, em seguida, o envio de uma *notificação de ajuda* (apresentada mais adiante) para os seus companheiros de grupo.

Por outro lado, o monitoramento aplica-se também para o acompanhamento e avaliação de aspectos de colaboração, iniciado a partir do início das interações entre membros de um grupo, principalmente de *notificações em mapa*. Ao enviar uma notificação em mapa, dependendo do seu propósito (ex. um pedido de ajuda), o monitoramento do deslocamento dos outros participantes seria iniciado, para permitir identificar se eles estão se comportando conforme necessário/esperado após um pedido de colaboração/ajuda.

3.7 *Live streaming* colaborativo

Para efetivamente promover o *live streaming* colaborativo a partir de smartphones, fornecemos a grupos um conjunto de funcionalidades que permitem que *streamers*, dentro do mesmo grupo, possam se comunicar e coordenar suas ações em tempo real, enquanto levam a cabo a cobertura ao vivo de um evento.

Essas funcionalidades são oferecidas por *Serviços de colaboração* e respectivos *clientes de colaboração* (Figura 6). Um *Serviço de colaboração* abrange os *software* e infraestrutura necessária para, dada uma funcionalidade de colaboração, permitir a troca de mensagens entre membros de um grupo. O *cliente de colaboração*, disponível no aplicativo mobile, permite que o *streamer* componha, envie, receba e visualize mensagens de uma funcionalidade de colaboração

específica, interagindo diretamente com o respectivo *serviço de colaboração*.

Até o momento deste trabalho, duas funcionalidades de colaboração (serviço e cliente) foram estabelecidas: *chat de texto em grupo* e *notificações em mapa*. Essas funcionalidades são baseadas em funcionalidades típicas de jogos *multiplayer*, tipicamente incluindo uma visão em mapa ou radar e interações por texto ou voz [MONTANE-JIMENEZ 2014], cada qual com vantagens e desvantagens de acordo com um propósito específico.

As funcionalidades de colaboração têm em comum a necessidade de que a comunicação ocorra em tempo real, assim, protocolos de comunicação em tempo real tornam-se úteis na implementação dos serviços e clientes de colaboração. Especificamente, em nosso protótipo implementamos ambos os componentes baseados na arquitetura do XMPP, onde o *server-side* consiste basicamente de um servidor Jabber/XMPP e o *client-side* de clientes XMPP [XMPP 2015].

A primeira funcionalidade é o *chat de texto em grupo*, que permite que membros de um grupo se comuniquem e coordenem em tempo real por meio da troca de mensagens de texto. A troca de mensagens de texto enquanto realiza-se a transmissão de vídeo ao vivo já encontra-se presente na maioria de plataformas de *live streaming*, entretanto, ela ocorre apenas entre um *streamer* e os seus espectadores. O serviço de chat em grupo oferece a *streamers*, membros de um mesmo grupo, um canal de comunicação horizontal, ou seja, entre eles mesmos.

As principais vantagens no uso de mensagens de texto são a flexibilidade (liberdade de expressar-se sobre qualquer assunto) e simplicidade de implementação. Dentre as desvantagens, destacamos que a composição de mensagens de texto durante a realização de um *live streaming* em smartphone pode ser lenta e inclusive influenciar na qualidade do conteúdo gerado, pois demanda atenção do *streamer* e pode causar distração da atividade de captura.

Uma consequência indesejada seria, por exemplo, encontrar constantes segmentos de vídeo contendo a captura do chão, já que é comum posicionar a câmera traseira do dispositivo (a câmera principal) virada para o chão enquanto

escreve-se uma mensagem de texto. Por outro lado, na cobertura de protestos *streamers* (como no Mídia NINJA) já interagem com os seus espectadores por meio de mensagens de texto (o canal de *feedback*), provando certa familiaridade com esse tipo de funcionalidade e abrindo uma brecha para a adequação de um chat entre *streamers* de um mesmo grupo.

Notificações em mapa, a segunda funcionalidade de colaboração, são mensagens que carregam um sentido semântico bem definido e incluem informações geográficas que são apresentadas usando um mapa. O seu uso parte da intenção de um *streamer* em notificar os demais participantes com informações que dependem de sua localização. Inicialmente o autor seleciona o tipo de notificação que deseja enviar, cada um deles serve para propósitos específicos. Em segundo lugar, são fornecidas as coordenadas geográficas ao selecionar um ponto na visualização de mapa do aplicativo mobile. Por último, fornece-se uma breve descrição em texto. Depois de criada, a notificação em mapa é então enviada para todos os membros do grupo.

Quando uma notificação em mapa é recebida pelo aplicativo mobile, ela é apresentada na visualização de mapa em seguida, usando as coordenadas geográficas a ela associadas. A descrição textual da notificação é apresentada ao streamer após ser selecionada no mapa. A apresentação (ou a sua estética) de notificações varia de acordo com o seu tipo. Notificações em mapa são temporárias, desaparecendo depois de um determinado tempo.

Quatro tipos de notificações em mapa foram definidos até o momento: *notificação de alvo, notificação de perigo, notificação de ajuda, notificação de deslocamento*. Todas elas inspiradas em jogos de tipo *multiplayer* [MONTANE-JIMENEZ 2014].

A *notificação de alvo* serve para informar um objetivo em mapa que possa ser de interesse dos participantes, por exemplo, em manifestações, o local de pronunciamentos ou assembleias. O seu uso inclui também a definição de pontos de encontro.

A Figura 9 ilustra a forma de uso a notificação de alvo. Nela (e nas demais figuras usadas para representar as demais notificações), o círculo de cor verde representa o *streamer* que envia a notificação de mapa e, os círculos azuis, os demais participantes do grupo (que recebem a notificação). O triangulo anexado a cada *streamer* representa o seu campo de visão.

Do lado esquerdo (Figura 9.a) é representado o instante em que a notificação é enviada, incluindo a localização geográfica (marcações circulares verdes em sobreposição) e a sua descrição (ex. “encontremo-nos aqui”). Do lado direito (Figura 9.b) é representado o comportamento esperado dos participantes depois de recebê-la, neste caso específico, todos os interessados deslocam-se em direção do local informado.

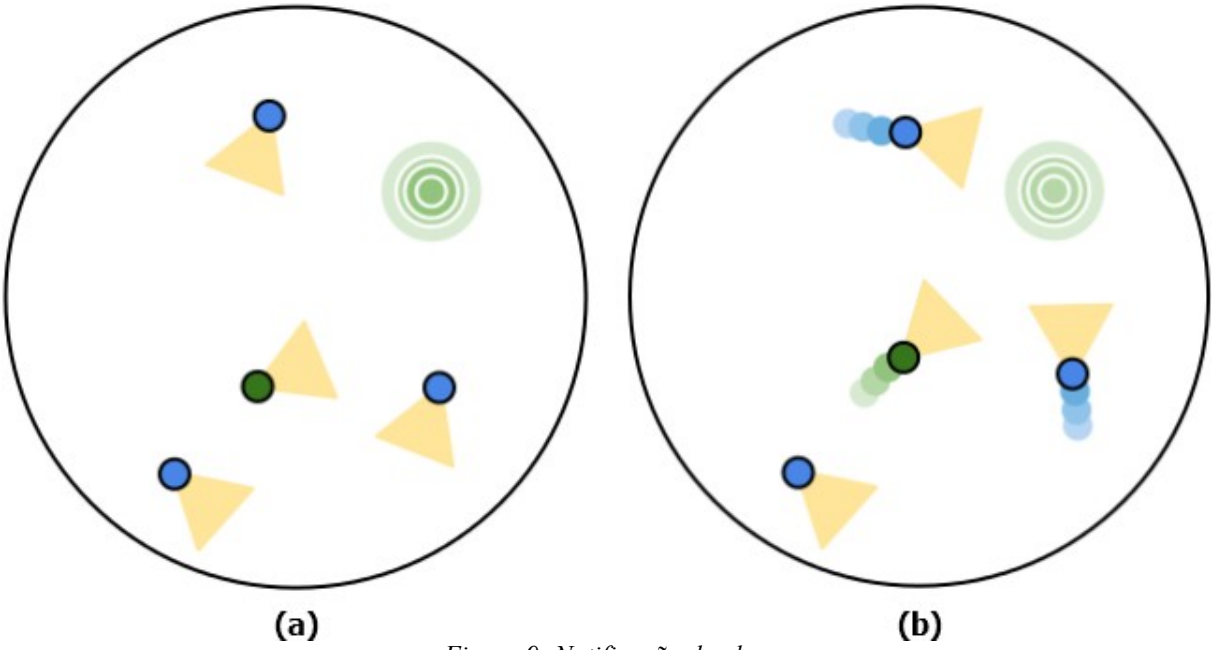

Figura 9: Notificação de alvo

A *notificação de perigo* tem o objetivo de advertir sobre a presença/ocorrência de situações de perigo durante o evento, especialmente aquelas que coloquem a vida de participantes em risco, dessa forma eles podem evitá-las. Devido à possibilidade de representar conteúdo crítico, notificações de

perigo devem ganhar destaque no momento de apresentá-las aos autores de transmissões.

A Figura 10.a mostra o envio de uma notificação de perigo em sua localização geográfica, representada por marcações circulares de cor vermelha em sobreposição. Exemplos de notificação de perigo, como em uma manifestação, são ações violentas de policiais ou manifestantes, os quais são fornecidos na descrição de texto (ex. “confronto”). A Figura 10.b mostra que, depois de obter conhecimento sobre uma situação de perigo, os participantes que receberam a notificação retiram-se e mantêm distância do local informado, incluindo o *streamer* que enviou a notificação de perigo.

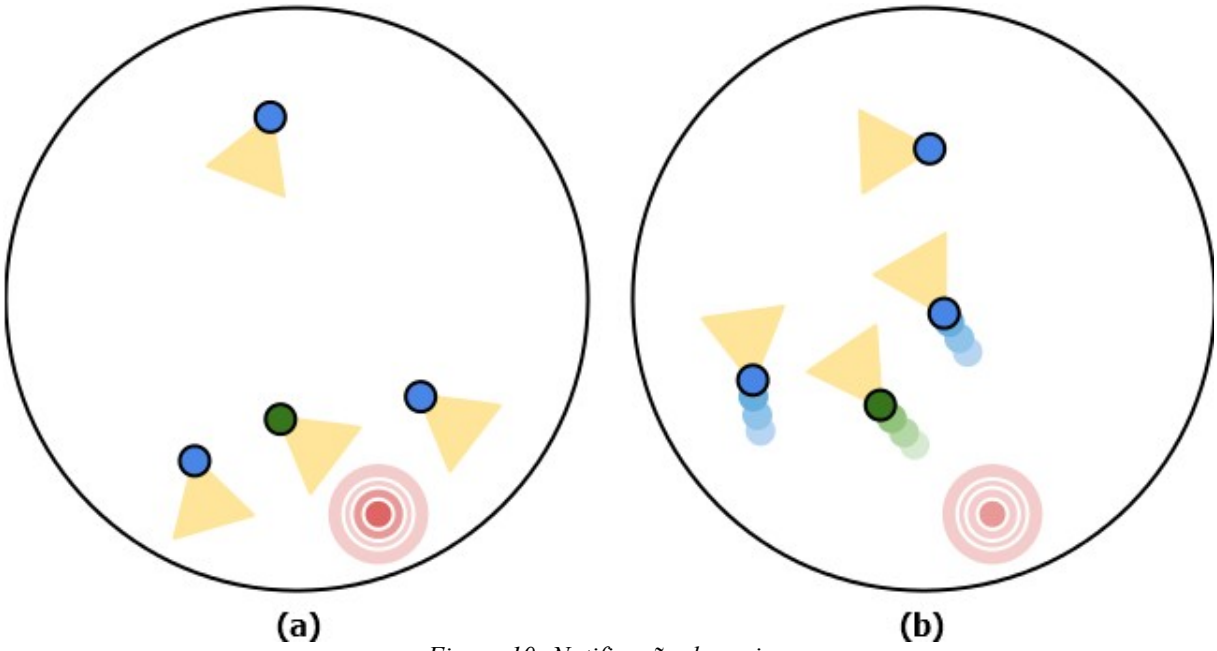

Figura 10: Notificação de perigo

A *notificação de ajuda* serve para solicitar ajuda a participantes próximos. Elas são genéricas, e as situações nas quais se aplicam vão desde problemas técnicos (ex. pouca bateria ou compartilhamento de acesso à Internet) até ameaças físicas (ex. machucar-se, ficar encurralado ou ser detido).

