# Crowdstreaming: transmisión de vídeo y audio por una red ad hoc de teléfonos móviles

Crowdstreaming: video and audio transmission in a MANET (mobile ad hoc network)


**TRABAJO FIN DE GRADO**

**Francisco Calero Velasco**
**Sergio Manzanaro Caraballo**
**David Salido Camacho**

**Grado en Ingenería de Software**
**Facultad de Informática**
**Universidad Complutense de Madrid**

**Director: Simon Pickin**

**Curso 2019/2020**

# Agradecimientos

En primer lugar queremos agradecer a nuestro principales apoyo del día a día, nuestra familia y amigos.

En segundo lugar a la fundación "Shuttleworth Foundation"[1], la cual financió el proyecto en el que se enmarca este trabajo fin de grado para la compra del material necesario, sin el cuales no hubiera sido posible la realización de este.

En tercer lugar a nuestro director Simon Pickin, quien nos ha ayudado en todo momento a dar vida a este proyecto y a conseguir el material necesario para llevar a cabo la implementación..

En cuarto lugar, a los pequeños desarrolladores que han subido repositorios de gran ayuda a Github, con los cuales hemos obtenido mucha información útil, tales como los usuarios "Anagramrice" con su aplicación NAN o "Karmakargopi" con su aplicación ChitChat.

En quinto lugar queremos agradecer a los creadores de TEXIS cuya plantilla de latex nos ha ahorrado mucho tiempo a la hora de realizar esta memoria.

Por último lugar, a los profesores cuyas enseñanzas han hecho posible la obtención de conocimientos necesarios para poder afrontar desafíos tales como este proyecto.

# Palabras clave

- Wifi Aware
- Android
- Streaming
- Ad hoc
- Wifi
- Witness

# Keywords


- Wifi Aware
- Android
- Streaming
- Ad hoc
- Wifi
- Witness

# Resumen

El objetivo principal de este proyecto ha sido desarrollar una aplicación para dispositivos móviles capaz de crear una red *ad hoc*, y de retransmitir flujos de vídeo y audio directamente desde la cámara y micrófono de uno a varios dispositivos de la red. La aplicación propuesta debe poder crear una red infinita por interconexión de dispositivos cercanos, todo esto sin usar la infraestructura de red de los operadores de telecomunicaciones.

Esta idea de proyecto se comenzó a implementar el año pasado en el TFG "Device to Device streaming en dispositivos móviles" pero debido a las limitaciones de la tecnología utilizada, no se pudo llegar a implementar toda la funcionalidad esperada. Por ello, hemos llevado a cabo la implementación mediante una nueva tecnología llamada "Wifi Aware" la cual nos ha permitido completar alguna de las características que no pudieron desarrollarse y mejorar las que ya estaban hechas. No obstante, nos encontramos una serie de desafíos nuevos a los que tuvimos que hacer frente los cuales explicamos a lo largo del desarrollo de la memoria.

El código final de nuestro proyecto se encuentra en un repositorio de GitHub:

https://github.com/davidsalido/CrowdStreaming

# Summary

The main objective of this project is to develop an application for mobile devices capable of creating an ad hoc network, and to broadcast video and audio streams directly from the camera and microphone to one or several devices in the network. The application has to be able to create an infinite network by interconnecting nearby devices, all without using the network infrastructure of the telecommunication operators.

The development of this idea began last year on the final project "Device to Device streaming on mobile devices" but due to the limitations of the technology used, it was not possible to implement all the expected functionality. Therefore, we have carried out the implementation using a new technology called "Wifi Aware" which has allowed us to complete some of the features that could not be developed and improve those that were already done. Nevertheless, we found a series of new challenges that we had to face which we explain throughout the development of the memory.

The source code of our project is hosted in a public repository on GitHub at the following URL:

https://github.com/davidsalido/CrowdStreaming

# Índice

# 1. Introducción

## 1.1. Motivación e interés

La mayoría de las comunicaciones de hoy en día se realizan mediante internet [2], lo que conlleva una gran dependencia con esta tecnología. Esta dependencia nos lleva a pensar en la necesidad de buscar alternativas para casos particulares en los que la comunicación es vital y no se dispone de conexión a la red.

Hay que buscar una solución a estas situaciones en las que no se dispone de una conexión a internet, buscando crear una red *ad hoc* para permitir la comunicación. Estos tipos de redes no necesitan ningún punto de acceso para comunicar los dispositivos, si no que la conexión se hace directamente entre ellos.

Una plataforma ideal sobre la que construir nuestra alternativa a internet son los *smartphones*. Estos son utilizados por gran parte de la población, son fáciles de transportar, disponen de una gran potencia y ofrecen un gran abanico de tecnologías para la comunicación.

Con nuestro proyecto, buscamos la creación de una aplicación que permita comunicar dispositivos móviles para transmitir audio y vídeo en casos donde no existe la posibilidad de conexión a internet. Por ejemplo, en situaciones como un desastre natural o un país bajo un régimen opresivo que censure la comunicación por internet.

Para lograr esta comunicación, existen tecnologías sin infraestructura de red que nos permiten crear redes *ad hoc* desde nuestros *smartphone*. Entre estas podemos destacar Bluetooth, Wifi Direct y Wifi Aware. Estas tres tecnologías permiten crear redes inalámbricas de corto alcance con las que se puede transmitir datos sin necesidad de intermediarios. La mayor diferencia entre estas es su ancho de banda, velocidad de transmisión y alcance, siendo de las tres, Wifi Aware la más avanzada en términos técnicos seguida por Wifi Direct.

El año pasado se propuso un TFG [3] en esta misma facultad con el mismo objetivo al nuestro, donde se usaba Wifi Direct para establecer la comunicación. Esta tecnología provocó que este proyecto tuviera una serie de problemas por los que no pudo llegar a implementar toda la funcionalidad descrita. Con nuestro proyecto intentaremos solventar los problemas que tuvieron, además de implementar las características que faltaban, haciendo uso de Wifi Aware como alternativa.

El principal problema de utilizar Wifi Aware, razón por la cual los alumnos del año pasado no la eligieron, es el poco conocimiento sobre esta y la reducida cantidad de dispositivos que lo llevan implementado, viéndose reducido, de momento, a

dos dispositivos, el Pixel 3 y el Samsung Galaxy Note 10, cuya implementación de Wifi Aware está certificada por la Wifi Alliance. Esto puede dar a lugar a un desarrollo lento, más centrado en la investigación, ya que apenas existen aplicaciones que nos permiten comprobar su comportamiento.

Por otro lado, los alumnos del año pasado propusieron una serie de características que se podrían implementar en un futuro, de las cuales destacamos: controlar que el contenido no ha sido alterado, filtros para detección de contenido inapropiado como podría ser pornografía infantil o establecer medidas para garantizar el anonimato de los usuarios, las cuales también serán estudiadas.

## 1.2. Requisitos breves e informales

En la aplicación existirá un emisor principal que transmitirá a uno a varios receptores que actuarán de nodo intermedio, a su vez retransmiten a otros nodos intermedios hasta llegar a uno que dispone de conexión a internet (nodo gateway). Este último podrá subir el *stream* a internet desde la propia aplicación.

Para cumplir esta serie de requisitos, hemos propuesto una serie de objetivos los cuales iremos cumpliendo:

1. Implementar un nodo simple sobre Wifi Aware:
   - Modo "publisher": se permitirá a un usuario emitir vídeo y audio producido por la cámara y el micrófono de su dispositivo.
   - Modo "subscriber": se permitirá que un usuario reciba vídeo y audio.
2. Implementar un "nodo de tránsito simple" sobre Wifi Aware:
   - Modo "dual": permite a un usuario recibir vídeo y audio y simultáneamente emitirlo hacia un tercer dispositivo.
3. Implementar un "nodo de tránsito completo" sobre Wifi Aware:
   - Modo "dual": permite a un usuario recibir vídeo y audio y simultáneamente emitirlo a todos los dispositivos vecinos sin límite de dispositivos.
4. Implementar un "nodo de tránsito gateway":
   - Permite a un usuario recibir vídeo y audio por Wifi Aware y simultáneamente emitirlo a través de un punto de acceso Wifi o una estación base de la red de telefonía móvil.
5. Ajustar el funcionamiento correcto de la red para la distribución de vídeo.

6. Estudiar los problemas de seguridad y privacidad que pueden surgir y proponer/implementar posibles soluciones a estos problemas.

Dada los diferentes modos de uso que se pueden dar para la aplicación, hemos decidido distinguir dos escenarios relevantes según el entorno donde se vaya a usar. Estos son:

- **Humanitarian**: este escenario se centra en casos donde no hay conexión a internet debido a acciones ajenas al ser humano como puede ser catástrofes naturales o apagones de la red eléctrica de forma accidental. Permitiendo que la información se transmita desde los focos de dicho escenario hasta puntos donde haya cobertura, donde se puede pedir ayudar o compartir los vídeos con los medios de comunicación.
- **Witness**: este escenario estaría enfocado a situaciones en las que un régimen opresivo limita o censura las comunicaciones. Ejemplos de estos escenarios podrían ser ejecuciones o actos contra los derechos humanos, los cuales serían útiles grabarlos para su denuncia internacional.

## 1.3. Restricciones de los escenarios

Como hemos mencionado anteriormente la aplicación considerará dos escenarios de uso distintos para los cuales debemos tener una serie de restricciones que debemos considerar:

- **Humanitarian**: debido a la naturaleza de uso de este escenario, el cual será usado con fines informativos y solidarios, supondremos que los usuarios de la red no harán un uso malintencionado de esta. Por tanto, no tendremos en cuenta las restricciones de seguridad o ataques en este tipo de uso.

  Por otro lado, para mantener el anonimato de los usuarios y de las posibles víctimas de estos incidentes, tendremos en cuenta poder difuminar las caras mediante el uso de alguna biblioteca.

- **Witness**: el hecho de que este escenario este enfocado a denunciar actos opresivos o violaciones de los derechos humanos, puede dar a lugar a que se intente perjudicar a los autores de dicho contenido atacando a la red. Hemos considerado una serie de posibles ataques y problemas de seguridad los cuales deberíamos controlar para garantizar un buen uso de esta y que sus usuarios no sean perjudicados. Estos posibles casos son los siguientes:

  - **Inyección de otros flujos**: uno de los posibles escenarios que hemos considerado es que algún usuario de la red inyecte vídeos inapropiados por la red para hacérselo llegar a los espectadores como por ejemplo pornografía infantil.

- **DeepFake** [4]: es una técnica de inteligencia artificial que permite editar vídeos falsos de personas que aparentemente son reales, utilizando para ello algoritmos de aprendizaje. Debido a que es una técnica muy moderna no se disponen de métodos efectivos para detectar esta mala práctica. El hecho de no controlar este tipo de técnicas conlleva que los vídeos no puedan ser tomados como veraces del todo para ser usados en un juicio.
- **Modificación y reemisión de un vídeo como si fuera el original**: en algunos casos, si la retransmisión muestra acciones contra los derechos humanos o comprometedoras, puede ser beneficiario para los autores de estas acciones modificar este vídeo del tal manera que parezca original y que no ocurre nada fuera de lo normal. Al igual que ocurre con las modificaciones *DeepFake*, si no verificamos que ha ocurrido esto en el *stream*, puede conllevar no ser tomados como prueba ante un tribunal.
- **Ataques de denegación de servicio**: es un tipo ataque de a una red que causa que un servicio o recurso sea inaccesible a los usuarios. En nuestra aplicación podría causar que usuarios que intentan censurar la comunicación saturen la red, impidiéndose así el uso legítimo de esta e imposibilitar la compartición de información.
- **Anonimato de los usuarios**: debido las posibles represiones que puede haber sobre un usuario de nuestra aplicación, el cual quiere denunciar abuso de un régimen opresivo, es vital garantizar la seguridad y anonimato de los usuarios de nuestra aplicación. Hoy en día casi todo contenido multimedia lleva consigo metadatos los cuales pueden permitir el rastreo e identificación de los emisores. En nuestra aplicación se permitirá que el usuario elija si quiere mantener o no los metadatos de su contenido relacionados con la identificación y el rastreo del autor. Para una mayor protección, a la hora de usar la aplicación en modo Witness, la emisión de estos metadatos estará desactivada por defecto.
- **Protección del contenido generado por la aplicación**: dado que los usuarios podrán almacenar los *streams* emitios/visualizados es fundamental garantizar que en caso de emergencia (captura del dispositivo por parte del régimen opresivo), dispongan de un método que les permita deshacerse de cualquier rastro incriminatorio, respaldando la seguridad del usuario.

Una manera de solventar la modificación y remisión de un vídeo como si fuera el original, puede ser la autenticación de vídeo mediante firmas digitales permitiendo al receptor comprobar si un flujo de vídeo ha sido alterado entre su emisión y su recepción. Esta técnica no solventaría la inyección de otros flujos, ya que si el atacante tuviera el mismo software de autenticación que el emisor original, no se podría detectar si este flujo ha sido alterado [5].

Además, este tipo de autenticación todavía es un tema poco maduro, ya que habría que cifrar los fotogramas del vídeo y no sería posible cifrar todos, ya que añadiría una gran latencia. Se debería buscar una alternativa a esto, como podría ser cifrar un fotograma por cada N fotogramas o solo una parte de cada fotograma. Por lo tanto, todavía no se sabe muy bien como sería la técnica óptima a la hora de cifrar vídeos.

Para solventar el problema de inyección de flujos, además de la emisión de *deepfakes*, una posible solución es encriptar los vídeos mediante algún tipo de clave acordada de antemano entre los participantes de la red. Una solución básica para esto, sería el uso de algún protocolo como GDOI [6] el cual permite la gestión de claves de cifrado comunes a un grupo de personas, aunque esto, puede no servir si algún integrante filtra dicha clave.

Finalmente, para la protección del contenido generado por la aplicación, hemos pensado en la implementación de un botón del pánico, que mediante un clic podamos eliminar todo este contenido del dispositivo y la propia aplicación.

## 1.4. Línea cronológica

Debido a la poca cantidad de dispositivos que disponen de Wifi Aware, la tecnología la cual vamos a utilizar para la realización de nuestro proyecto, es necesaria la financiación de dispositivos sobre los que realizar el desarrollo. Para ello, nuestro director ha contactado con varios fundaciones y empresas privadas que estuviesen dispuestas a concedernos una beca. Dado estas casuísticas, hemos decidido planificar haciendo una división donde en una primera parte sin la financiación, nos dedicamos plenamente a la investigación y los requisitos y en la segunda parte con ya con los dispositivos, nos centramos en el desarrollo.

En la siguiente figura se muestra una planificación inicial divida por tres grandes tareas, las cuales se pueden dividir en subtareas

| Tarea | Inicio | Fin |
|---|---|---|
| | | |
| Redacción memoria | 12/09/2019 | 01/06/2020 |
| Investigación y especifiación de requisitos | 12/09/2019 | 31/01/2020 |
| Investigación Wifi Aware | 12/09/2019 | 01/10/2019 |
| Aplicaciones similares | 01/10/2019 | 15/10/2019 |
| Tecnologias parecidas | 15/10/2019 | 01/11/2019 |
| Especificación de requisitos | 01/11/2019 | 10/12/2019 |
| Diseño mockups | 10/12/2019 | 20/12/2019 |
| Tecnologias de conexión | 08/01/2020 | 20/01/2020 |
| Bibliotecas a usar | 20/01/2020 | 31/01/2020 |
| Desarrollo | 01/02/2020 | 30/04/2020 |
| Desarrollo de prototipo | 01/02/2020 | 20/02/2020 |
| Desarrollo de aplicación mediante metodología agile | 20/02/2020 | 14/04/2020 |
| Pruebas con usuarios ajenos al desarrollo | 14/04/2020 | 30/04/2020 |

Figura 1.1: Planificación

# 2. Introduction

## 2.1. Motivation and interest

Most of today's communications are done through the internet [2], which entails a heavy dependency on this technology. This dependence leads us to think about the need to look for alternatives for particular cases in which communication is vital and no network connection is available.