A Figura 11.a mostra o instante em que uma notificação de ajuda é enviada.

As coordenadas geográficas associadas a ela são as mesmas que as do solicitante, representada nas circunferências de cor verde que se sobrepõem ao solicitante. Além disso, é fornecida a descrição em texto, a qual especifica a situação que está enfrentando-se (ex. “bateria acabando” ou “estou preso”).

A Figura 11.b mostra que, após receber a notificação de ajuda, a fonte mais próxima ao solicitante desloca-se em sua direção, possivelmente para atender ao pedido de ajuda.

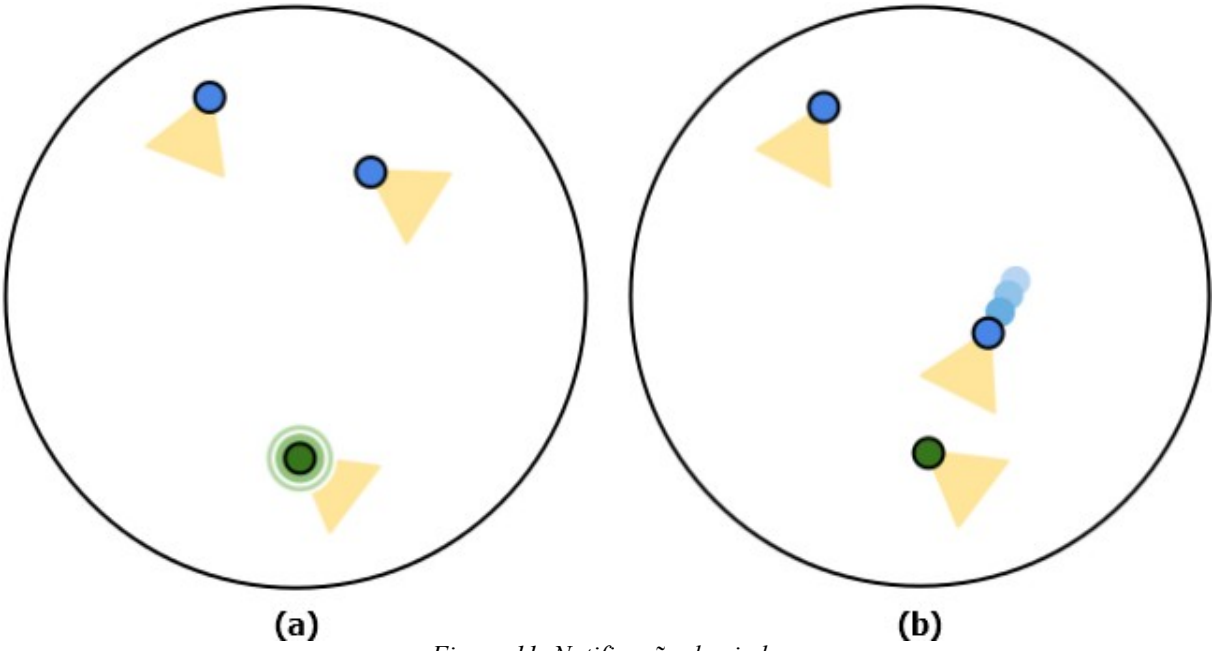

Figura 11: Notificação de ajuda

Notificações de ajuda também podem ser enviadas automaticamente pelos MCG ou cliente MCG. Como mencionado anteriormente, esses componentes de nossa arquitetura realizam, dentre suas atribuições, o monitoramento de informações de contexto e grupos. Assim, ao detectar que o nível da bateria atingiu certo limite predefinido, o cliente MCG poderia enviar uma notificação que contém as atuais coordenadas geográficas e a descrição “bateria acabando”.

Foi dito anteriormente que o MCG seria capaz de acompanhar a disposição um *streamer* em colaborar com os demais. Um cenário no qual isso pode ocorrer é a

partir do envio de uma notificação de ajuda, iniciando um ciclo de monitoramento, ou acompanhamento, de colaboração, encerrando-se no fim da notificação (já que elas são temporárias).

Uma situação desse tipo ocorre quando uma notificação de ajuda é enviada, o MCG daria início ao acompanhamento da movimentação (deslocamento) de *streamers*, chat em grupo e outras notificações em mapa que se seguissem, avaliando se as ações realizadas vão em direção a interagir com o *streamer* solicitante de ajuda (ex. deslocar-se em sua direção, fornecer *feedback*, etc.). A partir dessas avaliações, *streamers* poderiam obter *ratings* positivos que expressassem a sua capacidade de colaborar com os demais participantes.

A última notificação em mapa projetada neste trabalho foi a *notificação de deslocamento*. Ela serve para que um *streamer* informe a localização para a qual ele se deslocará. A Figura 12.a mostra o instante no qual um *streamer* envia uma notificação de deslocamento, representada como círculos intercalados em cor verde. Na Figura 12.b o mesmo *streamer* aproximando-se do local antes informado.

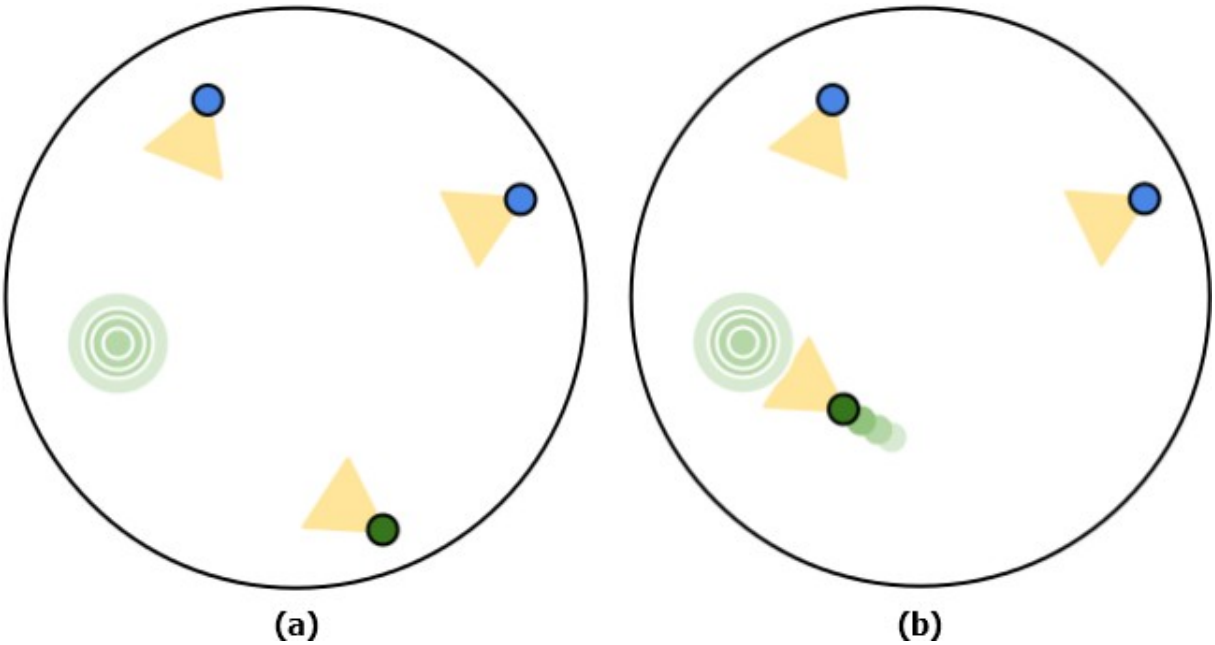

Figura 12: Notificação de deslocamento

A *notificação de deslocamento* tem dois propósitos principais. O primeiro relaciona-se à coordenação de transmissões colaborativas ao vivo em casos nos quais deseja-se ter cobertura mais ampla e menos redundante. Isso acontece uma vez que, quando um *streamer* ganha conhecimento a respeito das localizações atuais e futuras dos seus colegas, ele obtém melhores informações para decidir as suas próximas ações (ex. evitando locais já cobertos). O segundo propósito é mais imediato, e consiste em usar notificações de deslocamento para fornecer um *feedback* a pedidos de ajuda. Assim, quando um *streamer* receber uma notificação de ajuda e tomar a decisão de atendê-la, ele poderia enviar uma notificação de deslocamento, passando as coordenadas geográficas do solicitante (quem pediu ajuda). Desta maneira, o solicitante recebe um *feedback* e toma conhecimento de que alguém irá atendê-lo. Os demais participantes também ganham conhecimento do fato e, dependendo da situação, podem desconsiderar o pedido de ajuda (já que outra pessoa interveio).

4 IMPLEMENTAÇÃO E AVALIAÇÃO EXPERIMENTAL DO PROTÓTIPO DESENVOLVIDO

Neste capítulo apresentamos aspectos relacionados à implementação do protótipo, tal como o desacoplamento entre as funcionalidades propostas e as plataformas de *live streaming* existentes, o que permite fazer o *piggyback* com qualquer uma delas, mantendo as funcionalidades de colaboração. Descrevemos resumidamente também o protocolo XMPP, base para a implementação dos serviços de colaboração, além do modelo de entidades usado para representar informações de contexto e grupos. O capítulo também inclui a análise de alguns experimentos realizados a partir do protótipo implementado.

4.1 O Protocolo XMPP

O XMPP (*eXtensible Messaging and Presence Protocol*) é uma tecnologia aberta para a comunicação em tempo real, usando o XML como o formato de base para a troca de informações [SAINT-ANDRE et al. 2009]. Ele tem sido usado em aplicações de mensagens instantâneas bastante populares, como o Facebook Messenger, GTalk (Google), Skype e WhatsApp. Particularmente, o WhatsApp usa uma versão customizada do XMPP, chamada de FunXMPP [WHATSAPP 2014].

O protocolo XMPP é usado em diversas aplicações, dentre elas a troca de mensagens instantâneas (IM), chat em grupo, jogos, VoIP, transferência de dados, entre outros. Esse leque de possibilidades deve-se à quantidade de serviços já implementados para o protocolo, incluindo, além da troca de mensagens um-para-um e em grupo (*multi-party*), canais criptografados, autenticação, presença, lista de contatos ou descoberta de serviços.

Outra característica do XMPP é a extensibilidade. Com mais de cem extensões, o XMPP provavelmente fornece todos os meios necessários para o desenvolvimento de aplicações em tempo real. A sua especificação afirma que criar

novas extensões é fácil o suficiente, assim como a interação com a comunidade XMPP no sentido de padronizar as novas funcionalidades propostas [XMPP 2015].

A arquitetura do XMPP consiste nos chamados servidores Jabber (ex. ejabberd e Openfire) e clientes Jabber (ex. os mensageiros instantâneos Gajim, Pidgin e Psi) [XMPP 2015]. Além disso, existem também dezenas de bibliotecas disponíveis para um grande número de linguagens de programação, particularmente também para ambientes mobile (ex. Smack, para ambiente Android, e xmppframework, para ambiente iPhone), assim permitindo desenvolver uma ampla variedade de aplicações baseadas em XMPP.

Especificamente, o XMPP usa uma arquitetura cliente-servidor descentralizada, de forma parecida às arquiteturas Web e de *e-mail*. Isto permite que usuários de em domínios de rede distintos possam se comunicar, desde que os servidores Jabber em cada um deles estejam vinculados.

Em termos de endereçamentos, um identificador é chamado de JabberID é atribuído a cada usuário. O JabberID tipicamente depende de serviços DNS, ao invés de usar endereços IP.

Um JabberID é escrito na forma <*usuário*>@<*domínio*>/<*recurso*>, que traduz-se a um determinado usuário (<*usuário*>), pertencente a um determinado domínio (<*domínio*>), acessando a partir de um determinado recurso, ou dispositivo (<*recurso*>). Por exemplo, *joao@lprm.inf.ufes.br/galaxys2* é o JabberID de João, vinculado ao laboratório LPRM (de domínio *lprm.inf.ufes.br*), logado a partir de um smartphone Galaxy S II.

Os recursos mais interessantes do XMPP são as suas primitivas para a comunicação, chamadas de *stanza*, unidade básica de comunicação análoga a pacotes ou mensagens em outros protocolos de Internet. As stanzas são escritas respeitando o formato XML. O XMPP define três tipos de stanzas, sendo elas *mensagem* (*message*), *presença* (*presence*) e *IQ*.

Uma *mensagem* é a forma mais básica e genérica de troca de informações

entre dois usuários, tais como aquelas usadas em mensageiros instantâneos (IM). Mensagens contêm os JabberID de origem e destino, além do *corpo* de texto, que contém o seu conteúdo, ou as informações que querem se transmitidas. Com essas informações, dois usuários XMPP são capazes de comunicar-se, onde um servidor XMPP é responsável por fazer o encaminhamento das mensagens enviadas.

Enquanto as mensagens XMPP típicas são de um-pra-um, em nossa proposta todos os membros de um grupo interagem simultaneamente por meio destas mensagens. Por outro lado, dentre as muitas extensões XMPP disponíveis, algumas delas são dedicadas ao uso diferente de mensagens, tais como as extensões XEP-0045 (ou *XMPP Extension Protocol* 0045) e XEP-0060, que respectivamente disponibilizam o chat entre múltiplos usuários e o padrão *publish-subscribe*.

Quando implementado pelo servidor XMPP, o XEP-0045 (ou *Multi-user Chat*) permite a criação de salas de chat onde podem participar múltiplos usuários ao mesmo tempo. As salas de chat são mantidas por um serviço chamado de *conference*, e cada uma delas possui um JabberID associado a ele(ex. *grupodestreaming@conference.lprm.inf.ufes.br*). Para interagir cada participante deve inicialmente ingressar no grupo (através de *stanza presence*) e, em seguida, fornecer o JabberID da sala de chat como destinatário no momento de enviar uma nova mensagem. Todas as mensagens enviadas a um grupo são encaminhadas a todos os seus membros.

A primitiva de *presença* permite ter conhecimento sobre a disponibilidade de usuários XMPP, funcionando de maneira similar a um mecanismo de *keepalive*. A primeira etapa para obter informações de disponibilidade de um determinado usuário é subscrever-se a ele, isto equivale a adicioná-lo à sua lista de contatos. O conjunto de subscrições é chamado de *roster*. Mesagens de presença são enviadas automaticamente por cada usuário XMPP em direção do servidor XMPP. Em seguida, para cada presença de um determinado usuário, o servidor encaminha-as a todos os demais usuários que estão subscritos no primeiro. Quando um usuário encerra a sua sessão XMPP, os seus subscritos são informados.

Por último, a primitiva *IQ* (*Info/Query*) é baseada em requisições e respostas, similar a protocolos como o HTTP, ditos como síncronos. Por outro lado, stanzas para *mensagens* e *presença*, são ditas assíncronas (o usuário não aguarda por respostas a mensagens enviadas). O objetivo das IQs é o de obter informações disponíveis sobre as entidades XMPP. No caso de serviços, tomando o exemplo do serviço *conference* para o chat em grupo, é possível "perguntar" quantas salas de chat em grupo existem e para cada sala, quem são os seus membros.

Ainda é possível requisitar informações a demais usuários/clientes XMPP, como no caso de atualização das coordenadas geográficas. Isto é interessante em nosso protótipo, pois os elementos de nossa arquitetura (ex. o MCG) podem ter associados a eles um JabberID e, por meio de IQs, os seus respectivos clientes (ex. clientes MCG) são capazes de requisitar os serviços oferecidos pelo MCG.

4.2 O protótipo implementado

O protótipo desenvolvido contempla as funcionalidades de agrupamento e colaboração propostas neste trabalho. Ele está disponível em repositório online, usando o sistema de versionamento Git por meio da plataforma Github. As implementações dos componentes no lado servidor (MCG, correlacionador) estão disponíveis no endereço https://github.com/callesjuan/ninjalprm e, o aplicativo mobile está disponível em https://github.com/callesjuan/ninjabubble.

As decisões de tecnologias adotadas levaram em consideração os desafios tecnológicos mencionados no capítulo 1, incluindo o acesso limitado à banda larga, necessidade de infraestrutura de *live streaming,* entradas e saídas para usuários limitadas ou reduzidas, etc.

Os componentes de arquitetura no lado servidor foram implementados em linguagem Python. Para o cliente mobile usamos a plataforma Android, principalmente pela sua ampla popularidade em mercado e também na comunidade de desenvolvedores. Ambos os lados da arquitetura adotam a tecnologia XMPP na troca de informações de todas as etapas de nossa proposta (fornecimento e

percepção de contexto, formação de grupos e colaboração). As bibliotecas Smack e SleekXMPP foram usadas para Android e Python respectivamente, ambas fornecendo alguma API para criar, enviar e receber mensagens XMPP de maneira intuitiva.

No início do projeto o XMPP, mais especificamente o serviço *conference,* seria usado apenas como instância de serviços de colaboração (elementos da arquitetura proposta), possibilitando o chat em grupo e envio de notificações de mapa. Entretanto, em razão de sua versatilidade, o XMPP também foi adotado na troca de mensagens de contexto e grupos, com o Mapeador de Contexto e Grupos (MCG). De forma alternativa, esta troca poderia ser realizada usando protocolos baseados em requisições e respostas, como o HTTP.

De fato, o XMPP possui flexibilidade para lidar com diferentes padrões de comunicação, alguns dos quais presentes em nossa proposta. Na forma *um-pra-um*, por exemplo, o protocolo possibilita mensagens síncronas e assíncronas, respectivamente em *stanzas* do tipo IQ e Message. Por outro lado, na forma *muitos-para-muitos* além de chat *multi-user* pode usar-se também o padrão *pubsub*.

Dentre os componentes do lado servidor, o Mapeador de Contexto e Grupos (MCG) consiste de um *daemon* implementado em ambiente Python. Ele aguarda por tentativa de comunicação de *streamers* por meio de *mensagens de contexto* ou *mensagens de grupo*, apresentadas neste trabalho no capítulo anterior. Para isso, além de *streamers*, também é atribuído um JabberID ao MCG.

O modelo de dados instanciado no protótipo é mostrado na Figura 13. Percebe-se que o compartilhamento de contexto é desassociado da transmissão de vídeo, ambos vinculados a um *streamer*. Para recuperar e apresentar uma transmissão ao vivo junto às informações disponíveis em compartilhamentos de contexto, basta que eles sejam associados temporalmente. Ou seja, dado o intervalo de duração de um compartilhamento de contexto, obter, no canal associado a um *streamer* (ex. no Twitcasting), o conteúdo de vídeo produzido no mesmo intervalo.

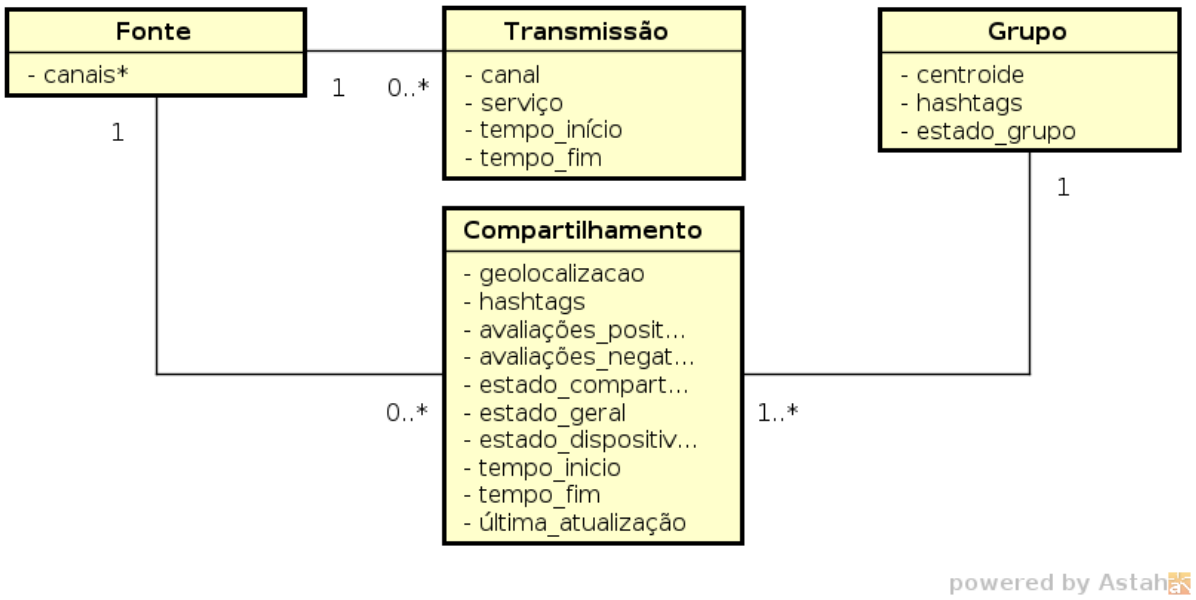

Figura 13: Modelo de dados

O MCG armazena informações sobre compartilhamentos de contexto e grupos no MongoDB, uma base de dados não relacional [MONGO 2015]. As informações contidas na base são atualizadas quando recebidas mensagens de contexto ou grupos. As mensagens de atualização de contexto implementadas foram: *atualizar hashtags* (fornecidas pelo *streamer*) e *atualizar localização* (coordenadas geográficas e orientação de câmera obtidas do GPS e sensores de um smartphone).

No MongoDB, usamos uma primitiva chamada de *near*, a qual, dado um par de coordenadas geográficas e uma distância máxima e mínima, recupera elementos pertencentes a essa vizinhança. Usando as coordenadas do próprio *streamer*, podemos então obter todos os grupos próximos dele mesmo quando, por exemplo, enviada uma mensagem de *encontrar grupos*.

O *correlacionador* de grupos é executado como parte do MCG. Ele varre periodicamente os grupos ativos identificando aqueles que participam de um mesmo evento, baseando-se em localização geográfica. A identificação de grupos próximos é realizada usando o algoritmo de agrupamento [SCIKIT 2014], onde um

agrupamento é identificado por sequencias de grupos cujos centroides estão dentro de um determinado limite.

Ao identificar agrupamentos, para cada um deles, o *correlacionador* toma um grupo como referência (pivô) e em seguida envia, via XMPP, uma *mensagem de sugestão de grupo* a todos os membros dos demais grupos. O primeiro critério para a definição do pivô foi a quantidade de membros no grupo e, em segundo lugar, o tempo agregado de transmissão (a soma de tempo da duração de transmissão de todos os membros do grupo).

Os serviços de colaboração (chat em grupo e notificações em mapa) consistem em instâncias do serviço *conference* do servidor XMPP. Em particular, para cada grupo no MCG são associadas duas salas de chat (XEP-0045), em uma trocam-se mensagens do serviço de chat em grupo e, na outra, notificações em mapa.

Os componentes da arquitetura no lado cliente foram implementados na plataforma Android. O lado cliente consiste basicamente de um gerenciador de interface com o usuário e o cliente MCG, que implementa a captura de dados de sensores para representar o contexto de uma transmissão e lida com o envio e recebimento de mensagens relacionadas a compartilhamentos de contexto (ciclo de vida e atualização de contexto) e grupos.

Como propomos funcionalidades que estendem aquelas disponíveis em plataformas de *live streaming*, desenvolvemos um protótipo que pode ser usado em conjunto com qualquer uma delas. Isso permitiu que o projeto se concentrasse exclusivamente nos aspectos de contexto, grupos e colaboração entre *streamers*, aproveitando as ferramentas e infraestrutura já disponibilizadas pelas plataformas de *live streaming*. Esta opção também permite que os compartilhamentos de contexto possam ser desacoplados dos fluxo de vídeos gerados.