A solution must be found to these situations where an Internet connection is not available, by seeking to create an ad hoc network to enable communication. These types of networks do not need any access point to communicate the devices, but the connection is made directly between them.

An ideal platform to build our alternative to the internet is the smartphone. These are used by a large part of the population, are easy to transport, have great power and offer a wide range of technologies for communication.

With our project, we are looking to create an application that allows the communication of mobile devices to broadcast audio and video in cases where there is no internet connection. For example, in situations such as a natural disaster or a country under an oppressive government that censors internet communication.

To achieve this communication, there are technologies without network infrastructure that allow us to create ad hoc networks from our smartphones. Among these we can highlight Bluetooth, Wifi Direct and Wifi Aware. These three technologies allow us to create short range wireless networks which data can be transmitted without the need of any middlemen. The biggest difference between these is their bandwidth, transmission speed and range, with Wifi Aware being the most technically advanced of the three, followed by Wifi Direct.

Last year, a final proyect [3] was proposed to be held in this same faculty with the same objective as ours, where Wifi Direct was used to establish communication. This technology caused this project to have a series of problems for which it could not implement all the functionality described. With our project we will try to solve the problems they had, besides implementing the missing features, using Wifi Aware as an alternative.

The main problem of using Wifi Aware, which is why last year's students did not choose it, is the little knowledge about it and the reduced number of devices that have it implemented, being reduced, at the moment, to two devices, the Pixel 3 and the Samsung Galaxy Note 10, whose implementation of Wifi Aware is certified by

the Wifi Alliance. This may lead to a slow development, more focused on research, since there are hardly any applications that allow us to check it behavior.

On the other hand, last year's students proposed a series of features that could be implemented in the future, of which we highlight: control that the content has not been altered, filters to detect inappropriate content such as child pornography or establish measures to ensure the anonymity of users, which will also be studied.

## 2.2. Short and informal requirements

In the application there will be a main transmitter that will broadcast to one or more receivers that will act as an intermediate node, which in turn will broadcast to other intermediate nodes until reaching one that has an internet connection (gateway node). This last one will be able to upload the retransmission to the Internet from the application itself.

To meet this set of requirements, we have proposed a series of objectives which we will fulfil:

1. Implement a simple node using Wifi Aware:
   - Publisher mode: a user will be allowed to broadcast video and audio captured by the camera and microphone of his device;
   - Subscribe mode: a user will be allowed to receive video and audio
2. Implement a single transit node using Wifi Aware:
   - Dual mode: allows a user to receive video and audio and simultaneously broadcast it to a third device
3. Implement a full transit node using Wifi Aware
   - Dual mode: allows a user to receive video and audio and simultaneously broadcast it to all adjacent devices with no device limit.
4. Implement a gateway transit node using Wifi Aware:
   - It allows a user to receive video and audio via Wifi Aware and simultaneously broadcast it through a Wifi access point or a mobile phone network base station.
5. Set up the proper operation of the network for video distribution.
6. Study the security and privacy issues that may arise and propose/implement possible solutions to these problems

Given the different ways to use the application, we have decided to distinguish two relevant scenarios according to the environment where it will be used. These are:

- **Humanitarian**: this scenario focuses on cases where there is no internet connection due to non-human actions such as natural disasters or accidental power outages. It allows information to be transmitted from the origin to points where there is internet coverage, where people can ask for help or share the videos with the media.
- **Witness**: this scenario would focus on situations where an oppressive government limits or censors communications. Examples of these scenarios could be executions or acts against human rights, which would be useful to record for international denunciation.

## 2.3. Restrictions of each scenario

As mentioned above the application will consider two different usage scenarios for which we must have a number of restrictions to consider:

- **Humanitarian**: due to the nature of the use of this scenario, which will be used for information and solidarity purposes, we will assume that the users of the network will not make a malicious use of it. Therefore, we will not take into account security restrictions or attacks on this type of use.

  On the other hand, to maintain the anonymity of the users and possible victims of these incidents, we will take into account being able to blur the faces by using some library.

- **Witness**: The fact that this scenario is focused on denouncing oppressive acts or human rights violations, can lead to attempts to harm the authors of such content by attacking the network. We have considered a number of possible attacks and security issues which we should monitor to ensure that it is used properly and that users are not harmed. These possible cases are as follows:

  - **Injection of other streams**: one of the possible scenarios we have considered is that some network user injects inappropriate videos through the network to get them to viewers such as child pornography.
  - **DeepFake** [4]: is an artificial intelligence technique that allows you to edit fake videos of apparently real people, using learning algorithms. Because it is a very modern technique there are no effective methods to detect this misconduct. The fact that this type of technique is not controlled means that the videos cannot be taken as completely true to be used in a trial.

- **Modification and reissue of a video as if it were the original**: in some cases, if the broadcast shows actions against human rights or compromising, it may be in the interest of the authors of these actions to modify this video in such a way that it looks original and that nothing out of the ordinary happens. As with DeepFake modifications, if we do not verify that this has occurred in the broadcast, it may result in not being taken as evidence in a trial.
- **Denial of service attacks**: is a type of network attack that causes a service or resource to be inaccessible to users. In our application it could cause users who try to censor communication to saturate the network, thus preventing legitimate use of the network and making information sharing impossible.
- **Anonymity of users**: due to the possible reprehension that can be taken on a user of our application, who wants to report abuse of an oppressive regime, it is vital to guarantee the security and anonymity of the users of our application. Nowadays almost all multimedia content carries metadata which can allow the tracking and identification of the issuers. In our application the user will be allowed to choose whether or not to keep the metadata of their content related to the identification and tracking of the author. For a better protection, when using the application in Witness mode, the emission of these metadata will be disabled by default.
- **Protection of application-generated content**: since users will be able to store the video of broadcasts it is essential to ensure that in case of emergency (capture of the device by the oppressive regime), they have a method to get rid of any incriminating traces, supporting the security of the user.

One way of solving the modification and remission of a video as if it were the original, can be the video authentication through digital signatures allowing the receiver to check if a video stream has been altered between its emission and its reception. This technique would not solve the injection of other streams, since if the attacker had the same authentication software as the original sender, it would not be possible to detect whether this stream has been altered

In addition, this type of authentication is still an immature issue, as the frames of the video would have to be encrypted and it would not be possible to encrypt all of them, as it would add a large amount of latency. An alternative to this should be sought, such as encrypting one frame per N frames or only a part of each frame. Therefore, it is not yet clear what the optimal technique would be for encrypting videos.

To solve this problem, in addition to the broadcast of deepfakes, a possible

solution is to encrypt the videos by means of some kind of key agreed beforehand among the network participants. A basic solution for this would be the use of some protocol such as GDOI [6], which allows the management of common encryption keys for a group of people, although this may not be useful if any member filters the key.

Finally, for the protection of the content generated by the application, we have thought about the implementation of a panic button, that with a click we can remove all this content from the device and the application itself.

## 2.4. Timeline

Due to the small amount of devices that have Wifi Aware, the technology which we are going to use for the realization of this project, it is necessary to finance devices on which to carry out the development. For this purpose, our director has contacted several foundations and private companies that would be willing to grant us a scholarship. Given these cases, we have decided to plan by making a division where in the first part without the financing, we are fully dedicated to research and requirements and in the second part with already the devices, we focus on development.

The following figure shows an initial planning divided by three major tasks, which can be divided into sub-tasks.

| Tarea | Inicio | Fin |
|---|---|---|
| | | |
| Redacción memoria | 12/09/2019 | 01/06/2020 |
| Investigación y especifiación de requisitos | 12/09/2019 | 31/01/2020 |
| Investigación Wifi Aware | 12/09/2019 | 01/10/2019 |
| Aplicaciones similares | 01/10/2019 | 15/10/2019 |
| Tecnologias parecidas | 15/10/2019 | 01/11/2019 |
| Especificación de requisitos | 01/11/2019 | 10/12/2019 |
| Diseño mockups | 10/12/2019 | 20/12/2019 |
| Tecnologias de conexión | 08/01/2020 | 20/01/2020 |
| Bibliotecas a usar | 20/01/2020 | 31/01/2020 |
| Desarrollo | 01/02/2020 | 30/04/2020 |
| Desarrollo de prototipo | 01/02/2020 | 20/02/2020 |
| Desarrollo de aplicación mediante metodología agile | 20/02/2020 | 14/04/2020 |
| Pruebas con usuarios ajenos al desarrollo | 14/04/2020 | 30/04/2020 |

Figura 2.1: Planification

# 3. Estado del arte y Antecedentes

## 3.1. Tecnologías y protocolos de la comunicación

### 3.1.1. Bluetooth

Bluetooth [7] es una tecnología de comunicación inalámbrica de corto alcance que permite que dispositivos como ordenadores, teléfonos móviles o periféricos transmitir voz o datos. Su propósito es reemplazar los cables que normalmente conectan dispositivos mientras se mantiene una comunicación segura.

Usa la frecuencia de 2.4 GHz que es la misma que usan los teléfonos fijos sin cable y el router Wifi. Crea un red con radio de 10 metros llamada PAN (Personal Area Network) que puede conectar entre 2 o 8 dispositivos. Requiere menos batería para su uso y tiene un menor coste de implementación en comparación al Wifi. Es menos propenso que otras tecnologías a sufrir o provocar interferencias en la franja de 2.4 GHz.

La última versión salió en 2019 con un ancho de banda de 50 Mbit/s y una transmisión inferior a 1Mb. Aun así, ancho de banda es escaso y su alcance limitado, lo que la hacen una tecnología no adecuada para la emisión de audio y vídeo.

### 3.1.2. Wifi

Wifi [8] es una tecnología de redes que permite que dispositivos electrónicos, tales como teléfonos móviles y ordenadores, interactúen entre sí de forma inalámbrica. Wifi se basa en los protocolos IEEE 802.11, implicada en la capa física y en la de enlace de datos del modelo OSI.

Las frecuencias que se utilizan para crear esta red de conexión inalámbrica son distintas, pudiendo ser de 2,4GHz que se usa en el estándar 802.11n y de 5GHz en el 802.11ac. En la actualidad la frecuencia con mejores prestaciones es la de 5GHz ya que permite la transferencia de datos con menos interferencias a la par que mas rápida.

La seguridad del Wifi depende del cifrado que se aplique en la comunicación entre dispositivos. Puede ser de tipo WEP, el cual está obsoleto y es inseguro. Ahora mismo el que más se usa por su seguridad y mejora en intercambio de información es el cifrado WPA2, que es una evolución de WPA.

El uso más común de los protocolos Wifi es con un punto de acceso, es decir, un dispositivo hardware de red que permite a dispositivos móviles comunicarse mediante

Wifi con una red física de cables (o de fibra óptica).

El punto de acceso suele estar integrado con un router, para comunicar con el resto de internet, y muchas veces también con un switch, para canalizar las comunicaciones entre los dispositivos Wifi que están conectados al mismo punto de acceso, formando de este modo una red de área local inalámbrica (*Wireless LAN*).

En el presente trabajo, nos interesa la comunicación directa entre dispositivos por los protocolos Wifi, es decir, la comunicación que no pasa por una *Wireless LAN* como la que hoy en día tenemos casi todos en casa. En los escenarios de uso previstos para la aplicación que queremos desarrollar, no hay *Wireless LAN* disponible, por lo que no podemos usar Wifi como lo hacen los sistemas que conocemos, como Chromecast.

Nuestra idea en este trabajo es constuir una red *ad hoc* de teléfonos móviles comunicandose directamente entre si mediante Wifi.

### 3.1.3. Wifi Direct

Wifi Direct [9] es un tipo de conexión que permite la comunicación D2D (Device to device), enlazando dispositivos sin necesidad de un intermediario. Un dispositivo gestiona las conexiones y los otros se conectan a él usando WPS y WPA2 como protocolos de seguridad.

En comparación con Bluetooth, Wifi Direct tiene unas mejores especificaciones técnicas llegando a los 250 MBit/s y un alcance cuatro veces mayor.

El primer dispositivo en incorporar Wifi Direct fue fabricado en noviembre de 2010 sin embargo, en 2011 Google implementó Wifi Direct en Android 4.0 de forma nativa. Debido a esta decisión de Google, casi cualquier dispositivo Android en la actualidad dispone de esta tecnología.

Por lo tanto, parece que Wifi Direct sería una buena opción para la emisión de audio y vídeo ya que tiene ancho de banda decente, muchos dispositivos lo incorporan y no depende de ninguna infraestructura de red, pero como veremos en la sección 3.2.1 no será la tecnología elegida, debido a restricciones como la necesidad de un propietario del grupo o Group Owner el cual hace de intermediario en todas las comunicaciones entre los dispositivos que pertenecen a la red.

### 3.1.4. Wifi Aware

Wifi Aware [10] es una tecnología que extiende las capacidades de Wifi para el descubrimiento, conexión e intercambio de datos sin la necesidad de una infraestruc-

tura de red tradicional, conexión internet o señal GPS.

Esta tecnología nos permite establecer conexiones punto a punto basadas en la localización inmediata y preferencia de los usuarios en tiempo real con un consumo de batería mínimo. Como hemos comentado en el punto anterior, Wifi Direct usa un concepto de grupo en el que uno de los dispositivos, llamado propietario del grupo o Group Owner, tiene un papel especial,ya que los otros dispositivos se juntan al grupo conectándose a el (topología estrella) y la comunicación entre ellos siempre pasa por este, sin embargo, Wifi Aware siempre busca dispositivos cercanos y el flujo de datos entre dispositivos no tiene que pasar por un Group Owner.

Aunque para conectar dos o más dispositivos con Wifi Aware se necesite que estén dentro de un rango específico, el consumo mínimo de batería, la transferencia de datos bidireccional y su aumento de velocidad de retransmisión frente a Wifi Direct, hacen de Wifi Aware la tecnología perfecta para el proyecto.


Figura 3.1: Descubrimiento de dispositivos cercanos mediante Wifi Aware [55]

### 3.1.5. Apple Wireless Direct Link

Esta tecnología permite el descubrimiento y conexión entre dispositivos, tanto inalámbrico como por cable. Respecto a su implementación en entornos inalambri-

cos, Apple implementó este mecanismo de redes *ad hoc* tanto sobre Bluetooth como sobre Wifi.

En iOS11, quitaron la implementación sobre Bluetooth quedando solo la de Wi-fi. El mayor problema de esta tecnología es que no hay documentación publica sobre cómo funciona esta, por lo que no podemos saber si sería compatible con las necesidades de este proyecto.

Algunos investigadores han intentado analizar su comportamiento [11]. De esta investigación, sale que hay una especie de Group Owner, por lo que se da a entender que es más parecido a Wifi Direct que a Wifi Aware.

### 3.1.6. NFC

NFC [12] significa "Near Field Communication", fue aprobada en 2003 y ya en 2008 había periféricos que usaban NFC combinado con bluetooth. Es una tecnología de comunicación de corto alcance usada para la comunicación instantánea entre dispositivos sin la necesidad de emparejamiento previo.

Trabaja en bandas de frecuencia altas (13.56 MHz) a través de la inducción de un campo magnético. Su tasa de transferencia puede llegar a alcanzar los 424 Kbit/s por lo que el objetivo de esta tecnología no es una transferencia grande de datos entre dispositivos sino la comunicación instantánea.

NFC, funciona de dos modos distintos, en el modo "activo", ambos dispositivos con NFC crean un campo electromagnético para poder así realizar el intercambio de datos, mientras que el "pasivo", es un solo dispositivo con NFC el encargado de crear el campo para que el otro dispositivo pueda aprovecharse del NFC y poder realizar la transferencia.

Aunque dispone de una gran velocidad de emparejamiento entre dispositivos, esta tecnología es inviable para el proyecto, ya que su velocidad de transmisión es muy reducida para realizar *streams* y el rango máximo para conectar dos dispositivos debe ser inferior a 20 cm.

### 3.1.7. TCP

TCP [13] es un protocolo de la capa de transporte encargado del control de transmisión de datos. Este fue creado entre 1973 y 1974, utilizado para crear conexiones entre dispositivos y enviar flujo de datos entre estos, garantizando que los datos se entregan correctamente y en el mismo orden que fueron enviados.

TCP es utilizado por muchas aplicaciones en internet como por ejemplo los navegadores y protocolos de aplicación como HTTP O FTP. TCP, se encuentra en la capa intermedia entre el protocolo de red IP y la aplicación, permitiendo así que la transmisión sea segura y sin errores.

En este protocolo, se usan los números de puertos para diferenciar aplicaciones emisoras y receptoras. A su vez, se distinguen 3 tipos de puertos, los puertos bien conocidos, que van del 0 al 1023 y son usados por el sistema, los puertos registrados que van del 1024 al 49151 y los usan las aplicaciones de forma temporal para conectarse con los servidores, y por último, los puertos dinámicos o privados que van del 49152 al 65535 y son usados por aplicaciones aunque son menos comunes que los anteriores.

### 3.1.8. UDP

UDP [14] es un protocolo perteneciente a la capa de transporte, el cual se encarga del control de la transmisión de datos a la hora de enviar flujo de información entre dos dispositivos, sin que se haya establecido previamente una conexión. Este protocolo no garantiza que los datos se entregan correctamente ni en el mismo orden que fueron enviados por lo que lo hace un protocolo no fiable. Como ventaja, este protocolo tiene menos latencia, ya que no hay establecimiento de conexión y no hay mecanismos de seguimiento.

Es principalmente utilizado en aplicaciones donde el intercambio de información no es crítico, ya que la pérdida de algún datagrama es insignificante y se requiere mayor velocidad. Debido a estos motivos su principal uso es la transmisión de audio y vídeo en tiempo real.

### 3.1.9. Streaming

Streaming [15] es una tecnología usada para transmitir contenido digital como audio o vídeo a dispositivos mediante una red por un flujo continuo de datos que permite a los receptores disponer del contenido inmediatamente.

Este método de consumo de contenido permite acceder a este mientras se descarga, lo que lo hace una opción más rápida que otras tradicionales en los que el contenido se tiene que descargar completamente antes de poder usarlo.

Por estos motivos es una tecnología ideal para poder hacer retransmisiones en directo, ya que se puede enviar el contenido mientras el emisor graba sin la necesidad de haber terminado el vídeo.

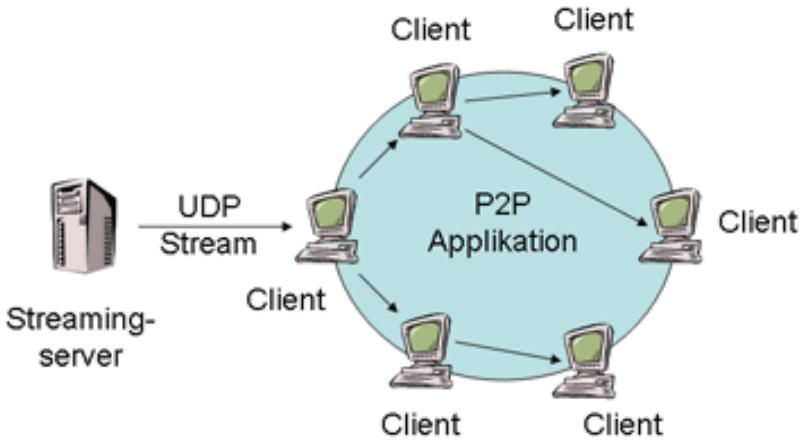



Figura 3.2: Ejemplo de P2P Streaming [56]

### 3.1.10. RTP y RTCP

Estas dos tecnologías fueron creadas para funcionar conjuntamente y conseguir mantener la calidad en la transmisión de datos y proporcionar información sobre los participantes al iniciar la sesión.

RTP(Real-time Transport Protocol) [16], es un estándar creado con el objetivo de transmitir voz y vídeo a través de Internet en paquetes de forma periódica a todos los participantes de la sesión. Conjuntamente, se define un protocolo adicional para el envío de datos de control y datos de mediciones realizadas durante la transmisión, este es RTCP.

Los paquetes RTCP [17] se envían periódicamente (normalmente cada 5 segundos) dentro de la secuencia de paquetes RTP y contienen datos que ayudan a verificar las condiciones de transmisión en el extremo remoto. Cada uno de ellos utiliza un puerto distinto y estos dos puertos suelen ser consecutivos, el RTP usa el puerto par y el RTCP el impar.

### 3.1.11. RTSP

RTSP [18] es un protocolo de control que se usa para establecer, controlar y cerrar sesiones de *stream*. Este se tiene que usar conjuntamente con un protocolo de transmisión de flujo (como RTP).

Funciona de tal manera que, cuando un usuario intenta transmitir vídeo desde un origen remoto, el cliente envía una solicitud al servidor para determinar las opciones disponibles como por ejemplo pausar, reproducir y grabar. Una vez llega la solicitud, el servidor responde con la lista de solicitudes que se aceptan mediante RTSP..

A continuación el servidor manda información sobre el mecanismo de transporte y sobre el multimedia al cliente. Una vez el cliente sabe esta información, este empieza el proceso de retransimisión diciéndole al servidor que mande la información a través del mecanismo de transporte indicado.

Este protocolo marcó el inicio de la reproducción de audio y vídeo directamente desde internet, pudiendo dar una alternativa a tener que descargar los archivos.

## 3.2. Otros enfoques y aplicaciones parecidas

Dado que Wifi Aware es una tecnología muy innovadora, no existen apenas aplicaciones que la incorporen. Encontramos una aplicación con Wifi Aware la cual permitía tener una conversación entre varios dispositivos a través de un chat, pero tenía muy poca funcionalidad y era poco intuitiva.

Por otro lado, encontramos también otras aplicaciones con tecnologías anteriores como Wifi Direct que tenían una finalidad parecida a la nuestra en cuanto a transmisión de archivos, pero eran difícil de manejar, entender y tenían muy limitado el alcance de conexión entre dispositivos.

Por lo general hay muy pocos ejemplos en los que fijarnos y que nos puedan servir de guía para nuestra aplicación ya que Wifi Aware es una tecnología muy innovadora y en la que muy poca gente y compañías se han centrado en desarrollar.

### 3.2.1. D2D Streaming

Aplicación desarrollada por los alumnos del TFG "Device to device streaming en dispositivos móviles" [3] capaz de conectar los dispositivos entre ellos sin utilizar la infraestructura de red de los operadores de telecomunicaciones, con el objetivo de compartir vídeo y audio. Esta aplicación está implementada sobre Wifi Direct, tecnología antecesora a Wifi Aware.

Durante el desarrollo de esta aplicación se encontraron varios obstáculos. El principal fue la limitación de Wifi Direct que impedía a un dispositivo ser miembro de más de un grupo a la vez. Esta funcionalidad es una parte optativa de la especificación del Wifi Alliance de Wifi Direct (es decir, optativa para los fabricantes que quieren conseguir la certificación de sus dispositivos) y Google decidió no implementar esta parte optativa en el S.O. Android.

Para resolver este problema los alumnos del año pasado intentaron implementar una solución en la que el nodo intermedio se conectaba al Group Owner A para reci-

bir un flujo de vídeo y audio, se desconectaba de este y se conectaba al Group Owner B para retransmitírselo. Sin embargo este cambio continuo entre Group Owners suponía una latencia inadmisible lo que impidió que llevaran a cabo completamente el TFG.

Según la documentación de Wifi Aware, los problemas descritos anteriormente deberían poder solventarse con esta tecnología. La razón por la que ellos no eligieron Wifi Aware, fue la falta de desarrollo y de documentación sobre esta y la falta de dispositivos que son compatibles.

### 3.2.2. ShareIt

Esta aplicación [19], disponible para Windows, macOS, Android y iOS, nos permite transferir archivos entre dispositivos a máxima velocidad conectándose Directamente entre ellos mediante Wifi Direct (sin necesidad de Internet o de pasar por un router) a velocidades superiores a los 20 MB/s.

### 3.2.3. ChitChat

Aplicación de chat [20] entre dos dispositivos que se comunican por Wifi Aware. Solamente ha sido publicada en un repositorio de GitHub, no es una aplicación con fines comerciales.

Es de lo poco que se puede encontrar con Wifi Aware al ser una tecnología tan nueva y con pocos dispositivos compatibles. No hay ninguna aplicación publicada en medios donde hay que pagar por publicar, como pueden ser el Play Store.

### 3.2.4. Serval

Aplicación de comunicación de voz [21] *ad hoc* que permite crear una MANET (Mobile *ad hoc* Network, es decir, red móvil a medida) entre dispositivos para realizar llamadas e intercambiar mensajes.

Esta aplicación usa Wifi *ad hoc* mode (Wifi IBSS), ahora considerado un poco anticuado ya que tiene problemas de seguridad. Otro inconveniente es que hay que implementar cambios de bajo nivel en Android para que pueda funcionar una aplicación en "*ad hoc* mode", con los problemas que supone estos cambios de bajo nivel para la portabilidad y la actualización del S.O.

Nos decidimos a probar esta aplicación para ver cómo funcionaban las alternativas a nuestro proyecto. Lo primero que llama la atención es su interfaz, la cual no era nada intuitiva, no se veía de forma clara que podía hacer la aplicación. Nada

más iniciar la aplicación te pide el número de teléfono, lo que no le vimos mucho sentido ya que funciona sobre una red *ad hoc*. Una vez puesto el teléfono, hay que conectarse por Bluetooth desde fuera de la aplicación. Tras estas configuraciones conseguimos hablar correctamente por el chat, sin embargo, las llamadas no se escuchaban correctamente.

### 3.2.5. MANET voice chat

Aplicación creada para lo comunicación [22] de voz estilo walkie-talkie creando una MANET para funcionar. Usa la misma tecnología que Serval (wifi *ad hoc* mode), comentada anteriormente, con sus mismas limitaciones.

Al igual que Serval, este programa también fue probado sin mucho éxito, ya que tampoco tenía una interfaz clara. Además de no funcionar las llamadas tampoco funcionaba el chat por lo que no pudimos coger nada como ejemplo para el proyecto.

Estas dos últimas aplicaciones nos hicieron ver que las alternativas para la comunicación por red *ad hoc* a día de hoy son de muy baja calidad, ya que hay muy poco desarrollo sobre esto.

### 3.2.6. GoTenna

GoTenna [23] es una dispositivo de comunicación por radio que permite enviar texto y compartir datos de localización con otros usuarios desde un *smart device*, lo que la convierte en una buena alternativa a los teléfonos vía satélite.

Se desarrollo con el objetivo de comunicarse en lugares donde no hay cobertura y podemos requerir de señales para comunicarnos, como puede ser a la hora de acampar o realizar expediciones.

Aunque es buena alternativa a las redes convencionales de telecomunicaciones, solo sirve para enviar mensajes de texto o compartir información de la localización, sin permitir enviar contenido multimedia ni contactar a los servicios de emergencia.

### 3.2.7. ObscuraCam

Aplicación [24] permite realizar fotos y vídeos a la vez que proteger la privacidad de las personas que aparecen en estos. Con esta aplicación puedes difuminar las caras de las personas y borrar toda la información sobre la propiedad del vídeo.

Al ser una aplicación fácil de descargar y probar, nos dispusimos a probarla. Lo primero que notamos era su interfaz, era poco intuitiva y había pocas aclaraciones

de cómo utilizarla. Al conseguir ponerla en marcha, la difuminación de las caras no fue del todo satisfactoria, ya que no siempre detectaba las caras y no las difuminaba como se esperaba. Esta aplicación fue probada por los dispositivos Xiaomi A2 y Honor 10, con la versión 8 y 9 de Android respectivamente.

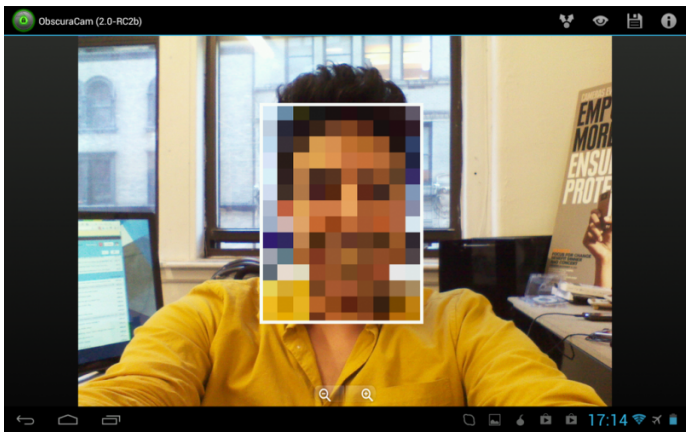



Figura 3.3: Ejemplo de uso ObscuraCam [57]

### 3.2.8. ProofMode

Aplicación que permite de manera sencilla capturar, encriptar y compartir contenido multimedia en particular [25] generado por la cámara y micrófono del dispositivo, de forma verificable en un *smartphone* o *tablet*, manteniendo la seguridad y la privacidad del usuario.

Fue desarrollado para el uso de periodistas o activistas trabajando en situaciones de alto riesgo para grabar pruebas de abuso y violación de derechos. Pensado para facilitar la recogida de testimonio visual que puede servir como prueba fehaciente de abusos y violaciones de derechos humanos.

### 3.2.9. Ripple

Aplicación que implementa una forma de "botón del pánico" [26], de tal forma que manda un mensaje de tipo trigger a la aplicación que selecciones. La aplicación elige que hacer una vez le llegue el mensaje como puede ser bloquearse, ocultarse, borrar información privada y más.

Esta aplicación está enfocada a situaciones donde no haya tiempo para reaccionar y quieres hacer alguna funcionalidad en un clic.

### 3.2.10. Eyewitness

Esta aplicación permite capturar fotos y vídeos los cuales tienen incrustados unos metadatos especiales en el momento de la captura [27]. Estos metadatos los cuales son necesarios para procesos judiciales y que se incluyen de manera inalterable.