O protótipo mobile foi então desenvolvido a partir da API de serviços do ambiente Android, ao invés do uso típico de *Activities* [GOOGLE e OPEN HANDSET ALLIANCE 2016]. Tal medida foi necessária porque no ambiente Android não é

possível executar duas *Activities* simultaneamente (ex. o aplicativo de captura do Twitcasting e o nosso protótipo).

Apesar da API de serviços do Android não dispor do gerenciamento automático da interface com usuário (UI), tal como ocorre com *Activities*, contornamos isto fazendo uso direto da interface *WindowManager*. Ao acessar diretamente a interface *WindowManager* é possível introduzir quaisquer elementos de *Layout* típicos de um aplicativo baseado em *Activities*, mesmo que outro aplicativo esteja em executando em primeiro plano. Esta técnica é similar à usada por aplicativos como o Facebook Messenger (por meio das Chat Heads) e o Link Bubble [LINKBUBBLE 2014]. Em compensação, o desenvolvimento de interfaces com o usuário diretamente a partir do *WindowManager* é mais trabalhoso do que a construção de aplicativos com as convencionais *Activities*.

Para conciliar as interfaces gráficas do aplicativo mobile (ex. TwitCasting) e do protótipo, definimos, para o segundo, dois modos de visualização: escondido e sobreposto. No modo escondido (ver Figura 14.a) o destaque ou prioridade é no aplicativo de captura, assim é presente na maior parte do tempo. Nele, além do aplicativo mobile, mantém-se um ícone que, quando selecionado, altera para o modo de visualização sobreposto. No modo sobreposto (ver Figura 14.b), as funcionalidades propostas e instanciadas no protótipo são de fato apresentadas. Na barra superior de ícones é possível acessar, da esquerda para a direita: (i) as opções de contexto e grupos (“tela” inicial); (ii) a visualização de mapa (incluindo membros do grupo e notificações em mapa); e (iii) o serviço de chat em grupo. O último ícone (iv), quando selecionado, alterna novamente para o modo escondido.

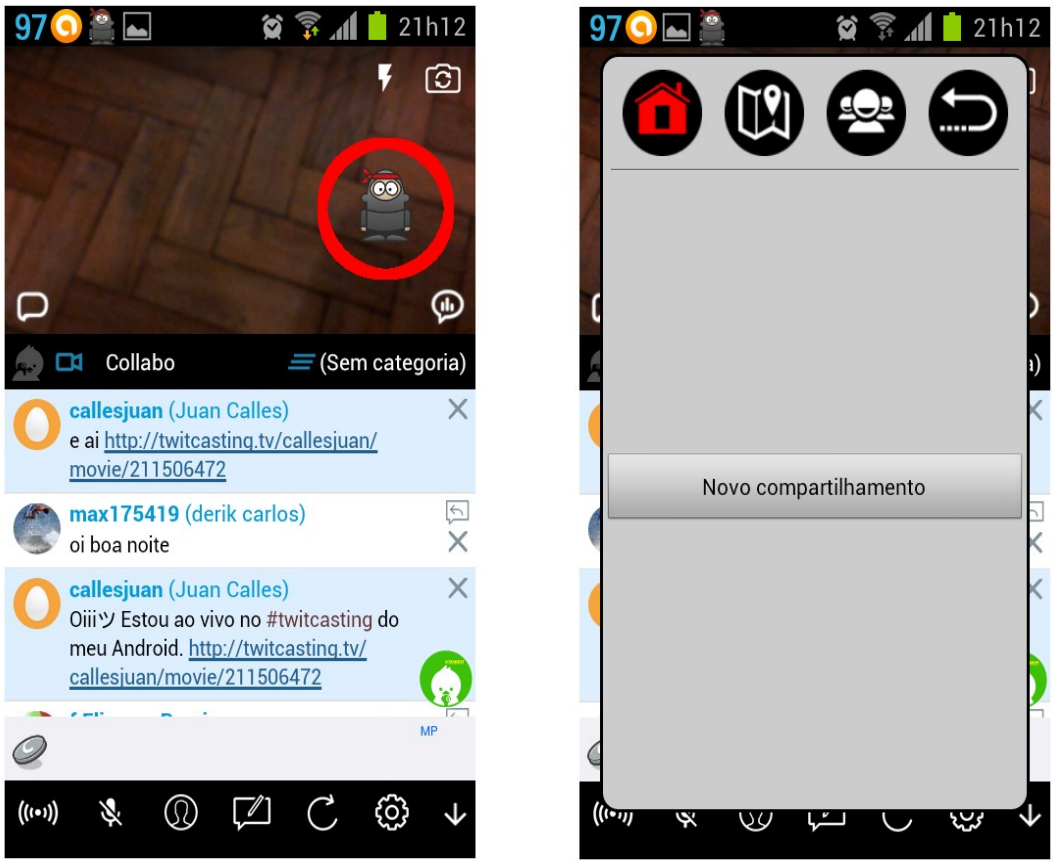

Figura 14: Protótipo e aplicativo de streaming (Twitcasting) executando simultaneamente

O ciclo de vida do compartilhamento de contexto é manipulado pelo *streamer* na interface de usuário do protótipo. Conforme ilustrado na Figura 15.a, as opções de iniciar (*initiate*), parar (*pause*), continuar (*resume*) e encerrar (*close*), se tornam visíveis de acordo com o estado atual do compartilhamento de contexto. O envio e recebimento de mensagens do ciclo de vida de um compartilhamento de contexto são feitos pelo cliente MCG, por meio de protocolo XMPP.

Quando o compartilhamento de contexto está em estado ativo a visualização de mapa (Figura 15.b) e o chat em grupo tornam-se acessíveis na barra superior de ícones do protótipo. Na visualização de mapa é possível encontrar pontos de referência (ruas, avenidas, praças, etc.). Ela foi implementada usando a biblioteca OSMDroid, baseada em informações do OpenStreetMap.

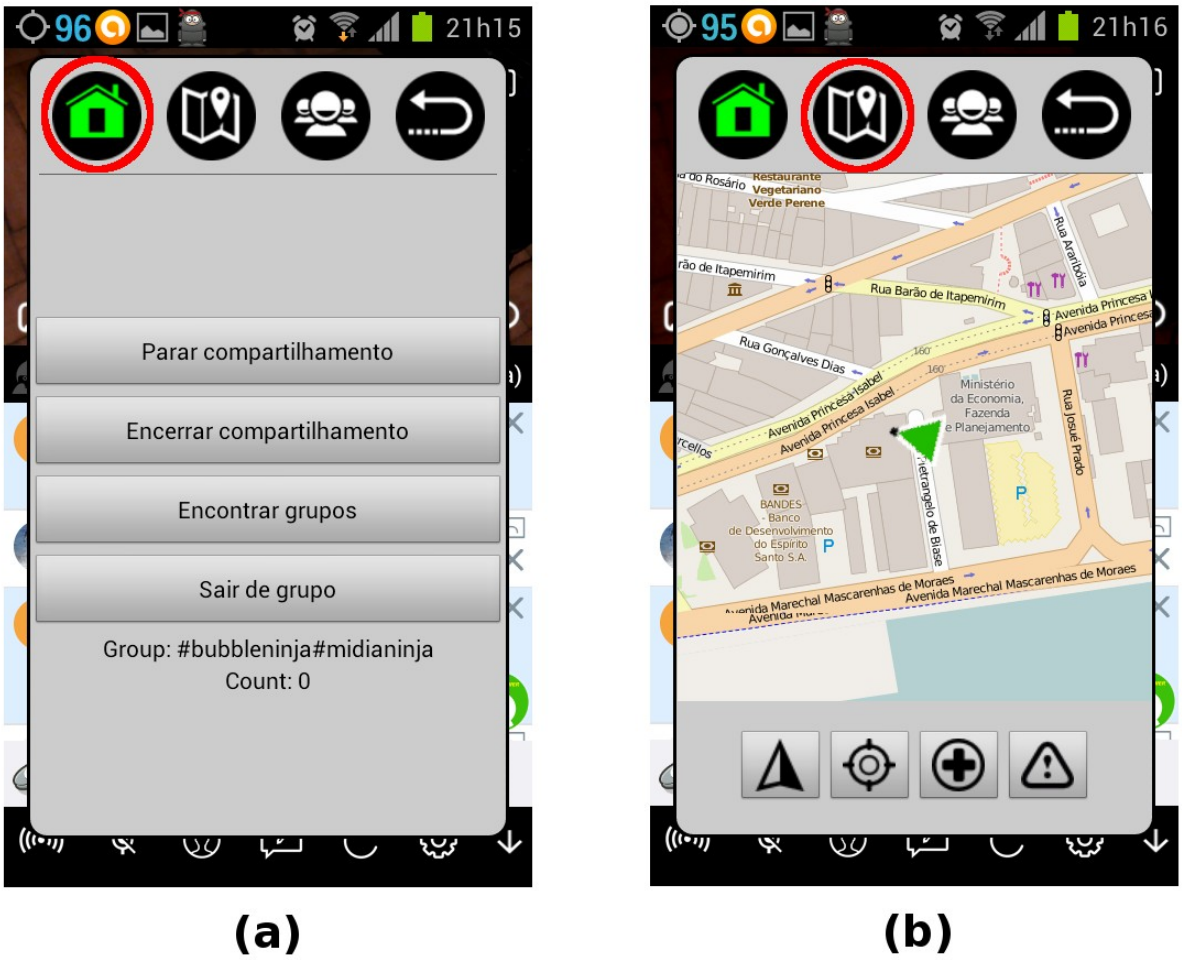

Figura 15: Compartilhamento de contexto no protótipo

As informações de localização geográfica e orientação de câmera são capturadas a partir do instante que um compartilhamento de contexto entra no estado ativo. A captura é realizada por *listeners* e *timers*, que fazem parte do cliente MCG. *Hashtags* são fornecidas ao iniciar um novo compartilhamento de contexto. Os *timers* enviam periodicamente dados de contexto atualizados para o MCG via XMPP, nas chamadas mensagens de atualização de contexto (ex. *atualizar localização* e *atualizar hashtags*).

Figura 16: Agrupamentos no protótipo

Ainda no menu de contexto e grupos, ao selecionar a opção “encontrar grupos” (*find groups*) é solicitado ao MCG que recupere todos os grupos ativos localizados nas redondezas (inicialmente consideramos redondeza 1km de distância). Quando a informação sobre grupos ativos forem recebidas pelo cliente MCG, uma caixa de diálogo é mostrada ao *streamer*, conforme mostrado na Figura 16.a. Ela apresenta uma lista de todos os grupos disponíveis, a *hashtag* agregada e a quantidade de membros de cada um deles. Ao selecionar um grupo e confirmar o desejo de ingressar nele, o *streamer* abandona o seu grupo atual e migra para o grupo selecionado.

Ao receber uma *sugestão de agrupamento* do *correlacionador* usamos a

barra de notificações do Android para destacá-la ao *streamer*, informando, por exemplo, “entre no grupo #MarchaNoCentro”. Ao selecioná-la, uma nova caixa de diálogo é apresentada ao *streamer*, perguntando se deseja ou não ingressar no grupo sugerido. Caso a sugestão não seja aceita a notificação é então descartada.

Uma vez dentro de um grupo, quando a visualização em mapa é selecionada, é possível visualizar a localização geográfica dos membros do grupo, representados por círculos azuis na Figura 16.b.

Os últimos aspectos relevantes do protótipo são os serviços de notificação em mapa e chat em grupo, ilustrados, respectivamente, nas Figuras 17.a e 17.b.