Una vez capturados, también ofrecen la capacidad de guardarlo en su servidor. Al subir los vídeos a los servidores de eyewitness, esta organización se ocupa de asegurar la cadena de custodia de confianza, que es el término legal para un registro de quién ha tenido acceso a la información capturada. Esta cadena demuestra que la información original no ha sido cambiada de ninguna manera. Garantizar la seguridad de la información almacenada en sus servidores es la clave para permitir que las fotos y los vídeos se utilizan como pruebas en los tribunales.

Nos dispusimos a probarla y lo primero que hicimos es grabar un vídeo y realizar una foto. Luego los subimos a su servidor y parecía que todo iba bien, pero como lo que grabamos no era contenido de índole jurídica, pues no pudimos llegar más allá con la prueba.

### 3.2.11. SvcAuth

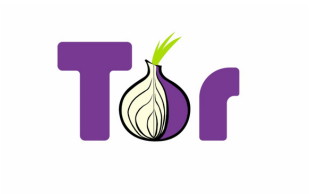
Tal como se ha explicado al final de la sección 1.3, la autenticación de vídeos es un tema todavía poco maduro y objeto de investigación activa. A pesar de ello, ya existe algunas propuestas de solución e incluso unas bibliotecas que las implementan.

SvcAuth [28] es una biblioteca para autenticar transmisiones de vídeo utilizando el estándar de codificación de vídeo H.264/ SVC, punto a punto sin el uso de un servidor. Para dicha autenticación tanto el emisor como los receptores deben conocer el mecanismo de autenticación. Esta biblioteca puede ser empleada como complemento de software sin ningún cambio en la codificación.

## 3.3. Utilidades

### 3.3.1. Red Tor

Sus siglas vienen de “The Onion Router” [29]. El objetivo de esta red es la comunicación superpuesta al internet habitual. La Dark Web, es una parte de Internet oculta con direcciones IP enmascaradas y solo accesibles con navegadores especiales. La red más popular es TOR.



TOR está diseñado para proteger las comunicaciones entre usuarios y para poder así garantizar el anonimato y la privacidad de los datos. Al contrario de un navegador corriente, el envío de los datos no sigue un camino directo del cliente al servidor, sino un camino que pasa por muchos nodos de una red de ordenadores ofrecidas por voluntarios, conjuntamente con el uso de muchas capas de cifrado de los datos y de las cabeceras, en particular de la dirección IP de los ordenadores del camino seguido.

### 3.3.2. Twitter API

API que proporciona la empresa Twitter para poder usar las funcionalidades de sus servicios [30] desde aplicaciones de terceros. Entre sus múltiples funciones se encuentran las de publicar Tweets, ver tu perfil o subir algún contenido multimedia como vídeos.

Dado que Twitter es una red social orientada a la opinión y la difusión de noticias, sería una posible alternativa de repositorio para los vídeos.

### 3.3.3. Netcipher

Es una librería de código abierto para Android que proporciona varios medios para mejorar la seguridad de las redes en aplicaciones móviles [31]. Provee las mejores prácticas de configuración TLS (transport layer security) usando el estándar de Android HttpURLConnection y Apache HTTP Client, dando una simple integración de TOR haciendo fácil la configuración proxy para la conexión HTTP e instancias WebView (vistas nativas de Android que nos permiten visualizar una pagina web) de enlaces de la Dark Web.

### 3.3.4. SecureDrop

SecureDrop [32] es un sistema de código abierto de envío de documentos que los medios de comunicación u otros organismos pueden instalar para recibir información de usuarios anónimos. Originalmente diseñado y desarrollado por Aaron Swartz y Kevin Pulsen.

Este sistema funciona con dos servidores, el primero se encarga de la comunicación cifrada con los proveedores de información y el almacenamiento cifrado del contenido de estas comunicaciones. El segundo, que no está conectado a la red (técnica conocida como “air gapping”), custodia las claves para descifrar los documentos almacenados en el primero. Los periodistas solo pueden consultan los documentos

en un ordenador que no está conectado a la red. Este servicio solo está disponible como un servidor oculto de Tor, que requiere que las fuentes usen un navegador de la misma tecnología. Esto permite el borrado de metadatos del envío de información, consiguiendo que estos no se pueden recuperar y así evitar posibles represalias sobre los emisores del contenido. En la figura 2.4 mostramos gráficamente el funcionamiento de SecureDrop.

Existe una gran variedad de periódicos importantes que disponen de este tipo de sistema, entre ellos The Guardian, The Washington Post, Financial Times y USA Today. Estos ofrecen un repositorio para el envío de archivos de hasta 500MB sobre Tor, permitiendo de una forma sencilla compartir información con ellos.

Todos estos periódicos están disponibles en una API de SecureDrop, con su respectiva URL donde suben los vídeos y algo de información sobre el periódico.

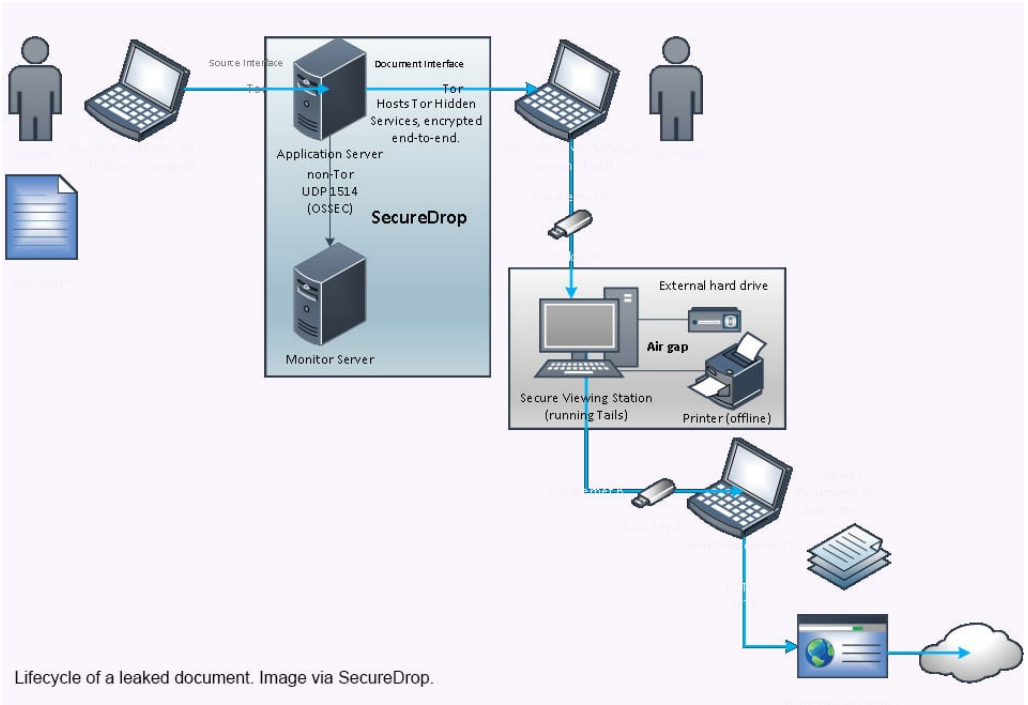



Figura 3.4: Funcionamiento de SecureDrop [58]

### 3.3.5. Guardian Project

Aunque los *smartphones* son un paso adelante en generar comunicación y nuevas formas de comunicación a la vez son un paso atrás en la privacidad, seguridad y anonimato.

Guardian Project [33] crea aplicaciones seguras fáciles de usar, librerías de código abierto y soluciones personalizadas orientadas a las personas que quieren proteger su información personal de intrusión injusta o monitorización.

Entre las aplicaciones o librerías más útiles para nuestro proyecto, se encuentra la anterior mencionada Netcipher, TOR Browser, ProofMode, ObscureCam o Ripple.

# 4. Elección de tecnologías

## 4.1. Tecnologías utilizadas

### 4.1.1. Wifi Aware

Hemos escogido Wifi Aware [10] ya que es una versión mejorada del Wifi Direct, tanto en especificaciones como en menos número de restricciones. Con esta tecnología se intentará solventar los problemas del proyecto que desarrollaron los miembros del TFG del año pasado.

### 4.1.2. Android

Al tratar de una aplicación para dispositivos móviles, nuestra única opción era la utilización de un sistema operativo para dispositivos móviles.


Nos decantamos por desarrollarla para el sistema operativo Android [34] ya que es el sistema con mayor tasa de utilización en lo que se refiere a *smartphone* y nuestro objetivo es llegar al máximo número de dispositivos posibles. Además de los factores ya comentados, cabe destacar que Android da soporte con su API para Wifi Aware como comentaron nuestros compañeros del TFG del año pasado. Se programará para la versión Android 8 y posteriores, ya que es a partir de la que se tiene desarrollado Wifi Aware y de las que disponen los dispositivos que vamos a usar para hacer las pruebas.

### 4.1.3. Android Studio

Al elegir Android como sistema operativo para desarrollar la aplicación, hemos elegido Andorid Studio [35] como entorno de desarrollo ya que es la plataforma oficial ofrecida por Android.

Otro factor a destacar, es el hecho de que tenemos experiencia en este programa, no es de pago, da muchas facilidades para crear interfaces gráficas de usuario y tiene la capacidad de probar la aplicación en diferentes versiones de Android y en los dispositivos físicos que tenemos.

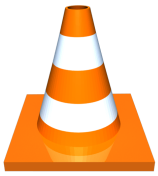
### 4.1.4. Java

Java [36] es un lenguaje de programación multiplataforma orientado a objetos.


Este es uno de los lenguajes mas utilizados en la actualidad y es el lenguaje en el que mas experiencia tenemos. Además mayor parte de la documentación de Android y Wifi Aware está totalmente disponible en Java mientras que en otros lenguajes no está completa o es inexistente.

### 4.1.5. LibVLC

LibVLC [37] es una librería de código abierto para la gestión de contenido multimedia.



Se puede integrar de manera fácil en nuestra aplicación y nos puede ofrecer muy buenos resultados. Es necesaria en el proyecto ya que compartimos audio y vídeo a través de los dispositivos y necesitamos una manera de poder gestionar dicho contenido.

### 4.1.6. RTSP

El protocolo de transmisión en tiempo real, es un protocolo que permite el control de la transmisión de vídeo o audio y que funciona a través de varios protocolos de transporte como UDP Y TCP, aunque el que nos interesa es el primero de estos.

Este protocolo trabaja a nivel de aplicación y controla que la entrega de datos se realiza correctamente entre clientes ya que el contenido con el que se trabaja normalmente al hacer retransmisiones es muy sensible a la sincronía temporal o a la falta de ella.

### 4.1.7. RTP sobre UDP

Las siglas de RTP significan Real Time Transport Protocol. Es un protocolo que define un formato de paquete estándar para el envió de vídeo y audio sobre internet. Lo utilizaremos en nuestro proyecto, ya que este protocolo se utiliza en sistemas de comunicación y en aplicaciones de transmisión de datos.

UDP es un protocolo de datagramas de usuario que se encarga del transporte de paquetes. Este protocolo mantiene el tiempo de transmisión de paquetes muy bajo permitiéndonos enviar información rápidamente entre dispositivos. A pesar de que el protocolo no asegura que el envío de un paquete llegue completo, si permite conocer aquellos paquetes defectuosos mediante la suma de verificación.

Creemos que es la mejor elección para nuestro proyecto ya que este protocolo nos permite la transmisión de paquetes de manera muy veloz y el hecho de perder algún paquete implicaría perder algún fotograma, lo que no es un gran problema. Aunque HTTP Streaming también nos serviría, como ya investigaron los alumnos del proyecto del curso pasado [3], no nos interesa facilitar el salto a cualquier punto del *stream* o la capacidad que nos ofrece de una tasa de bits adaptable (la calidad se adapta al ancho de banda del usuario) ya que la perdida de algún paquete no nos impide transmitir vídeo y audio de manera efectiva.

Por esas razones, no nos importa usar los protocolos RTSP y RTP, aunque sean un poco anticuados para hacer *streams* de vídeo, en vez del actualmente más popular HTTP Streaming.

### 4.1.8. Libstreaming

Libstreaming [38] es un API que permite con unas pocas lineas de código transmitir la cámara y micrófono de nuestro dispositivo Android usando RTP sobre UDP. Hemos elegido esta librería frente a las varias opciones que había, debido a que está implementada en Java para Android.

### 4.1.9. NetCipher

NetCipher, como mencionamos anteriormente, es una librería de código abierto que permite acceder a la red Tor. Vamos a usar esta librería para poder llevar a cabo la subida de vídeos de forma anónima a las plataformas correspondientes de los medios de comunicación. Esta tiene una fácil implementación en una aplicación Android ya que está diseñada para ello.

## 4.2. Herramientas utilizadas

### 4.2.1. Git y Github

Git [39] es un software de control de versiones pensado en la eficiencia y fiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número

de archivos de código fuente. Esta herramienta nos permite llevar un registro en los cambios que se producen en los diferentes archivos y facilita el trabajo de forma colaborativa minimizando conflictos.


Github [40] es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones de Git. esta plataforma permite alojar de manera gratuita múltiples repositorios de código.

Hemos decidido usar estas herramientas dado que todos los miembros del equipo hemos trabajado anteriormente con estas y nos parecen sencillas y familiares además de ajustarse a nuestro proyecto.

### 4.2.2. Trello

Trello [41] es un servicio en línea de gestión de tareas basado en Kanban. Los usuarios pueden crear sus tableros de tareas con varias columnas y mover las tareas entre ellas. Típicamente las columnas incluyen los estados de las tareas: To Do, In progress, Done. Trello tiene una variedad de usos como por ejemplo la administración de proyectos software y el diseño web.


Nos hemos decantado por usar Trello ya que lo hemos utilizado en anteriores proyectos de la universidad y por su facilidad de uso. Con esta herramientas nos administraremos nuestras tareas para así poder ver en todo momento que están haciendo los integrantes del equipo y que queda por hacer.

### 4.2.3. TEXIS y LATEX

LaTeX [43] es un lenguaje y sistema de maquetación de documentos.


Es una colección de comandos diseñados para facilitar el uso del "procesador de palabras" TeX de Donald Knuth. LaTeX permite escribir documentos que se formatean automáticamente para ajustarse lo más posible a las normas tipográficas. Una característica distintiva de LaTeX es su modo matemático, que permite componer fórmulas complejas.

Aunque LaTeX era un lenguaje desconocido por nosotros, gracias a la plantilla TeXiS [42], hecha por profesores de esta facultad, nos ha sido mas sencillo poder aprenderlo para llevar a cabo la memoria.

### 4.2.4. Overleaf

Overleaf [44] es una aplicación web la cual permite poder editar documentos maquetados en látex y compilarlos para su posterior transformación en PDF.


Con la versión gratuita se puede tener documentos con hasta 4 autores, lo cual era mas que suficiente para la escritura de nuestra memoria. Es fácil e intuitiva de usar y el poder escribir simultáneamente nos ha facilitado mucho el trabajo.

### 4.2.5. Drive

Drive [45] es una herramienta de Google en la nube, muy útil para rellenar documentos de forma interactiva y simultánea y un lugar donde guardar documentos de forma online.


La hemos usado sobre todo para ir apuntando referencias, artículos de interés y para alojar tanto manuales como documentación. Hemos decidido elegir esta herramienta de almacenamiento en la nube frente a otras ya que la usamos asiduamente y gracias a las cuentas de correo de la facultad disponemos de almacenamiento ilimitado.

### 4.2.6. Discord


Discord [46] una aplicación de comunicación de voz sobre IP gratuita disponible tanto en su versión web, móvil como aplicación de escritorio. Hemos utilizado esta aplicación para comunicarnos durante la situación de confinamiento causada por el COVID-19.
La razón por la que hemos decidido utilizar esta aplicación sobre las demás disponibles es debido al hecho de ser gratuita además del gran uso que le damos día a día lo que nos facilita su uso.

### 4.2.7. TeamViewer

TeamViewer [47] es una aplicación de fácil acceso, que permite conectarse por control remoto a otros equipo. Entre sus funciones están control remoto , uso compartido de escritorio, reuniones en línea, conferencias web y transferencia de archivos entre ordenadores.


Existen versiones diversos sistemas operativos. También es posible el acceso a un equipo remoto mediante un navegador web. Aunque el principal uso de la aplicación es el control remoto, también incluye funciones de trabajo en equipo y presentación.

Hemos usado TeamViewer para desarrollar nuestra aplicación y así poder colaborar todos los miembros, ya que debido al estado de alarma y a la división de los dispositivos, solo uno de los integrantes del equipo podía depurar y solucionar los errores que nos han ido surgiendo.

# 5. Especificación de requisitos

## 5.1. Especificación a alto nivel

La funcionalidad principal de la aplicación es la conexión entre varios dispositivos móviles mediante Wifi Aware para permitir la transmisión de vídeo y audio.

En la aplicación existirá un emisor principal (**Publisher**) que transmitirá a uno a varios receptores que actuarán de nodo intermedio, estos a su vez, podrán retransmitir a otros nodos intermedios hasta llegar a alguno que disponga de conexión a internet (**nodo gateway**) que podrá compartir el *stream* mediante la red Tor, ideal en el escenario Witness, para poder denunciarlo públicamente.

Cualquiera de los nodos podrá tener la opción de guardar el vídeo por si cualquiera de los otros no llega a internet.

En esta aplicación hemos distinguido cinco roles distintos según la finalidad de uso:

- **Publisher**: es el creador del vídeo original que se retransmite mediante la red de nodos.
- **Subscriber**: usuario que ve el vídeo retransmitido pero no lo comparte a más nodos.
- **Nodo de tránsito**: usuario intermedio que re-emite un vídeo recibido con otros dispositivos. Este vídeo puede provenir de un Publisher o de otros nodos de tránsito.
    - **Simple**: Retransmite a un solo nodo

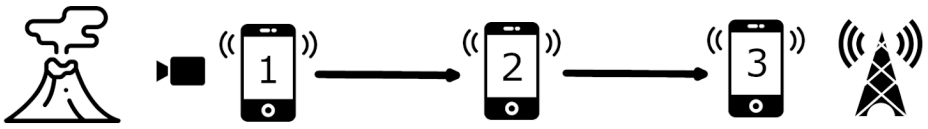



Figura 5.1: Especificación nodo de tránsito simple

- **Completo**: Retransmite a todos los nodos posibles.

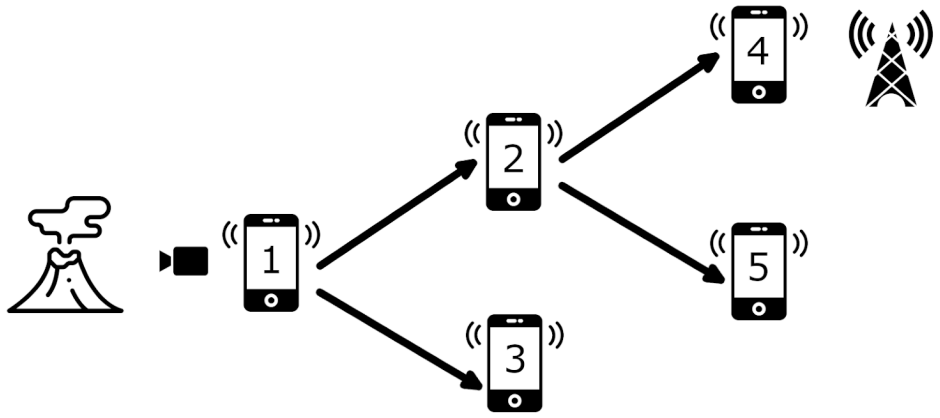



Figura 5.2: Especificación nodo de tránsito completo

- **Nodo gateway**: usuario que ve el vídeo retransmitido y además lo comparte por internet.
- **Usuario no conectado**: usuario que todavía no ha establecido conexión con otros dispositivos.

Debido al retraso a la hora de recibir los teléfonos móviles y a la poca documentación y ejemplos sobre Wifi Aware , nos vimos obligados a definir 2 escenarios de requisitos.

Según la documentación oficial de Wifi Aware los dispositivos son capaces de buscar y conectarse con otros automáticamente sin la necesidad de interacción por parte del usuario. No obstante, estas funcionalidades están sujetas a no implementarse, ya que los requisitos especificados por Wifi Aware contienen partes opcionales, las cuales pueden no llegar a implementarse por parte del fabricante.

Los dos escenarios según el funcionamiento de Wifi Aware son:

- En el **primer escenario** suponemos que para los dispositivos utilizados, la implementación de Wifi Aware, dispondrá de la mayoría de la funcionalidad descrita en la documentación oficial, por lo tanto los dispositivos son capaces de buscar otros dispositivos cercanos y conectarse automáticamente para

comenzar la transmisión. Obviando la necesidad de que el usuario tenga que elegir manualmente a qué dispositivo conectarse. Este primer escenario es menos restrictivo y suprimiría la implementación de algunas historias de usuario.

- En el **segundo escenario** suponemos que la implementación de Wifi Aware en los dispositivos utilizados no dispone de todas las funcionalidades y funciona algo mas parecido a su predecesora, Wifi Direct, en la cual tienes que hacer la conexión desde los ajustes. Por lo tanto, hemos definido las tres primeras historias de usuario para especificar un comportamiento en el cual el usuario tiene que hacer la conexión explicita pero desde la propia aplicación. Este, sin llegar a ser el de Wifi Direct, sería un punto medio entre el escenario uno y Wifi Direct.

Finalmente una vez tuvimos los dispositivos y empezamos a hacer pruebas con Wifi Aware vimos que el primer escenario era viable por lo cual decidimos descartar el segundo.

## 5.2. Requisitos

Para la especificación de requisitos hemos decidido seguir una forma de especificación ágil, usando historias de usuario. Las hemos listado según el orden de implementación que deberíamos seguir.

Hemos visto este tipo de especificación la mas idónea para nuestro proyecto ya que al desconocer la tecnología y sus restricciones, no sabemos del todo que funcionalidades se llegarán a poder implementar.

En algunas de las historias de usuario, debido a ser de mas relevancia, hemos visto conveniente dibujar diagramas de secuencia para apoyar la especificación. Puesto que muchos diagramas realizan funcionalidades triviales, para algunos de ellos no vimos necesaria su especificación, dejando solo aquellos que nos aportan valor a los requisitos.

---

**HU-1**

---

Como **usuario no conectado** quiero poder activar Wifi Aware para poder encontrar dispositivos cercanos a mi.

---

- **Dado** un **usuario no conectado**
- **Cuando** inicia la aplicación y no dispone de Wifi Aware

- **Entonces** se le avisará de que no dispone de Wifi Aware y por lo tanto no podrá usar las funcionalidades de la aplicación.

- **Dado** un usuario no conectado
- **Cuando** abre la aplicación y dispone de Wifi Aware
- **Entonces** se le mostrará el estado en el que se encuentra. (Activo/desactivo).

- **Dado** un usuario no conectado
- **Cuando** abre la aplicación y dispone de Wifi Aware y pulsa el botón de activo/-desactivo de Wifi Aware
- **Entonces** se pondrá el estado contrario al que esté actualmente. (Activo/desactivo).

---

**HU-2**

---

Como **Publisher** quiero poder transmitir el contenido de la cámara y el micrófono de mi dispositivo a los nodos cercanos para que puedan visualizar lo que estoy grabando.

**NOTA: La implementación de esta historia de usuario se realizará de manera incremental haciendo primero un Publisher simple, el cual transmite a un solo nodo cercano, y posteriormente, uno completo que retransmite a todos los cercanos.**

---

- **Dado** un **Publisher**
- **Cuando** pulsa el botón de abrir cámara y no ha dado los permisos de la cámara y el micrófono
- **Entonces** se le mostrará una solicitud de permisos de cámara y micrófono.

- **Dado** un **Publisher**
- **Cuando** acepte la solicitud de permiso de cámara y micrófono
- **Entonces** se le mostrará lo que visualiza su cámara y un botón para empezar la transmisión.

- **Dado** un **Publisher**
- **Cuando** cancele la solicitud de permiso de cámara y micrófono
- **Entonces** se le mostrará un mensaje “No será posible transmitir si no concede los permisos solicitados” y se le devolverá a la pantalla de inicio.

- **Dado** un **Publisher**
- **Cuando** pulsa el botón de abrir cámara y ya haya concedido los permisos de la cámara
- **Entonces** se le mostrará lo que visualiza su cámara y un botón para empezar la transmisión.

- **Dado** un **Publisher**
- **Cuando** pulsa el botón de empezar transmisión dentro de la vista de la cámara
- **Entonces** empezará la transmisión y estará disponible para los demás dispositivos.

- **Dado** un **Publisher**
- **Cuando** está transmitiendo y pulsa sobre el botón de cancelar transmisión
- **Entonces** dejará de transmitir, de estar disponible para otros dispositivos y se le devolverá la pantalla de inicio.

- **Dado** un **Publisher**
- **Cuando** está transmitiendo y cierra la aplicación / apaga el dispositivo
- **Entonces** dejará de transmitir y de estar disponible para otros dispositivos.

---

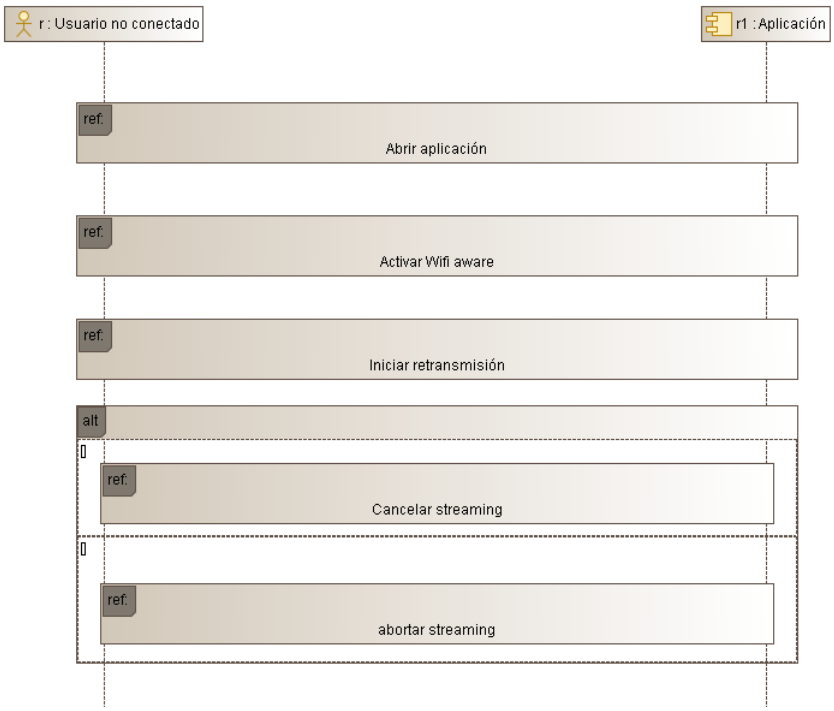



Figura 5.3: Diagrama flujo Historia de usuario 2

**HU-3**

Como **Subscriber/nodo de tránsito/nodo gateway** quiero poder ver los dispositivos cercanos que están transmitiendo para ver los que están compartiendo

- **Dado** un **Subscriber/nodo de tránsito/nodo gateway**
- **Cuando** entre a la pestaña de transmisiones disponibles
- **Entonces** se le mostrará la lista de dispositivos cercanos que están transmitiendo.

- **Dado** un **Subscriber/nodo de tránsito/nodo gateway**
- **Cuando** entre a la pestaña de transmisiones disponibles y no haya ninguna transmisión disponible
- **Entonces** se le mostrará un mensaje de "no hay transmisiones disponibles".

**HU-4**

Como **Subscriber/nodo de tránsito/nodo gateway** quiero poder unirme a la transmisión de cualquiera de los nodos cercanos para poder visualizar lo que están emitiendo.

- **Dado** un **Subscriber/nodo de tránsito/nodo gateway**
- **Cuando** entre a la pestaña de transmisiones disponibles y seleccione una de estas
- **Entonces** se le abrirá una nueva ventana donde visualizará la transmisión.

- **Dado** un **Subscriber/nodo de tránsito/nodo gateway**
- **Cuando** entre a la pestaña de transmisiones disponibles, seleccione una y no se haya podido visualizar la transmisión
- **Entonces** se le mostrará el mensaje: "No se ha podido visualizar la transmisión".

- **Dado** un **Subscriber/nodo de tránsito/nodo gateway**
- **Cuando** esté visualizando la transmisión y el Publisher deje de transmitir
- **Entonces** se le mostrará el mensaje: "La transmisión ha finalizado".

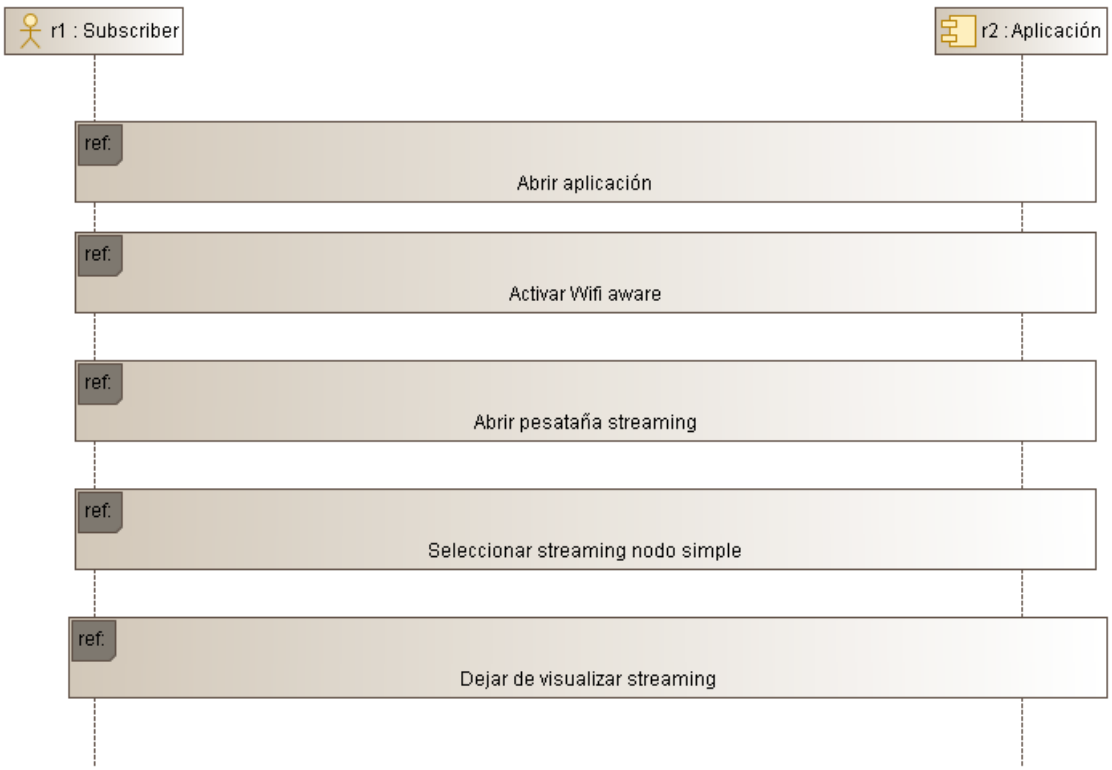



Figura 5.4: Diagrama flujo Historia de usuario 4

---

**HU-5**

---

Como **nodo de tránsito** quiero poder compartir la transmisión que estoy visualizando a los nodos cercanos para que puedan unirse a esta.