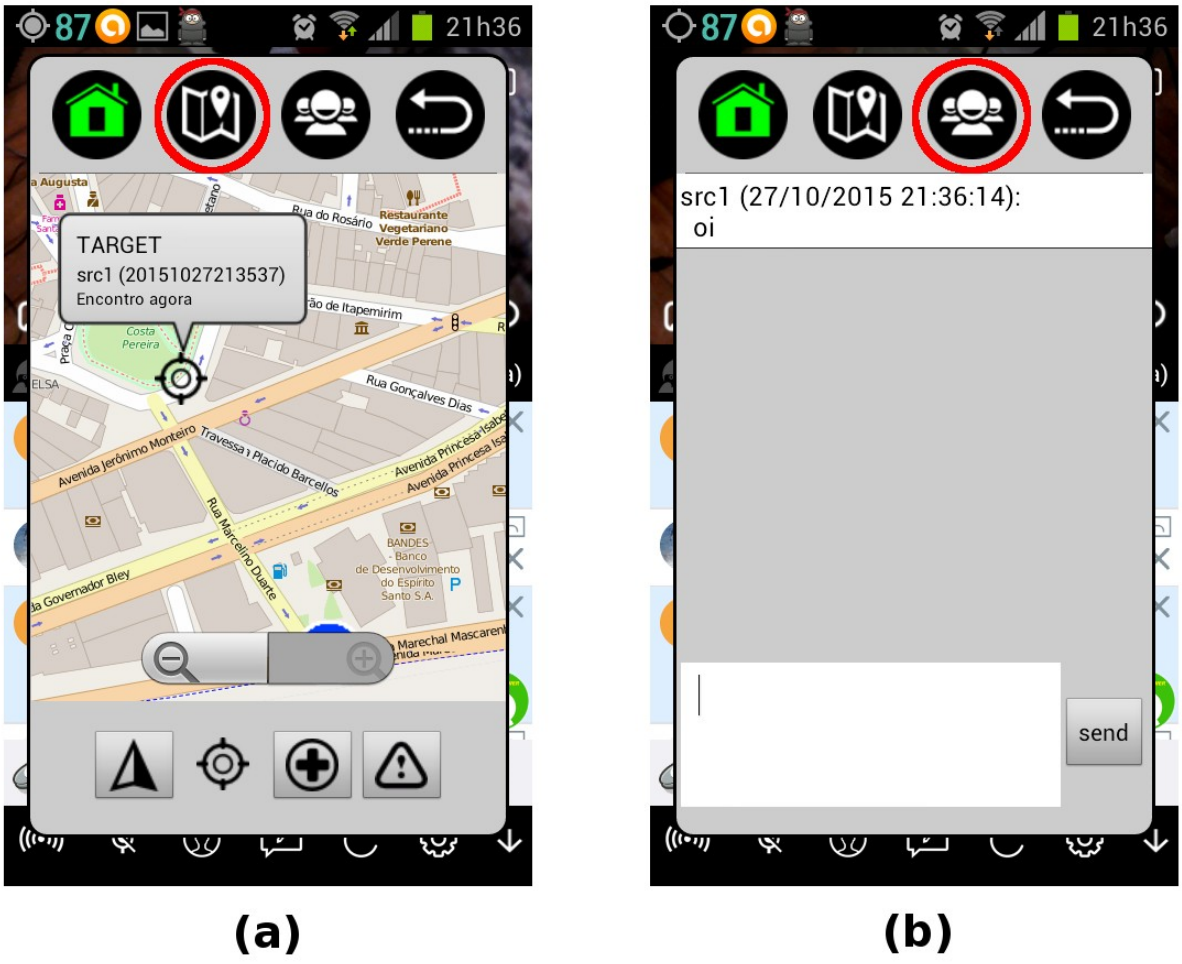

Figura 17: Colaboração entre membros de um grupo no protótipo

Na visualização em mapa (Figura 17.a), a criação de notificações em mapa inicia nos ícones localizados na parte inferior da interface. Da esquerda para a direita temos as opções de: (i) localizar, retornando o foco do mapa para a localização do próprio *streamer*; (ii) notificação de alvo; (iii) notificação de ajuda; e (iv) notificação de perigo.

Na Figura 18.a, num cenário hipotético, um *streamer* avista o Batalhão de Missões Especiais (BME) da Polícia Militar. Em seguida, ele decide informar os demais participantes usando a extensão de colaboração, Figura 18.b. Após selecionar a opção de notificação em mapa, as coordenadas geográficas da posição tocada no mapa são incluídas na notificação a ser enviada. Em seguida, apresenta-se uma caixa de diálogo ao *streamer,* onde é possível que informar uma descrição mais específica sobre a notificação. Na Figura 18.c, foi selecionada a notificação de perigo e a descrição “BME tá chegando” fornecida. Por fim, a notificação é enviada via XMPP para os colegas de grupo e, ao ser recebida, é incluída como um ícone no mapa visualizado pelos participantes do grupo. Quando uma notificação recebida é selecionada, abre-se uma pequena seção que mostra a identidade do seu autor e a descrição fornecida (Figura 18.d).

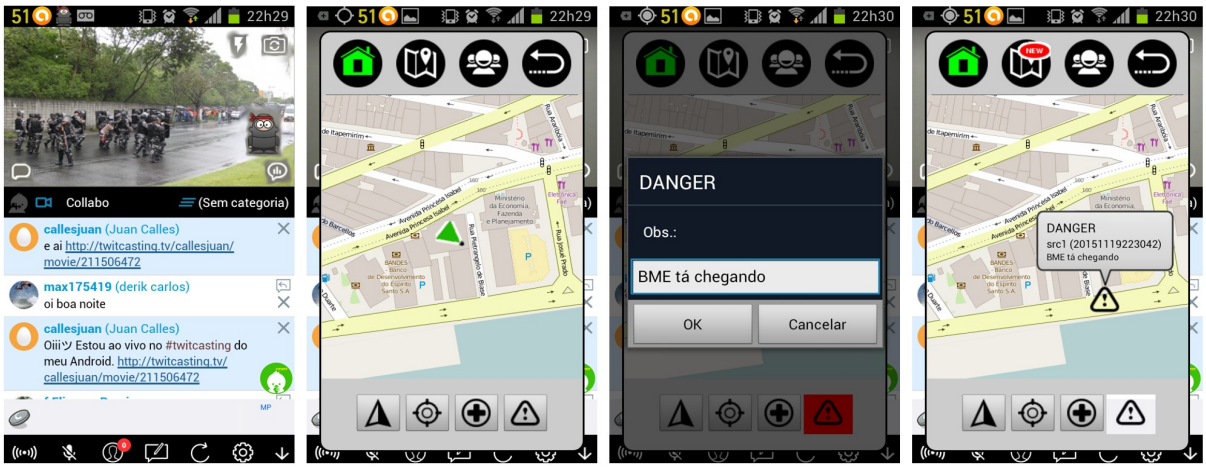

Figura 18: Envio de uma notificação de perigo

As notificações são temporárias e, após um pequeno intervalo de tempo,

são removidas do mapa. Inicialmente definimos um tempo de vida de cinco minutos para cada notificação recebida. Entretanto, a estratégia de tratamento de múltiplas notificações pode ser alterada sem maiores problemas.

Por último, a funcionalidade de chat em grupo (Figura 17.b) consiste de uma interface típica de chat, com uma lista de mensagens trocadas entre membros do grupo, ordenadas por data e hora em que foram enviadas. Cada mensagem apresenta também a identificação do seu autor. Na parte inferior, uma caixa de texto para compor novas mensagens e um botão para enviá-las são apresentados.

Ambas as notificações em mapa e o chat em grupo, quando recebidas, são destacadas para o *streamer*. Por exemplo, o protótipo foi configurado para vibrar ao receber qualquer mensagem relacionada aos serviços de colaboração. Mesmo quando o protótipo está em modo escondido o seu ícone fica destacado até que o conteúdo recebido seja visualizado. Na barra superior do protótipo, já no modo sobreposto, quando uma notificação de mapa é recebida destaca-se o ícone de acesso para a visualização de mapa e, quando uma mensagem do chat em grupo é recebida, o ícone destacado é o do chat.

4.3 Avaliação experimental do protótipo

O objetivo dos experimentos foi avaliar o nosso protótipo com relação aos seguintes aspectos:

1. Atendimento aos requisitos funcionais estabelecidos: a oferta de *serviços de colaboração* utilizando uma estratégia desacoplada entre esses serviços e os da plataforma de *streaming* de vídeo;

2. Estimar o impacto, em termos de gasto de bateria, da inclusão dos serviços de colaboração associados ao *live streaming* e;

3. Estimar o retardo temporal (*delay*) compreendendo o instante de envio e o instante de recepção das notificações na estratégia de cooperação implementada para o *live streaming*.

Para cada um desses aspectos foi realizado um tipo de experimento específico, cujos resultados são detalhados nas próximas subseções.

4.3.1 Primeiro experimento

O primeiro experimento testou o protótipo funcionalmente e foi realizado por três participantes, sendo dois voluntários e o autor deste trabalho, nenhum dos quais com experiência de coberturas ao vivo de vídeo. Ele foi realizado no Centro da cidade de Vitória (ES), dentro de um raio de distância de 120 metros. Os dispositivos smartphones usados na experiência foram um Samsung Galaxy S II (I9100) e dois HTC One M7, todos usando conexões 3G a 1Mbps com boa intensidade de sinal.

Primeiramente, uma apresentação foi feita aos dois participantes voluntários, abordando as plataformas de *live streaming*, aplicativos de captura, os conceitos propostos neste trabalho (compartilhamentos de contexto, grupos e colaboração) e as funcionalidades oferecidas no protótipo. Em seguida, o experimento a ser realizado foi descrito.

O experimento consistiu em completar um simples cenário predefinido. Se o cenário fosse concluído em todas as suas etapas então o teste seria considerado como bem sucedido, caso contrário seria considerado como fracassado.

O cenário consistiu nas seguintes etapas:

1. *Setup*: Consistiu da atribuição aleatória da localização de início do experimento para cada voluntário e, em seguida, escolha, também aleatória, dentre os participantes, de um *notificador* (alguém responsável por enviar uma notificação de mapa instantes após o início do experimento). A identidade do notificador não seria revelada aos demais participantes.

2. Início de experimento: Voluntários deslocam-se em direção ao local aleatoriamente designado para início. Ao chegar ao local de partida, cada participante deveria imediatamente iniciar um novo compartilhamento de

contexto usando o protótipo e, em seguida, iniciar uma nova transmissão de vídeo, usando qualquer aplicativo de captura. Foi pedido que capturassem qualquer conteúdo que desejassem.

3. A formação de um grupo: Logo após iniciar o compartilhamento de contexto e o *live streaming*, os participantes foram instruídos a aguardar e aceitar mensagens de sugestão de grupo enviadas pelo *correlacionador*. O objetivo no fim desta etapa foi fazer com que todos os voluntários pertencessem ao mesmo grupo.

4. O chat em grupo: Imediatamente após ingressar no grupo sugerido, os participantes foram instruídos a manifestar-se no chat de grupo. Dessa forma todos confirmariam que o grupo do experimento foi formado com sucesso.

5. A notificação em mapa: Finalmente, o *notificador,* antes selecionado randomicamente, deveria enviar uma notificação de ajuda (que informa a sua própria localização), e os demais participantes deveriam deslocar-se para então finalizar o experimento.

Para esse experimento o mesmo cenário foi repetido quatro vezes. Em cada uma das repetições, as localizações de início e o *notificador* (a etapa de *setup*) foram selecionados aleatoriamente.

No final de sua execução, três (entre quatro) tentativas foram bem sucedidas em completar o cenário predefinido. Adicionalmente, como *feedback* geral, os voluntários mencionaram não terem tido dificuldades em lidar com o protótipo, além do fato de não terem a certeza do instante em que receberiam a mensagem de *sugestão de grupo*.

De fato, na segunda repetição do primeiro experimento, o *correlacionador* não foi iniciado corretamente (erro de configuração) e assim as mensagens de *sugestão* não foram enviadas. Entretanto, quando perceberam que havia um problema no envio ou recebimento da *sugestão de grupo*, e como as funcionalidades do protótipo haviam sido previamente apresentadas, eles foram

capazes de manualmente utilizar as funcionalidades de *encontrar grupos* e *entrar em grupo*, formando o grupo de todos os voluntários com sucesso.