**NOTA: La implementacion de esta historia de usuario se realizará de manera incremental haciendo primero un nodo de tránsito simple, el cual transmite a un solo nodo cercano, y posteriormente, uno completo que retransmite a todos los cercanos.**

---

- **Dado** un **nodo de tránsito**
- **Cuando** entré en la pestaña de preferencias
- **Entonces** se le mostrará la opción de si quiere compartir la transmisión a la vez que visualizas.

- **Dado** un **nodo de tránsito**
- **Cuando** seleccione la emisión que quiere visualizar y haya seleccionado que quiere retransmitir
- **Entonces** a la vez que visualiza estará disponible para retransmitir a otros dispositivos.

- **Dado** un **nodo de tránsito**
- **Cuando** está retransmitiendo y cierra la aplicación / apaga el dispositivo
- **Entonces** dejará de retransmitir y de estar disponible para otros dispositivos.

- **Dado** un **nodo de tránsito**
- **Cuando** minimiza la aplicación o bloquea el dispositivo y tiene la opción de retransmitir activada
- **Entonces** seguirá retransmitiendo a los dispositivo conectados a su *streamcolor*.

---

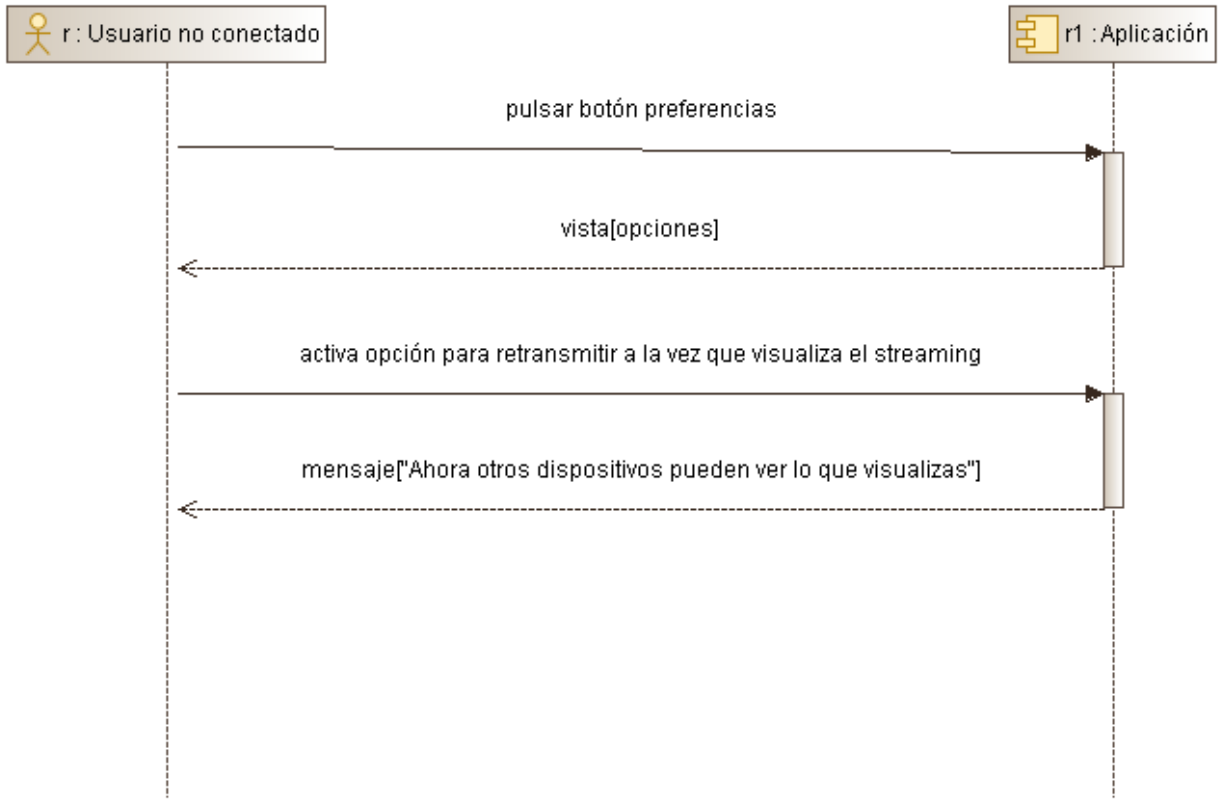



Figura 5.5: Diagrama flujo Historia de usuario 5: Selección de preferencias

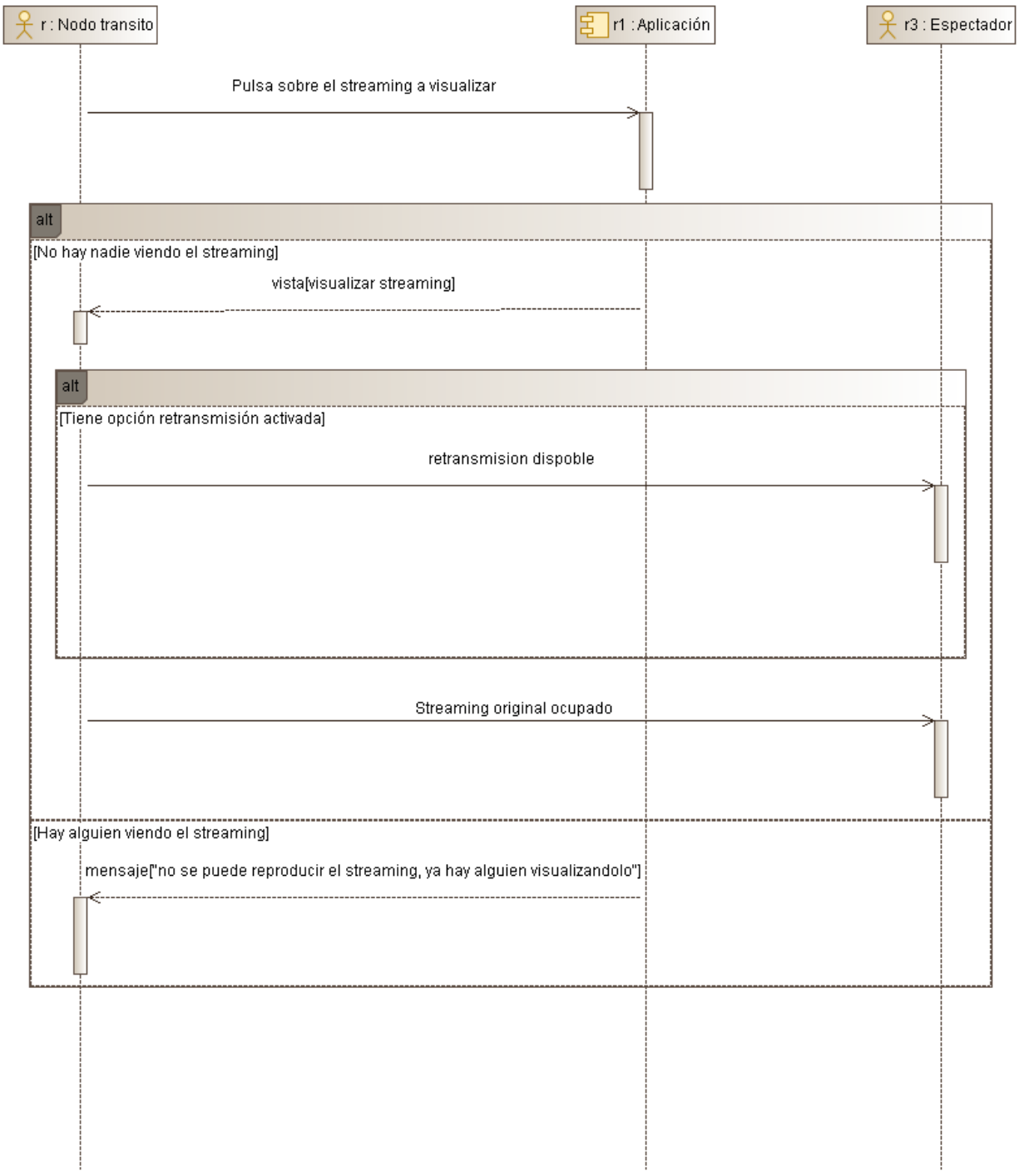



Figura 5.6: Diagrama flujo Historia de usuario 5: Comportamiento nodo tránsito

**HU-6**

Como **nodo de tránsito** quiero poder compartir automáticamente la transmisión que estoy recibiendo para que más dispositivos puedan unirse a esta.

- **Dado** un **nodo de tránsito**
- **Cuando** esté activa la opción de nodo de tránsito automático
- **Entonces** se conectará a la primera transmisión disponible y empezará a compartirlo de forma automática.

- **Dado** un **nodo de tránsito**
- **Cuando** minimiza o bloquea el dispositivo
- **Entonces** seguirá retransmitiendo a los dispositivos que estén conectados a su *stream*.

**HU-7**

Como **Publisher** quiero poder guardar el vídeo que estoy transmitiendo en mi dispositivo para poder compartirlo en un futuro.

- **Dado** un **Publisher**
- **Cuando** entre en la pestaña de preferencias
- **Entonces** se le mostrará la opción de si quiere guardar la transmisión a la vez que la comparte.

- **Dado** un **Publisher**
- **Cuando** pulse el botón de transmitir y haya concedido los permisos de almacenamiento
- **Entonces** se le guardará el vídeo a la vez que transmite.

- **Dado** un **Publisher**
- **Cuando** pulse el botón de transmitir, esté seleccionada la opción de guardar y no haya concedido los permisos de almacenamiento
- **Entonces** se le pedirán permisos de almacenamiento.

- **Dado** un **Publisher**
- **Cuando** cancele los permisos de almacenamiento
- **Entonces** se le mostrará el mensaje: “No se posible guardar el vídeo sin los permisos de almacenamiento”.

- **Dado** un **Publisher**
- **Cuando** acepte los permisos de almacenamiento y tenga activada la opción de guardar vídeo
- **Entonces** se le guardará el vídeo a la vez que transmite.

---

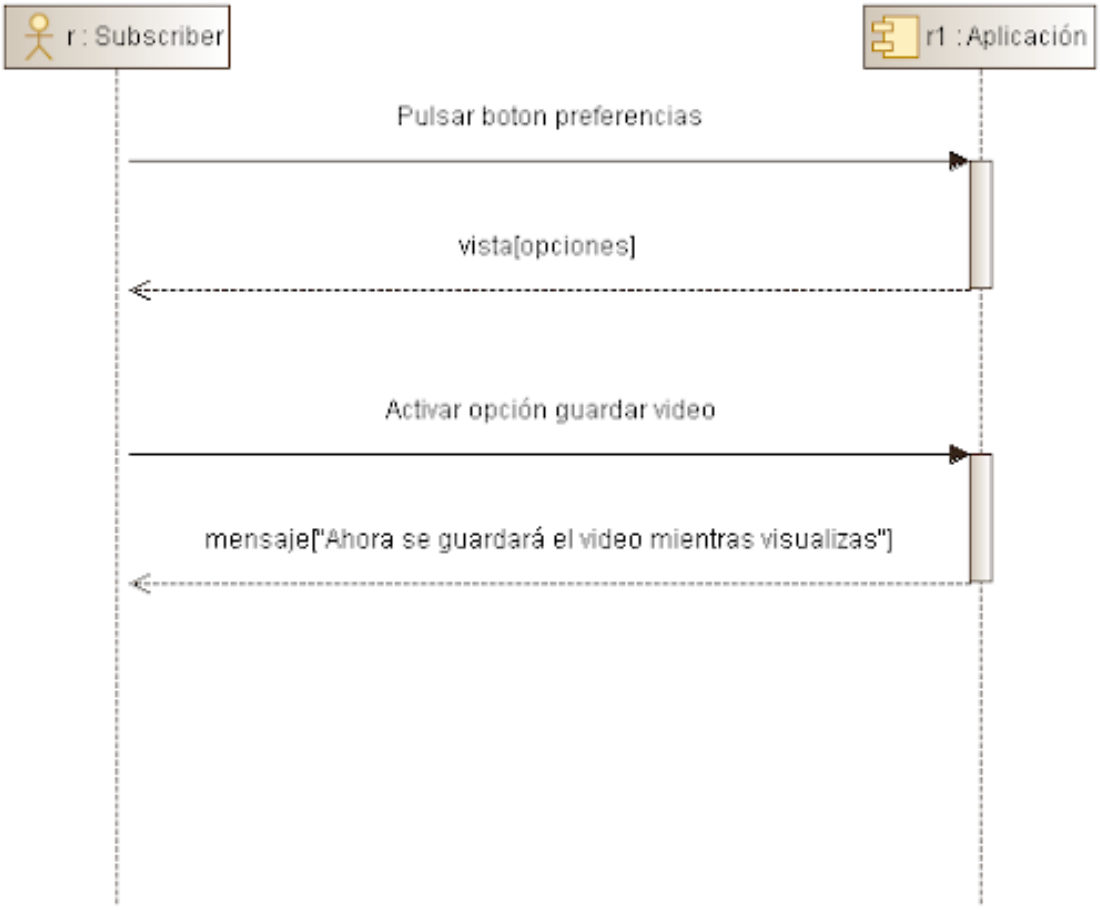



Figura 5.7: Diagrama flujo Historia de usuario 7: Selección de preferencias

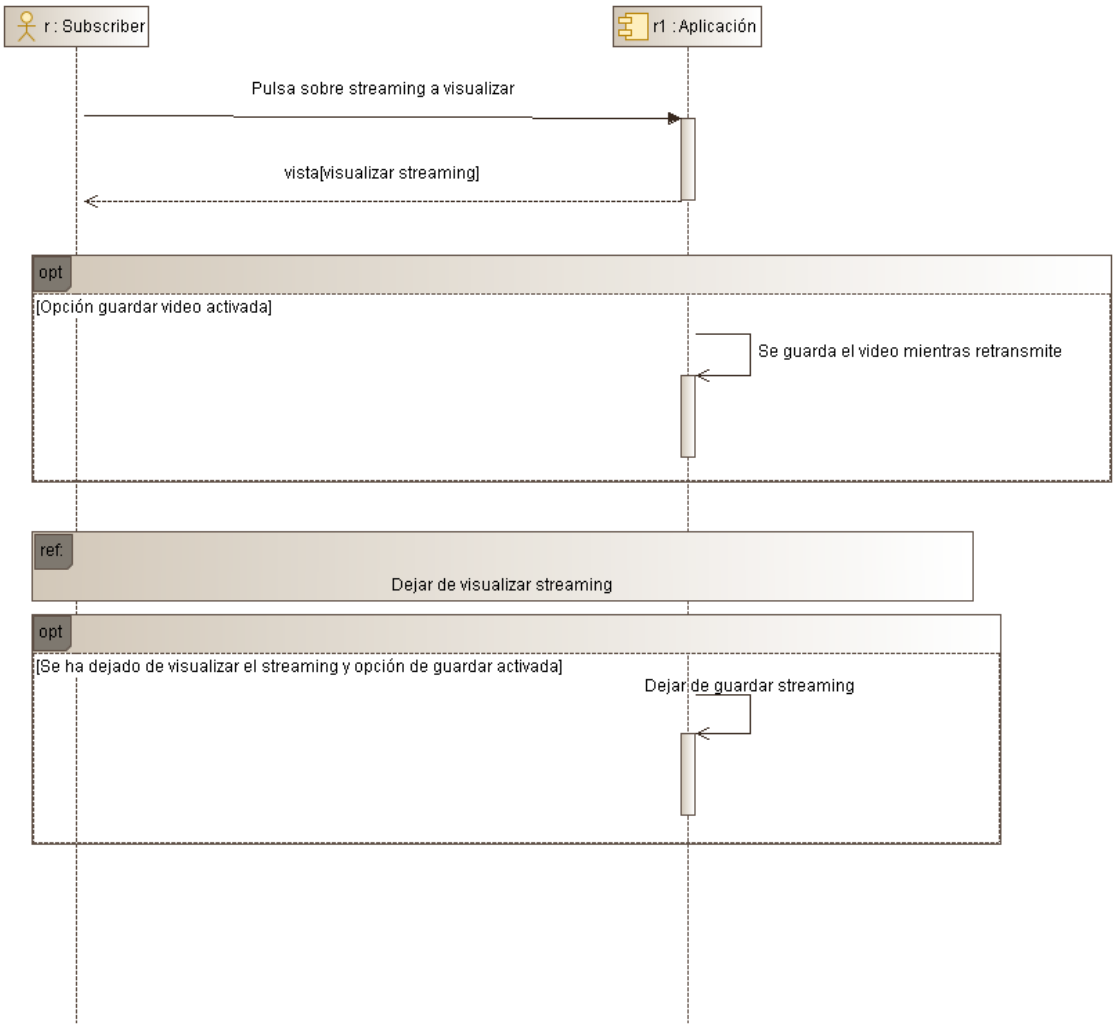



Figura 5.8: Diagrama flujo Historia de usuario 7: Guardar vídeo

---

**HU-8**

---

Como **Subscriber/nodo de tránsito/nodo gateway** quiero poder guardar el video que estoy visualizando para poder compartirlo en un futuro.

---

- **Dado** un **Subscriber/nodo de tránsito/nodo gateway**
- **Cuando** se conecte a una una transmisión, tenga activada la opción de guardar video y haya dado los permisos de almacenamiento
- **Entonces** se le guardará el vídeo a la vez que está conectado a una transmisión.

- **Dado** un **Subscriber/nodo de tránsito/nodo gateway**
- **Cuando**se conecte a una transmisión, tenga la opción de guardar seleccionada y no haya concedido los permisos de almacenamiento
- **Entonces** se le pedirán permisos de almacenamiento.

---

**HU-9**

---

Como **nodo gateway** quiero poder subir la transmisión que estoy visualizando a internet para compartirla a más gente.

---

- **Dado** un **nodo gateway**
- **Cuando** entre en la pestaña de preferencias
- **Entonces** se le mostrará la opción de si quiere subir a internet la transmisión a la vez que esta conectado a esta.

- **Dado** un **nodo gateway**
- **Cuando** está conectado a una transmisión, tiene la opción de subir a internet activada y dispone de conexión a internet
- **Entonces** se subirá el video a internet mientras está conectado a la transmisión.

---

**HU-10**

---

Como **Publisher/Subscriber/ nodo gateway/nodo de tránsito** quiero poder ocultar rápidamente la visualización de la transmisión para poder evitar que me descubran utilizando la aplicación en entornos hostiles.

---

- **Dado** un **Publisher**
- **Cuando** esté transmitiendo
- **Entonces** se le mostrará la opción de si quiere ocultar la previsualización de su cámara

- **Dado** un **Subscriber/nodo gateway/nodo de tránsito**
- **Cuando** esté visualizando una transmisión
- **Entonces** se le mostrara la opción de ocultarla.

- **Dado** un **Publisher/Subscriber/ nodo gateway/nodo de tránsito**
- **Cuando** esté ocultando una transmisión
- **Entonces** se le mostrara la opción de volver a mostrarla.

---

**HU-11 (Relevante en escenario Humanitarian)**

Como **Publisher** quiero poder difuminar las caras de las personas que aparecen en la transmisión para proteger su privacidad

---

- **Dado** un **Publisher**
- **Cuando** entre en la pestaña de preferencias
- **Entonces** se le mostrará la opción de si quiere o no difuminar las caras que aparecen en la transmisión.

- **Dado** un **Publisher**
- **Cuando** está transmitiendo y tiene activada la opción de difuminar las caras
- **Entonces** las caras de las personas que aparecen en la transmisión se difuminarán.

---

**HU-12 (Relevante en escenario Witness)**

Como **Publisher/Subscriber/ nodo gateway/nodo de tránsito** quiero poder borrar metadatos de las transmisiones para evitar posibles represalias

---

- **Dado** un **Publisher/Subscriber/nodo gateway/nodo de tránsito**
- **Cuando** entré en la pestaña de preferencias
- **Entonces** se le mostrará la opción de si quiere o no borrar los metadatos de las transmisiones emitidas y recibidas.

- **Dado** un **Publisher/Subscriber/nodo gateway/nodo de tránsito**
- **Cuando** comparte o retransmite y tiene activada la opción de borrar los metadatos
- **Entonces** los metadatos del vídeo se borrarán a la vez que retransmite o comparte.

---

**HU-13 (Relevante en escenario Witness)**

Como **Subscriber/nodo gateway/nodo de tránsito** quiero poder verificar que el vídeo que estoy visualizando/compartiendo es el vídeo original y no está siendo modificado.

- **Dado** un **Subscriber/nodo gateway/nodo de tránsito**
- **Cuando** entré en la pestaña de preferencias
- **Entonces** se le mostrará la opción de si quiere o no verificar los vídeos que está visualizando/compartiendo.

- **Dado** un **Subscriber/nodo gateway/nodo de tránsito**
- **Cuando** visualice una transmisión y tenga activada la opción de verificar el vídeo
- **Entonces** se verificará el vídeo mientras se visualiza/comparte esta.

**HU-14**

Como **Publisher/Subscriber/nodo gateway/nodo de tránsito** quiero poder visualizar en la aplicación las transmisiones almacenadas para tenerlas mejor localizadas.

- **Dado** un **Publisher/Subscriber/nodo gateway/nodo de tránsito**
- **Cuando** entré en la pestaña de galería
- **Entonces** se le mostrará todos las transmisiones almacenadas con una miniatura para su previsualización

- **Dado** un **Publisher/Subscriber/nodo gateway/nodo de tránsito**
- **Cuando** pulse sobre la minaturia de un vídeo
- **Entonces** se reproducirá este

## 5.3. Mockups

Durante la fase de especificación de requisitos también diseñamos mockups de la interfaz gráfica de nuestra aplicación. Estos nos servirían de guía a la hora de programar la GUI. Entre estos incluimos unas breves instrucciones sobre el funcionamiento de la aplicación, además de un menú lateral que nos permite navegar sobre las distintas partes de esta. Una vista que nos permite ver las transmisiones cercanas a las que nos podemos conectar, una pantalla de ajustes, donde podremos activar o desactivar las distintas características que proporciona la aplicación. Por último se encuentra la galería de vídeos que hemos decidido guardar.


Figura 5.9: Pantalla de ajustes de la aplicación


Figura 5.10: Menu lateral de la aplicación


Figura 5.11: Pantalla de transmisiones disponibles


Figura 5.12: Pantalla de galería

# 6. Desarrollo

## 6.1. Introducción

A causa de la tardanza de la compra de los dispositivos llegando estos a principios de marzo sumada a la crisis del COVID-19, nos enfrentamos a una serie de reajustes en la planificación y en el modelo de trabajo, lo que dio lugar a no poder llegar a implementar algunos de los requisitos especificados.

Inicialmente dividimos los cuatro dispositivos entre dos personas para poder investigar a fondo la implementación de Wifi Aware en Android [53]. Lo que dio lugar a que durante el estado de alarma nos dispusiéramos de los cuatro dispositivos a la vez para poder implementar los nodos completos y de tránsito. Finalmente debido a la incertidumbre sobre la duración del estado de alarma y la incapacidad de avanzar, nos vimos obligados a enviarlos por correo con el fin de progresar con el desarrollo.

Para el desarrollo de la aplicación diseñamos una metodología de trabajo y un proceso de desarrollo que se ajuste a la situación extraordinaria. En los siguientes puntos procedemos a explicar en detalle sus características

## 6.2. Metodología

Dada la naturaleza de investigación de este proyecto hemos seguido una metodología ágil, la cual nos permite focalizarnos en los problemas que pueden ir surgiendo durante el transcurso del desarrollo. Esto implica que durante el desarrollo, iremos diseñando a la par que codificando, ya que incluso los requisitos descritos anteriormente podrían llegar a cambiar según las restricciones y necesidades que vayamos descubriendo.

Nuestra metodología mezcla características de las ya existentes, como por ejemplo XP, ya que a la hora de programar afrontar los problemas lo hacíamos como un equipo y no individualmente ya que necesitábamos utilizar hardware al cual no teníamos acceso todos los miembros del equipo.

Por otro lado también aplicamos características de Kanban [54], como es un tablero para organizar y administrar las tareas, además de no seguir plazos o entregas estrictas, ya que solo tenemos una fecha límite final. Los avances se los comunicábamos al director una vez estaban dados por finalizados en forma de reuniones, tanto presenciales como telemáticas, en periodos de dos semanas.

Para la realización del tablero, hemos usado la herramienta Trello, en la cual hemos añadido las historias de usuario ordenadas en función del valor que aporta cada una

de ellas a la aplicación. Aunque nos hemos visto obligados a modificar su orden alguna vez debido a la situación excepcional comentada en el sección 6.1, hemos seguido el orden esperado en la mayor parte del desarrollo. En la siguiente figura mostramos un ejemplo del tablero en Trello a mitad de desarrollo.

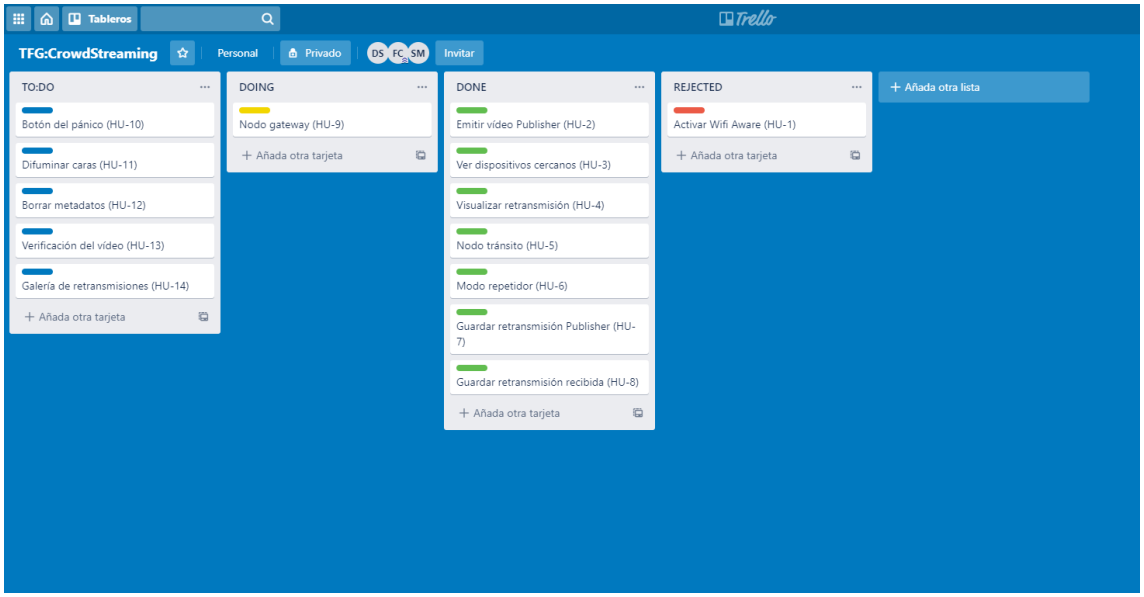



Figura 6.1: Tablero de Trello

## 6.3. Conceptos básicos de Android

A la hora de programar una aplicación Android en Java, debemos tener en cuenta varios factores los cuales no se ven a la hora de programar interfaces gráficas en otras plataformas. La componente gráfica principal de una aplicación Android se denomina *Activity*, la cual está compuesta en dos partes, la parte XML en la cual se define la interfaz y la parte Java que define su comportamiento.

La parte más relevante es la parte Java, en la cual se define el ciclo de vida de la *Activity*. En este parte hay que tener en cuenta como va a actuar la vista en los diferentes eventos que se pueden producir sobre esta. En la siguiente figura se refleja el ciclo de vida completo de una *Activity*, pero nos vamos a centrar únicamente en los utilizados en el desarrollo del proyecto.

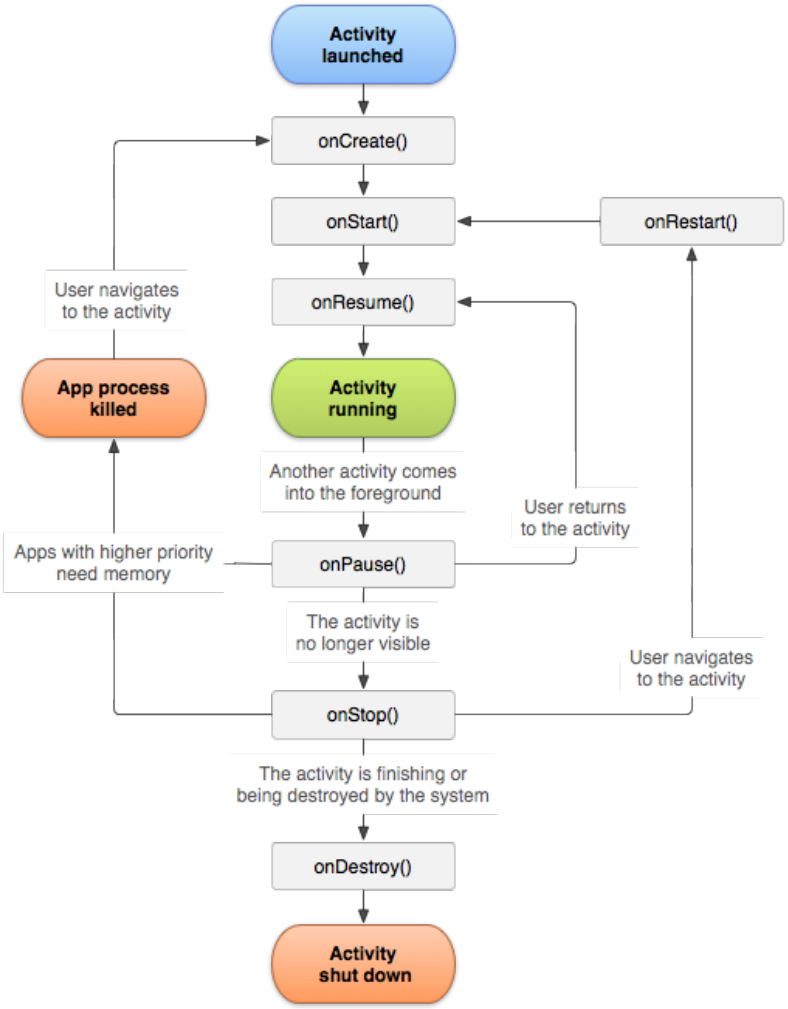



Figura 6.2: Ciclo de vida de una *Activity* en Android [59]

La función mas importante es "onCreate()", la cual se invoca cuando la vista es creada, normalmente se utilizar para inicializar las estructuras datos y los componentes gráficos. Por otro lado, existe la función "onResume()", la cual es llamada cuando el usuario va a interactuar con la *Activity*, por ejemplo cuando cambias de *Activity* y vuelves a una anterior. Las demás son menos importantes pero pueden ser útiles en determinadas situaciones como parar la vista.