4.3.2 Segundo experimento

O segundo experimento também testou o protótipo funcionalmente, mas com um cenário mais complexo em termo das atividades e do grau de colaboração. Ele foi realizado no Campus de Goiabeiras da Universidade Federal do Espírito Santo (UFES). A experiência foi realizada por 6 participantes, incluindo cinco voluntários (estudantes de pós-graduação em Informática da própria instituição), além do autor deste trabalho, nenhum dos quais com experiência na realização de coberturas de vídeo. Os requisitos desejáveis para participação foram possuir um smartphone com as seguintes especificações: Sistema operacional Android com versão mínima de 2.3 (*Gingerbread*), câmera, GPS e acesso à Internet (via 3G, de preferência).

A respeito do acesso à Internet, um dos participantes advertiu não possuir acesso à Internet via redes móveis, mas decidiu participar do experimento usando a rede de acesso *eduroam* (*education roaming*) instalada ao longo da UFES, mesmo estando em desvantagem quando transitasse em locais fora do alcance dessa rede.

Opcionalmente, solicitou-se a instalação de algum aplicativo de gravação de tela, pois nem todas as versões do sistema operacional disponibilizam essa funcionalidade no repositório de aplicativos padrão. O objetivo dessa solicitação era permitir o registro do comportamento dinâmico da aplicação de colaboração durante a realização do experimento.

O protótipo foi distribuído e instalado em cada dispositivo e, antes de iniciar o experimento, as suas funcionalidades foram brevemente apresentadas aos voluntários.

Ao invés de realizar um *live streaming* por participante, usando uma plataforma como Twitcasting, optou-se apenas por que os vídeos fossem gravados usando o aplicativo padrão de captura do sistema operacional. Essa decisão foi

tomada por a maioria dos voluntários não possuírem um plano de acesso à Internet móvel que permitisse realizar o *streaming* de vídeo sem custo.

Similar ao primeiro experimento, o segundo experimento consistiu na realização de um cenário pré-definido, se todos os participantes conseguissem concluí-lo, então seria considerado sucesso. Nesse cenário, que é descrito a seguir, o autor deste trabalho assumiu o papel de coordenador, iniciando as interações que representaram a colaboração. Além disso, todos os voluntários foram previamente agrupados durante a apresentação das funcionalidades do experimento.

O cenário consistiu nas seguintes sete etapas:

1. *Setup*: Todos os participantes, incluindo o coordenador, iniciaram o protótipo e, em seguida, a gravação de vídeo. Nesse primeiro instante todos encontravam-se no mesmo local dentro da universidade.

2. Definição do 1º alvo: O coordenador enviou em mapa uma notificação de alvo a todos, informando um novo local onde eles deverão se reencontrar. A distância máxima para o novo local é de 500 metros em relação ao local inicial.

3. Deslocamento até o primeiro alvo: Os participantes deslocaram-se até o primeiro alvo usando caminhos distintos, incluindo o próprio coordenador. Além do vídeo do trajeto que é gravado, recomendou-se aos participantes que acompanhassem em mapa a localização do grupo e interagissem no chat caso desejado. Para dar continuidade ao experimento era preciso que todos os participantes se encontrassem no local alvo, assim os primeiros a chegar deviam aguardar a chegada dos outros participantes.

4. Definição do 2º alvo: Depois de reencontrados os participantes, o coordenador informou que a mesma dinâmica seria repetida mais uma vez, enviando em seguida uma nova notificação de alvo informando o seguinte local onde eles deveriam se reencontrar.

5. Deslocamento para o segundo alvo: Os participantes deslocaram-se em

direção ao segundo alvo usando caminhos distintos. Dessa vez, entretanto, durante o seu deslocamento, o coordenador acompanhou com maior atenção a localização dos demais participantes, esperando o momento em que estivessem aproximadamente na metade do caminho.

6. 3º alvo: Quando os participantes chegaram na metade do caminho, o coordenador enviou uma nova notificação de alvo, alterando o local de encontro. O objetivo dessa notificação foi inserir um elemento de imprevisibilidade, pois à exceção do coordenador, os voluntários não sabiam previamente que surgiria um terceiro alvo substituindo o segundo. O local de encontro foi escolhido de forma que o coordenador fosse o primeiro a chegar nele.

7. Deslocamento até o 3º alvo e fim: Ao receber a notificação de alvo, cada participante suspendeu o segundo alvo e então deslocou-se para o terceiro e último alvo. Finalmente, quando cada participante chegou ao local de encontro, o coordenador, que foi o primeiro a chegar, solicitou que encerrassem a gravação de vídeo e o protótipo.

Ao final do experimento um questionário a respeito de aspectos funcionais e de usabilidade foi entregue a cada voluntário. As perguntas do questionário estão presentes na seção de anexos, no final deste trabalho. O questionário foi divulgado aos participantes usando a ferramenta online Google Forms.

O experimento foi concluído com sucesso, isto é, todos os participantes conseguiram concluir o cenário pré-definido, encontrando-se nos locais de encontro definidos pelo coordenador e informados usando notificações em mapa. De acordo com as respostas do questionário (presentes na seção de anexos), em relação a aspectos funcionais, todos os participantes receberam a notificação de mapa para o primeiro alvo e apenas um participante não recebeu as notificações para o segundo e terceiro alvos.

Todos os participantes também declararam ter recebido e conseguido enviar mensagens de texto no chat em grupo. Dentre os seis participantes, cinco

declararam ter usado o chat para confirmar informações sobre os alvos estabelecidos e três participantes declararam usá-lo por curiosidade. Todos os participantes declararam conseguir perceber o deslocamento dos demais participantes na visualização em mapa.

O participante que não recebeu as duas últimas notificações de alvos foi o mesmo que dependeu apenas da rede de acesso Wi-Fi. Apesar disso, ele foi capaz de concluir o cenário possivelmente interpretando os alvos no chat em grupo ou identificando o deslocamento dos demais participantes na visualização em mapa.

Em relação a falhas no protótipo, dois participantes declararam que o aplicativo deixou de funcionar ao menos uma vez. Um participante declarou que o aplicativo parou uma vez no momento que tentou iniciar as gravações simultâneas de vídeo e tela. Outro participante declarou que o protótipo deixou de funcionar várias vezes, porém tratava-se do participante que participou do experimento usando apenas a rede Wi-Fi da Universidade. Mesmo assim, o participante conseguiu interpretar as mensagens de colaboração e concluir o cenário proposto.

Aspectos avaliados de usabilidade e visuais são apresentados na Tabela 1, que representam as questões 10, 11, 12 e 13 do questionário pós-experimento. As mesmas usam uma escala Likert de 1 a 5 (onde 1 é ruim e 5 é bom) para avaliar cada aspecto. Em termos de usabilidade (Q10) o protótipo recebeu uma avaliação média de 3,5. A precisão da localização própria no mapa (Q11) e a precisão da localização dos colegas de grupo (Q12) receberam respectivamente uma avaliação média de 4 e 3,8. Destacamos que ambas as questões Q11 e Q12 possuem forte dependência dos serviços de geolocalização (GPS). Por último, a taxa de atualização dos ícones que representam *streamers* ou notificações em mapa (Q13) foi avaliada em 3,3.

	P1	P2	P3	P4	P5	P6	Média
Q10	3	4	4	2	5	3	3,5
Q11	3	4	5	4	5	3	4
Q12	3	4	4	4	5	3	3,8
Q13	3	4	5	2	3	3	3,3

Tabela 1: Resultados avaliados sobre usabilidade e interface visual

Além desses aspectos quantitativos de interface visual e usabilidade, seguem algumas sugestões apontadas pelos participantes do experimento: refinamento de telas, diminuição do tamanho dos ícones que representam *streamers*, remoção e filtros de notificações em mapa, a possibilidade de centralizar o mapa em notificações de alvo.

4.3.3 Terceiro experimento

O terceiro experimento testou o consumo de bateria pela aplicação instalada no smartphone que seria responsável pelo *live streaming*. Ele consistiu em medir o intervalo de tempo que demorava para que a bateria tivesse seu nível reduzido em cinco por cento, em seis configurações diferentes, sendo cada uma delas repetidas por três vezes. Em cada teste, recarregamos totalmente (cem por cento de carga) o aparelho smartphone e finalizamos o teste assim que o nível de bateria atingisse o valor de noventa e cinco por cento. A quantidade de bytes enviados e recebidos também foi medida.

Para esse experimento usamos um aparelho Samsung Galaxy S II (I9100) com sistema operacional Android *Ice Cream Sandwich*. Além disso, *streamers* simulados foram implementados para representar a interação de grupo de maneira simples (sem a necessidade de alocar voluntários) e testar o seu impacto em consumo de bateria. Os *streamers* simulados são capazes de enviar e receber mensagens de *contexto e grupos* (interagindo com o MCG) e também mensagens de colaboração (chat de grupo e notificações de mapa).

As seis configurações testadas foram as seguintes:

1. A primeira configuração, ou configuração padrão, consistiu apenas em executar o aplicativo de *live streaming* sem usar o nosso protótipo, mantendo uma transmissão ao vivo. O objetivo foi estimar valores de referência sobre o consumo de bateria sem o uso de funcionalidades adicionais de colaboração. O aplicativo de *live streaming* usado foi o Twitcasting.

2. A segunda configuração, assim como as demais configurações, incluiu o uso de compartilhamentos de contexto por meio do nosso protótipo e, para cada uma das configurações restantes, foi coletada também a quantidade de bytes trocados (enviados e recebidos) com o MCG e *streamers* simulados (quando houvesse). Especificamente, a segunda configuração consistiu apenas em manter um compartilhamento de contexto ativo e simultaneamente realizar uma transmissão ao vivo usando o Twitcasting.

3. Para a terceira configuração cinco *streamers* simulados foram adicionados, formando um grupo de seis membros ao todo. Para essa configuração não houve troca de mensagens de colaboração. O objetivo era apenas obter o custo da troca de mensagens de contexto e grupos (atualização da localização dos membros de um grupo), além da troca de mensagens *Presence* que fazem parte do protocolo XMPP.

4. Na quarta configuração mantiveram-se os cinco *streamers* simulados, mas, dessa vez, eles foram configurados para enviar mensagens no chat em grupo a cada minuto, simulando a interação com o grupo.

5. A quinta e a sexta configurações foram similares à terceira e à quarta, respectivamente, mas, ao invés de adicionar cinco *streamers* simulados, foram adicionados dez.

A Tabela 2 resume as seis configurações usadas no experimento e, como mencionado anteriormente, para cada uma delas foram realizados três testes. A escolha de cinco *streamers* é consequência do número máximo encontrados (até o

nosso conhecimento) de transmissões de canais ligados à Mídia NINJA na plataforma Twitcasting, usando o seu mecanismo de busca padrão. Para as configurações 5 e 6 apenas dobramos a quantidade anterior.

O servidor Openfire, adotado como implementação dos serviços de colaboração, por padrão limita a quantidade de usuários em um grupo em trinta, valor acima dos números de transmissões simultâneas encontradas. Entretanto, caso desejado, essa quantidade pode ser modificada para valores significativamente maiores vinculando o servidor a servidores de bancos de dados de terceiros.

Configuração	***Default***	**2ª**	**3ª**	**4ª**	**5ª**	**6ª**
Usou *Twitcasting*	Sim	Sim	Sim	Sim	Sim	Sim
Usou o Protótipo	Não	Sim	Sim	Sim	Sim	Sim
Tamanho do grupo	N/A	1	5	5	10	10
Interações de grupo	N/A	Não	Não	Sim	Não	Sim

Tabela 2: Configurações usadas no terceiro experimento

Com as medições realizadas para cada configuração, o tempo médio gasto no consumo dos cinco por cento de nível de bateria e a quantidade média de dados enviados e recebidos são mostrados na Tabela 3.

Percebe-se que a sexta configuração é a que leva ao consumo mais rápido de bateria, exatamente pelo maior número de mensagens trocadas entre os membros do grupo. Assim, no caso mais extremo do experimento, o maior tempo de cobertura perdido, ao usar o nosso protótipo, foi de aproximadamente 13,4 % em relação à configuração padrão. A segunda pior perda de tempo de cobertura ocorreu na quarta configuração, correspondendo a aproximadamente 11% também em relação à primeira configuração.

Configuração	Padrão	2ª	3ª	4ª	5ª	6ª
Tempo	9min16s	8min41s	8min33s	8min14s	8min26s	8min1s
Enviados (KB)	-	4,5	4,1	4,9	4,2	4,0
Recebidos (KB)	-	4,2	7,7	40,4	9,0	50,8

Tabela 3: Resultados do experimento de consumo de bateria

É também evidente, e de se esperar, que ambas as quarta e sexta configurações, que foram as que mais consumiram energia, foram também as configurações com maior quantidade média de dados trocados, correspondendo a 40,4 e 50,8 KBytes recebidos, respectivamente, devido à troca de mensagens simulada entre elementos do grupo.

Embora as outras configurações não tenham lidado com a transferência contínua de dados, elas ainda gastaram bateria mais rapidamente do que a configuração padrão. Aspectos conhecidos sobre a implementação podem justificar esse comportamento, dentre eles o uso constante do GPS, o envio periódico de mensagens para o MCG (ex. *atualizar localização*) via XMPP, o próprio envio de mensagens de tipo *Presence* que fazem parte do protocolo XMPP. Nos casos em que foi simulada a interação entre *streamers*, deve-se considerar também o acionamento do vibrador (*vibra call*) para destacar o recebimento de novas mensagens.

4.3.4 Quarto experimento

O quarto experimento testou o atraso (*delay)* na comunicação entre o aplicativo mobile do protótipo e os componentes no lado servidor. Em particular, foram testadas as mensagens de *sugestão de grupo*, *entrar em grupo* e *notificação em mapa*. Essas mensagens foram escolhidas devido ao papel que desempenham na formação de grupos e colaboração, conforme traçado no experimento funcional.

Para esse experimento usamos um aparelho Samsung Galaxy S II e um *streamer* simulado. O *delay* foi testado em redes Wi-Fi (10Mbps) e 3G (1Mbps).

Adicionalmente, o *delay* em rede 3G foi testado em dois ambientes diferentes, de acordo com a intensidade do sinal da rede (fraco e bom).

O experimento consistiu em iniciar primeiramente o *streamer* simulado e depois o *streamer* real, dessa forma o *streamer* simulado seria considerado como grupo pivô para o *correlacionador* (maior tempo de transmissão). O *streamer* real receberia então uma *sugestão de grupo*, a qual seria aceita, seguida de uma mensagem *entrar em grupo* para ingressar no grupo sugerido. Por último, o *streamer* real deveria enviar uma mensagem de *notificação em mapa*.

Durante a realização dese experimento, o qual foi repetido dez vezes para cada configuração de rede (Wi-Fi e 3G com diferentes intensidades de sinal), o *delay* foi medido para cada uma das mensagens mencionadas. Como as mensagens de *sugestão de grupo* e *notificações em mapa* são assíncronas, para medir o *delay* foi usado o tempo de *round-trip*.

Para esse último experimento, a Tabela 4 mostra as medições de *delay* médio associado a cada mensagem enviada no cenário proposto. Ao usar redes Wi-Fi e 3G com boa intensidade de sinal, a comunicação foi considerada praticamente como instantânea para todas as mensagens, sendo que o Wi-Fi obteve melhor desempenho do que o 3G.

Mensagem	**Sugestão de grupo**	**Entrar em grupo**	**Notificação em mapa**
Wi-Fi	177ms	54ms	46ms
3G (Bom)	268ms	171ms	229ms
3G (Fraco)	1021ms	1231ms	959ms

Tabela 4: Resultados do experimento de delay de comunicação

Em uma rede 3G com intensidade de sinal fraca, o *delay* de comunicação médio chegou a aproximadamente um segundo em todas as mensagens. Embora não apresentado na Tabela 4, e como é de se esperar, os testes realizados com a

intensidade de sinal fraca também apresentaram maior desvio padrão do que nos outros casos. Em alguns casos, esse desvio atingiu aproximadamente quatro segundos de atraso. Além disso, durante experimentos nessa configuração de rede, alguns problemas de perda de conexão foram observados.

4.4 Considerações

Com a realização dos experimentos relatados foi possível verificar as funcionalidades propostas de uma extensão colaborativa para plataformas de *live streaming*, apesar de limitações em quantidade de participantes (nos experimentos funcionais) e diversidade de dispositivos e plataformas (nos experimentos de consumo de bateria).

Os resultados obtidos indicam um aumento de gastos em termos de bateria (com perda de até treze por cento de tempo de cobertura) e atrasos na interação entre os componentes arquiteturais. Isto decorre da adição de recursos, tais como GPS, acelerômetro, giroscópio, além de novos recursos computacionais e largura de banda necessários ao protótipo e às novas interações entre *streamers*.

Apesar de nossa proposta depender de recursos adicionais, destacamos que as funcionalidades propostas potencialmente possibilitam a manutenção de coberturas em *live streaming* em casos de limitação de recursos (ex. bateria e acesso à Internet) e durante a ocorrência de imprevistos. Por exemplo, ao usar as funcionalidades de colaboração, um *streamer* com condições de compartilhar o acesso à banda larga poderia oferecê-lo a outros *streamers* em um mesmo evento, principalmente àqueles com dificuldade de acesso. Por outro lado, a extensão colaborativa também permite a troca de informações sobre acontecimentos que poderiam levar ao termino de uma transmissão, como o envolvimento em confrontos ou outras situações que impeçam que um *streamer* realize o seu trabalho.

5 CONCLUSÃO

Neste trabalho apresentamos uma proposta de extensão para plataformas de *live streaming* baseadas em smartphones. A extensão tem como principal objetivo permitir a colaboração entre os autores de uma transmissão de vídeo ao vivo, chamados também de *streamers*.

O problema do *live streaming* colaborativo foi abordado exatamente pelo fato de que as plataformas de *live streaming* mais populares atualmente (Livestream, UStream, Bambuser e Twitcasting) desconsideram o cenário no qual pelo menos dois *streamers*, realizando cada um uma cobertura de um mesmo evento, sejam capazes de colaborar. Embora, por exemplo, o Twitcasting ofereça a funcionalidade chamada de Collabo, o seu principal objetivo é o de aumentar a participação dos espectadores, além de possuir limitações como quantidade de participantes (máximo cinco) [TWITCASTING 2014].

A extensão para o *live streaming* colaborativo aqui proposta baseia-se no fornecimento de mecanismos de comunicação e coordenação entre *streamers* em um mesmo evento. Dessa forma, seria possível abordar alguns dos principais problemas encontrados na realização de transmissões ao vivo de eventos, desde problemas relacionados à tecnologia (ex. esgotamento de bateria, instabilidade de largura de banda) até problemas ligados à natureza do evento que está sendo transmitido. Por exemplo, em uma manifestação, o risco de deparar-se com confrontos entre manifestantes e policiais é relativamente alto.

Ao oferecer mecanismos de comunicação horizontais, isto é, entre *streamers* (uma vez que, no tempo desta dissertação, *streamers* comunicam-se apenas com os seus espectadores), é possível colaborar em uma diversidade de dimensões. A partir da comunicação é possível, por exemplo, que *streamers* em um mesmo evento coordenem a cobertura colaborativa, traçando alvos, solicitando ajuda quando necessário, informando sobre acontecimentos relevantes, etc.

Para identificar *streamers* em um evento, a nossa proposta trabalhou com o

conceito de compartilhamento de contexto e grupos espontâneos e dinâmicos. O compartilhamento de contexto corresponde a uma sessão na qual um *streamer* fornece informações que caracterizam o evento ao qual ele pertence. Inicialmente, as informações de contexto trabalhadas foram a localização geográfica e *hashtags* (pois podem representar uma descrição sobre o evento).

Em posse de informações sobre compartilhamentos de contexto, a extensão proposta pode potencionalmente correlacionar *streamers* que se encontram no mesmo evento, informando-os da existência de outros participantes. O conceito de grupos espontâneos e dinâmicos aplica-se a partir deste instante, uma vez que um *streamer* pode agrupar-se com outros quando desejado e, de mesma maneira, abandonar um grupo e migrar para outros.

Finalmente, funcionalidades de colaboração são oferecidas a *streamers* pertencentes a um mesmo evento (representados por um grupo) e com desejo de participar (uma vez que o ingresso em grupos é voluntário). As funcionalidades de colaboração propostas foram o chat entre *streamers* e o uso de notificações em mapa.

5.1 Resultados alcançados

No fim deste trabalho definimos um conjunto de funcionalidades que podem ser estendidas em plataformas de *live streaming* para smartphones. Os principais resultados obtidos podem ser listados da seguinte maneira:

1. O uso de informações de dados de geolocalização e/ou *hashtags*, aplicados de formas distintas entre cada plataforma de *live streaming*, foram agrupados no chamado compartilhamento de contexto (que poderá incluir outros dados que descrevam um evento).

2. Foi considerado um cenário de formação de grupos baseado na intenção de cada membro em fazer parte do grupo, fazendo-o e desfazendo-o a qualquer instante. Optou-se por esta forma de agrupamento principalmente devido à dificuldade de lidar com agrupamentos automáticos, havendo grande

variação na disponibilidade e mobilidade dos participantes. Assim, sabendo-se que os membros de um grupo ingressaram por vontade própria, é possível ter maior flexibilidade com as fronteiras que definem um grupo (ex. distância).

3. Foi definido um canal de comunicação entre *streamers*, o qual é inexistente em plataformas de *live streaming* no momento em que este trabalho foi realizado. Nesse canal, foram propostas duas formas de comunicação. O chat em grupo, a primeira proposta, consistiu na troca de mensagens de texto entre todos os membros do grupo. A segunda proposta, baseada em jogos do tipo *multiplayer* com mapa, consistiu na troca de mensagens mostradas em mapa, cada uma com um sentido semântico próprio (ex. pedido de ajuda, aviso de perigo, etc.).

4. Foi desenvolvido um protótipo baseado em plataforma Android que permite aplicar os conceitos propostos neste trabalho. Destacamos que o nosso protótipo permite simular uma extensão para plataformas de streaming, uma vez que é possível utilizá-lo sobreposto (*piggyback*) a aplicativos mobile de *live streaming* (ex. Twitcasting).

   Ainda, experimentos realizados em diferentes cenários sugerem que o protótipo pode diminuir o tempo de cobertura em até 13% e o atraso de suas interações (troca de mensagens) pode atingir em torno de um segundo. Por outro lado, as funcionalidades propostas e implementadas deixam abertas possibilidades de interações (entre *streamers*) que permitem o compartilhamento de recursos (ex. energia, acesso à banda larga, etc.) ou evitem o final repentino de live stream (ex. ser detido por policiais, etc.).

5.2 Limitações

Primeiramente destacamos que a nossa proposta está sujeita à estabilidade e qualidade das tecnologias de telecomunicações. Isto quer dizer que, em condições muito ruins de acesso à banda larga, continuará sendo impraticável a interação remota com outros participantes de um *live streaming* colaborativo.

Entretanto, se as condições de acesso mínimas forem garantidas pela rede, é possível que participantes interajam e colaborem.

O fato de termos implementado apenas os componentes propostos, sem implementar ou configurar servidores de *streaming* próprios, faz com que o acesso ao conteúdo de vídeo seja limitado. Para isso, as formas de acesso ao conteúdo são aquelas disponibilizadas pela plataforma de *live streaming* escolhida, efetuadas diretamente sobre o fluxo de vídeo transmitido. Ainda com relação ao acesso ao conteúdo do vídeo, algumas plataformas, como o Twitcasting, possuem API que permite obter algumas informações sobre canais e transmissões (ex. duração, *hashtags*, número de espectadores, etc.) e, inclusive, obter um *frame* de vídeo capturado no instante em que é requisitado. Entretanto, o uso desses recursos requer maior dependência de componentes externos àqueles que fazem parte do protótipo implementado.

Finalmente, a nossa proposta não garante completamente que os membros de um grupo estarão sempre dispostos a colaborar, por exemplo, respondendo a notificações em mapa (ex. de ajuda ou alvo). Para isso, assumimos que, pelo fato *streamers* ingressarem em um grupo apenas quando desejado, então há certa disposição em colaborar. Entretanto, elementos que estimulem a colaboração ainda devem ser estudados, tendo a atenção para não tornar o processo complexo.

5.3 Trabalhos futuros

Dentre os trabalhos futuros listamos os seguintes:

- Propor formas de apresentação do conteúdo colaborativo para espectadores, uma vez que este trabalho focou-se no problema da produção do *live streaming* colaborativo. A apresentação do conteúdo colaborativo deve explorar informações fornecidas em compartilhamentos de contexto (ex. usando mapas) e explorar também formas de interação entre todos os participantes (*streamers* e espectadores). Técnicas iniciais de apresentação de múltiplos conteúdos incluem visualizações em *grid*, ferramentas de *live*

mixing, etc.

- Investigar outras funcionalidades de colaboração, por exemplo, incluindo o compartilhamento de mensagens de áudio entre *streamers*, como ocorre em [PROCYK et al. 2014] ou inclusive em jogos de tipo *multiplayer*. As principais características da comunicação a partir de voz são maior expressividade e a rapidez em compor mensagens. Por outro lado, devem-se considerar aspectos como a disputa pelo canal de comunicação (ex. duas pessoas falando ao mesmo tempo).

- Estudar formas de compartilhamento de recursos, em especial o acesso à banda larga, respondendo a notificações de ajuda.

- Investigar e avaliar as interfaces de usuário do protótipo, em busca de identificar aspectos que tornem a colaboração mais intuitiva e estimulante, levando em consideração os limites impostos pelas dimensões das telas de aparelhos smartphone.

- Por último, desejamos aproximar-nos a grupos que realizam coberturas de manifestações, tal como a Mídia NINJA, apresentando a nossa proposta e a forma como ela pode ser usada para abordar alguns dos problemas que surgem em ação. Principalmente, ao estabelecer esse tipo de contato seriamos capazes de obter o *feedback* sobre as funcionalidades propostas e inclusive identificar novas demandas que ainda não foram contempladas por nós, estudando as suas práticas e estratégias de coordenação correntes.

BIBLIOGRAFIA

ALLAN 2007: Allan, S. Citizen Journalism and the Rise of "Mass Self-Communication": Reporting the London Bombings. *Global Media Journal: Australian Edition. n.* 1, p. 1-20. 2007.

ALMEIDA e EVANGELISTA 2013: Almeida, T.; Evangelista, A. Tecnologias móveis, mídias independentes e coberturas de mobilizações sociais urbanas: as influências do "midialivrismo" na sociedade midiatizada. *Anais do II Colóquio Semiótica das Mídias*. Japaratinga, AL. v. 2, n. 1. 2013.

AUSTERBERRY 2005: Austerberry, D. The Technology of Video and Audio Streaming. 2ª ed. *Focal Press*. Burlington, MA. 2005.

BAMBUSER 2014: Bambuser AB. Bambuser: Show The World. Disponível em http://bambuser.com/. Acessado em: Acessado em: 1 de jun. 2014.

BENGTSSON 2013: Bengtsson, R. Action! Live Streaming as means of civic engagement: A case study of citizen journalism in Egypt and Syria. *Glocal Times*. n. 19. p. 1-13. 2013.

BODDHU et al. 2012: Boddhu, S.; Williams, R.; Wasser, E.; Kode, N. Increasing situational awareness using smartphones. *Proc. SPIE 8389*. 2012.

BOIX et al. 2011: Boix, E.; Carreton, A.; Scholliers, C.; Van Cutsem, T.; De Meuter, W.; D'Hondt, T. Flocks: Enabling Dynamic Group Interactions in Mobile Social Networking Applications. *Proceedings of the 2011 ACM Symposium on Applied Computing.* p. 425-432. 2011.

CAMUÑAS 2014: Camuñas, M. Miradas colectivas: veinte años devideoactivismo en España. *Videoactivismo: Acción política cámara en mano*. p. 77-98. 2014.

COSTANZA-CHOCK 2012: Costanza-Chock, S. Mic Check! Media Cultures and the Occupy Movement. *Social Movement Studies.* v. 11. p. 375-385. 2012.

DE FREITAS e DEY 2015: de Freitas, A.; Dey, A. The Group Context Framework: An Extensible Toolkit for Opportunistic Grouping and Collaboration. *CSCW '15.* p. 1602-1611. 2015.

DEY e ABOWD 1999: Dey, A.; Abowd, G. Towards a Better Understanding of Context and Context-Awareness. *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing.* p. 304-307. 1999.

ENGSTROM et al. 2012: Engström, A.; Perry, M.; Juhlin, O. Amateur vision and recreational orientation: creating live video together. *CSCW '12.* p. 651-660. 2012.

ENGSTROM et al. 2012B: Engström, A.; Zoric, G.; Juhlin, O. The Mobile Vision Mixer: A mobile network based live video broadcasting system in your mobile phone. *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia.* 2012.

HESSEL 2013: Hessel, C. No meio do redemunho. *Estadão.* 2013. Disponível em http://www.estadao.com.br/noticias/geral,no-meio-do-redemunho,1050880. Acessado em: 7 de mar. 2016.

JAIN et al. 2013: Jain, P.; Manweiler, J.; Archarya, A.; Beaty, K. FOCUS: Clustering crowdsourced Videos by Line-of-Sight. *SenSys '13.* 2013.

JORVID 2014: Jorvid, N. Speaking Symbols: A semiotic analysis of the Smart Ping system in League of Legends. *Uppsala Universitet.* 2014.

JUHLIN et al. 2012: Juhlin, O.; Engström, A.; Reponen, E. Mobile broadcasting: the whats and hows of live video as a social medium. *MobileHCI '10.* p. 35-44. 2010.

JUHLIN et al. 2014: Juhlin, O.; Zoric, G.; Engström, A.; Reponen, E. Video Interaction: A

Research Agenda. *Personal and Ubiquitous Computing*. v. 18. n. 3. p. 685-692. 2014.
LINKBUBBLE 2014: Link Bubble. Link Bubble: mobile browsing done right. Disponível em http://linkbubble.com/. Acessado em: 1 de jun. 2014.
LIVESTREAM 2014: Livestream. Livestream: Watch or Broadcast Live Events. Disponível em http://livestream.com/. Acessado em: 1 de jun. 2014.
MEDEIROS 2013: Medeiros, A. Por que apenas a Globo foi hostilizada nos protestos?. *Observatório da Imprensa.* 2013. Disponível em http://observatoriodaimprensa.com.br/imprensa-em-questao/_ed756_por_que_apenas_a_globo_foi_hostilizada_nos_protestos/. Acessado em: 1 de jun. 2014.
MONGO 2015: MongoDB, Inc. MongoDB for GIANT Ideas. Disponível em https://www.mongodb.org/. Acessado em: 2 de maio 2015.
MONTANE-JIMENEZ 2014: Montané-Jiménez, L.; Benítez-Guerrero, E.; Mezura-Godoy, C. Towards a Context-Aware Framework for Improving Collaboration of Users in Groupware Systems. *EAI Endorsed Transactions on Context-aware Systems and Applications*. v. 1. 2014.
NINJA 2015: Mídia NINJA: Narrativas Independentes, Jornalismo e Ação. Disponível em https://ninja.oximity.com/. Acessado em: 7 de abr. 2015.
OCCUPYSTREAMS 2014: OccupyStreams: The revolution will not be televised. Disponível em http://occupystreams.org/. Acessado em: 17 de set. 2014.
PERISCOPE 2015: Periscope. Disponível em https://www.periscope.tv/. Acessado em: 24 de abr. 2015.
PROCYK et al. 2014: Procyk, J.; Neustaedter, C.; Pang, C.; Tang, A.; Judge, T. Exploring video streaming in public settings: shared geocaching over distance using mobile video chat. *CHI '14.* p. 2163-2172. 2014.
SA et al. 2014: Sa, M.; Shamma, D.; Churchill, E. Live Mobile Collaboration for Video Production: design, guidelines, and requirements. *Personal and Ubiquitous Computing.* v. 18. n. 3. p. 693-707. 2014.
SAINT-ANDRE et al. 2009: Saint-Andre, P.; Smith, K.; Tronçon, R. XMPP: The Definitive Guide. *O'Reilly Media.* 2009.
SCIKIT 2014: scikit-learn: machine learning in Python. Disponível em http://scikit-learn.org/. Acessado em: 1 de jun. 2014.
TWITCASTING 2014: Twitcasting: Comunique-se transmitindo ao vivo. Disponível em http://twitcasting.tv/. Acessado em: 1 de jun. 2014.
USTREAM 2014: Ustream: The World is Live on Ustream. Disponível em http://www.ustream.tv/. Acessado em: 1 de jun. 2014.
WEBREALIDADE 2014: Web Realidade: Canais simultâneos ao vivo! Mídia Livre!. Disponível em http://www.webrealidade.org/. Acessado em: 1 de jun. 2014.
WERNECK e STURM 2013: Werneck, F.; Sturm, H. Protesto termina em confronto e deixa três feridos. *Estadão*. 2013. Disponível em http://www.estadao.com.br/noticias/geral,protesto-termina-em-confronto-e-deixa-tres-feridos,1056417. Acessado em: 8 de mar. 2016.
WHATSAPP 2014: WhatsApp Protocol: FunXMPP. Disponível em https://github.com/mgp25/Chat-API/wiki/FunXMPP-Protocol. Acessado em: 16 de dez. 2014.
XMPP 2015: XMPP Standards Foundation. Disponível em http://xmpp.org/. Acessado em: 2 de maio 2015.

ANEXOS

ANEXO I - QUESTIONÁRIO DO EXPERIMENTO FUNCIONAL

Q1) Recebeu as notificações em mapa sinalizando o local de encontro?

- Do LPRM para a reitoria
- Da reitoria para o centro de línguas
- Retorno para o LPRM
- Nenhuma

Q2) Conseguiu gravar o vídeo durante a realização do experimento?

- Sim
- Apenas na ida
- Apenas na volta
- Não

Q3) Conseguiu gravar a tela durante a realização do experimento?

- Sim
- Apenas na ida
- Apenas na volta
- Não

Q4) O aplicativo ninjabubble deixou de funcionar em algum momento?

- Não
- 1 vez

- 2 vezes
- Várias vezes

Q5) Caso o aplicativo ninjabubble deixou de funcionar, descreva o(s) episódio(s).

Q6) Recebeu mensagens de chat?

- Sim
- Não

Q7) Foi capaz de enviar mensagens no chat?

- Sim (normalmente)
- Sim (com atraso/delay perceptível)
- Não
- Não percebi

Q8) Com que objetivo você usou o chat?

- Por curiosidade
- Para confirmar os locais de encontros
- Não usei

Q9) Conseguiu perceber em mapa o deslocamento dos demais participantes? (Em relação aos alvos estabelecidos em mapa)

- Sim
- Não
- Não percebi

Q10) Como você avalia a usabilidade da interface visual?

- Muito ruim
- Ruim
- Normal
- Boa
- Muito boa

Q11) Como você avalia a precisão da sua própria localização?

- Muito ruim
- Ruim
- Normal
- Boa
- Muito boa

Q12) Como você avalia a precisão da localização dos demais participantes?

- Muito ruim
- Ruim
- Normal
- Boa
- Muito boa

Q13) Como você avalia a taxa de atualização dos ícones em mapa? (Localização própria e participantes)

- Muito ruim

- Ruim
- Normal
- Boa
- Muito boa

Q14) O que você mudaria a respeito das funcionalidades já existentes? (Forneça sugestões)

Q15) Sentiu falta de alguma funcionalidade? (Forneça sugestões)