Hemos hablado de *Activity*, pero los *Fragments* son igual de importantes. Una *Activity* puede contener varios *Fragments*, los cuales tienen un comportamiento y ciclo de vida algo diferente. Su finalidad es ampliar la navegación entre pantallas ya que nos permite tener diferentes vistas dentro de una misma *Activity*. Aunque posee varios métodos en común tiene algunos específicos los cuales hemos usado, en particular es

la función "onCreateView()" el cual es llamada cuando se dibuja por primera vez el *Fragment* en la interfaz de usuario.

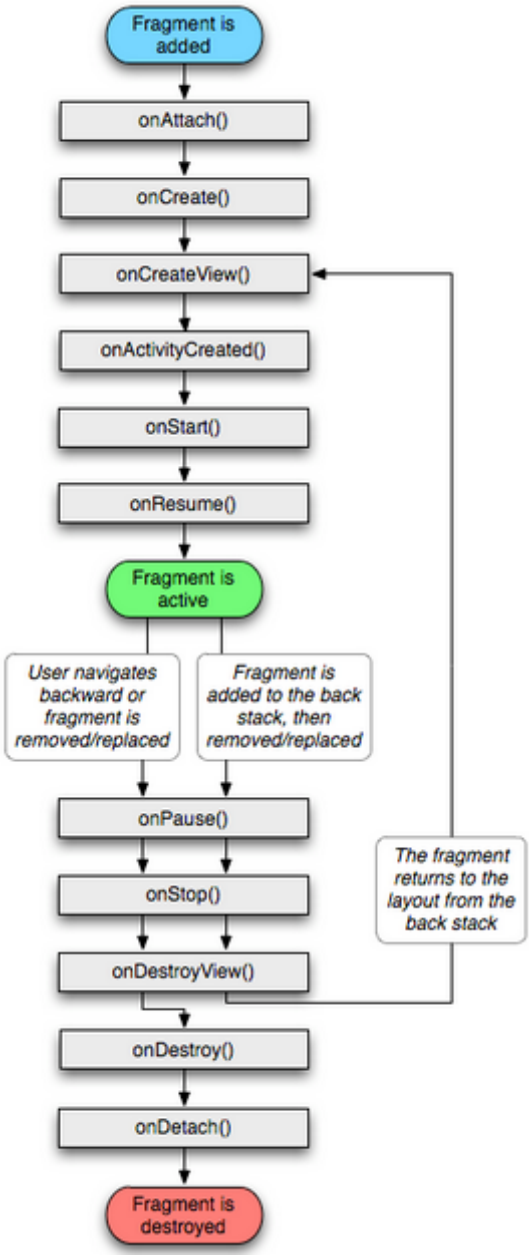



Figura 6.3: Ciclo de vida de un *Fragment* en Android [60]

## 6.4. Proceso de desarrollo en Android

Debido a que las aplicaciones se deben ejecutar en dispositivos móviles, en vez de hacerlo desde el mismo ordenador donde se esta programando, programar en Android requiere de unos conocimientos extra que tuvimos que adquirir.

Hay dos formas típicas de depurar una aplicación en Android: usando el emulador propio de Android Studio o conectando un móvil Android al ordenador vía USB. En ambos casos, Android Studio hace una instalación de la aplicación en el dispositivo(virtual o físico), pero permite mostrar la información de depuración en la maquina a la que esta conectado.

En nuestro caso, al usar Wifi Aware para desarrollar el proyecto, era obligatoria la ejecución conectando los dispositivos mediante cable USB, ya que el emulador no ofrece dispositivos virtuales que emulen Wifi Aware.

Por lo tanto, a la hora de probar nuestra aplicación, cada vez que añadíamos nuevas funcionalidades, debíamos realizar esta instalación en los dispositivos para así comprobar el correcto funcionamiento de la aplicación.

Además, Andorid Studio te ofrece las funcionalidades suficientes para publicar la aplicación en medios convencionales como “Google Play Store” o crear instalables para poder compartirlo mediante otras plataformas. En nuestro caso, creamos una release en Github con el instalable de la aplicación para que cualquiera pueda usarla en su dispositivo.

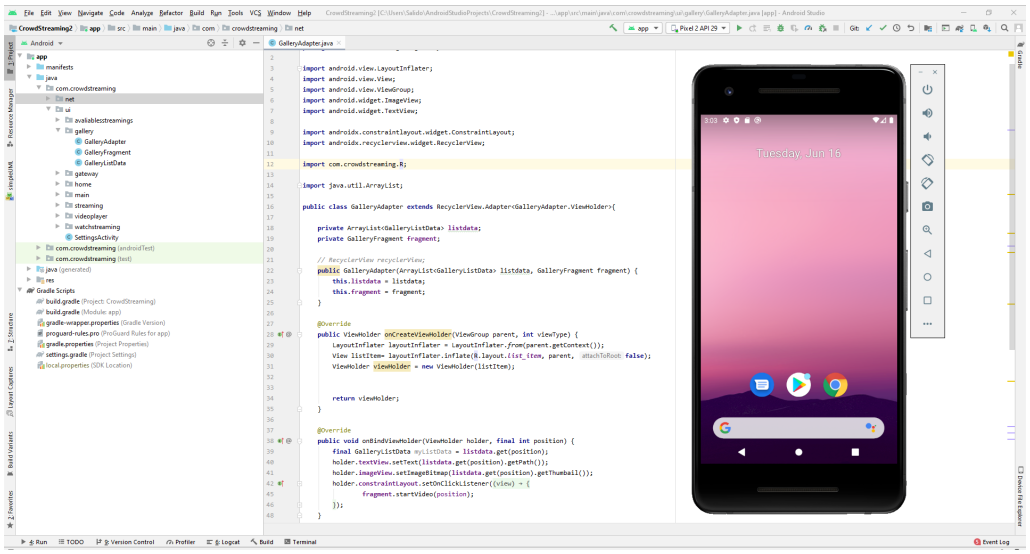


Figura 6.4: Ejemplo de Android Studio ejecutando el emulador

## 6.5. Desarrollo de prototipo

Puesto que la tecnología sobre la que vamos a desarrollar no solo es nueva para nosotros, si no para la mayoría de desarrolladores, existen muy pocos ejemplos sobre los que poder basarse a la hora de codificar. Para ello, siguiendo dos proyectos de Github [20] [52], desarrollamos un prototipo propio que nos permitía identificar si el dispositivo que está usando la aplicación dispone de Wifi Aware, además de conectar dos dispositivos e intercambiar mensajes de texto entre ellos.

Durante esta fase de desarrollo, nos dimos cuenta de varias peculiaridades. La primera, identificamos que en alguno de nuestros teléfonos personales ya disponían de Wifi Aware, pese a que en su especificación no lo contemplaba. Esto se debía a que los fabricantes habían construido los dispositivos con el hardware necesario para Wifi Aware pero el software no lo tenían desarrollado, lo que al uso lo hacia inservible.

Por otro lado, confirmamos una de nuestras principales hipótesis, ya que los dispositivos eran capaces de conectarse entre si sin la necesidad de una confirmación explicita del usuario. Esto daba lugar a que a partir de este punto descartáramos la opción de contingencia que teníamos en caso de que esto no se pudiera. Además este prototipo nos sirvió para descartar la primera historia de usuario (Permitir que usuario active/desactive Wifi Aware desde la aplicación) ya que la propia activación está implícita en la activación del Wifi del dispositivo.

En su lugar buscamos una alternativa con la cual el usuario pudiera activar/desactivar el Wifi del dispositivo desde la aplicación, pero el funcionamiento de Wifi Aware no permite esto ya que, para el funcionamiento de este, es necesario crear una sesión al inicio de la aplicación, la cual no puede volver a crearse a no ser que reinicies la aplicación. Por lo tanto, también descartamos la opción de activación/desactivación del Wifi desde la aplicación, obligando al usuario tener activado el Wifi antes de iniciar la aplicación. Si este no lo tuviera activado al abrir la aplicación se le avisaría.

Sobre los permisos de Android, también descubrimos que había algunos necesarios para el funcionamiento de Wifi Aware. Entre estos se encuentran el uso de la ubicación el cual permite que la aplicación descubra los dispositivos disponibles más cercanos.

Otra característica importante de tener en cuenta en Wifi Aware es que este dispone de dos roles por defecto, los cuales se les puede asignar al dispositivo. Estos roles, al igual que los descritos en nuestra aplicación, son “Publisher” y “Subscriber”. Entre estos dos roles existe la restricción por la cual, solo podrás encontrar y comunicarte con dispositivos que se identifiquen con el rol contrario al tuyo. A la hora de comunicarse, ambos lo podrán hacer bidireccionalmente, siempre y cuando

se cumpla la restricción mencionada.

En la siguiente figura se muestra la interfaz del prototipo. Este funciona de forma muy sencilla, cuando un usuario pulsa el botón "Publisher", su rol en Wifi Aware pasará a ser este y empezará a estar disponible para que otros "Subscriber" puedan comunicarse con él. Si este encuentra un Subscriber, se conecta a él automáticamente, y se les avisa a ambos dispositivos de esta conexión. En cualquiera de los lados, podían realizar un envío de un mensaje, el cual se mostrará en el área de texto de la parte de abajo de la interfaz en ambos dispositivos.

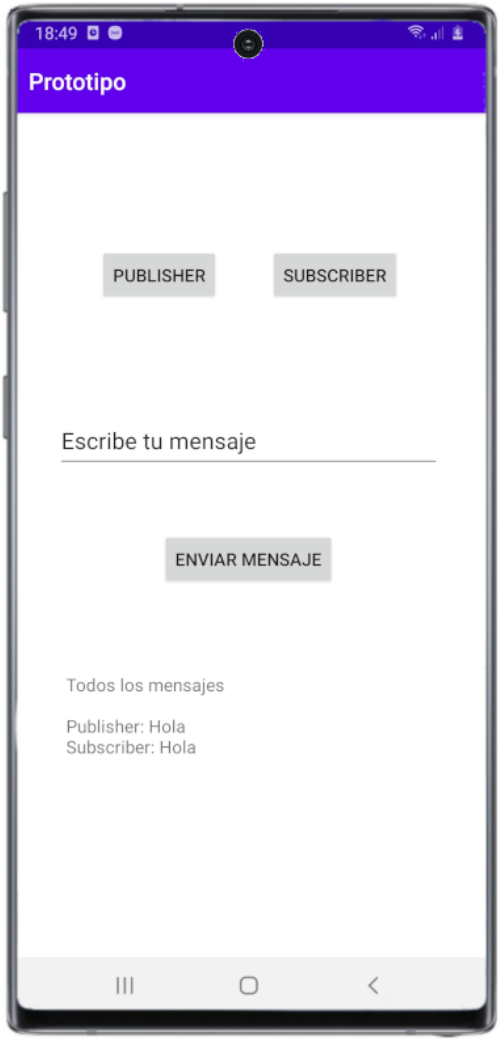



Figura 6.5: Prototipo Wifi Aware

## 6.6. Implementación de un nodo simple sobre Wifi Aware

En esta sección se explicará la implementación de las historias de usuario 2, 3 y 4 las cuales recogen todas la funcionalidad básica para realizar una emisión de un dispositivo a otro sin nodos intermedios.

Una vez detectadas las características y restricciones de Wifi Aware en Android, empezamos haciendo un diseño de clases con el cual llevar a cabo la implementación de los roles "Publisher" y "Subscriber". El Publisher será el encargado de transmitir el flujo de vídeo captado por su cámara y el flujo de audio por su micrófono y por otro lado, el Subscriber de visualizar dicho *stream* .

La primera opción que sopesamos para implementar esta funcionalidad fue el uso de la librería libstreaming, al igual que nuestros predecesores del curso pasado. Lo primero que hicimos fue investigar sobre su funcionamiento para poder extrapolarlo a Wifi Aware.

En primer instancia pensamos que esto sería factible ya que se necesitarían pocos cambios, pero una vez empezada la implementación comprobamos que esto no era posible. Cuando instanciábamos algunos elementos básicos para la creación del servidor RTSP con la clase de la librería RTSPClient, nos dimos cuenta de que usaban direcciones ipv4 al contrario que Wifi Aware que utiliza ipv6. Intentamos realizar los cambios pertinentes para adaptar el cambio de formato de la ip. Dicha clase utilizaba las clases nativas de Java "Socket", las cuales Wifi Aware también proporcionaba a la hora de conectarse. Supusimos que sería una buena idea modificar la clase "RTSPClient", de tal manera que en vez de crear la clase sus propios *sockets*, les proporcionaríamos los de Wifi Aware. Por alguna razón desconocida, los *sockets* de Wifi Aware no permitían la comunicación si se usaba bajo esta clase.

Por estas razones, investigamos más profundamente el código de libstreaming y extrapolamos las ideas que pudieran hacer posible el funcionamiento de nuestra aplicación. Fijándonos sobretodo, en la conexión de los *sockets* con la API de la cámara en Android, y la reproducción de dicho flujo a través del otro extremo del *socket*.

Esto nos hizo reflexionar sobre los protocolos que decidimos utilizar. Descubrimos que Wifi Aware proporciona una utilidad a la hora de realizar la conexión, por la cual, se decide el protocolo que se llevará acabo para la comunicación. Uno de estos era el protocolo SCTP [48], el cual vimos que se adaptaba a las necesidades de la aplicación, garantizando velocidad y seguridad. Aun así a nivel de aplicación no dispondremos de un protocolo de control, por lo que sopesamos implementar noso-

tros mismos este pero, debido a la complejidad no nos daría lugar a implementar la funcionalidades esenciales de la aplicación. En conclusión, decidimos aparcar la implementación del protocolo de control y centrarnos en maximizar el valor de la aplicación.

### 6.6.1. Diseño y codificación

Tras el desarrollo del prototipo y la investigación previa sobre esta tecnología, ya teníamos claro las clases necesarias a implementar para conseguir comunicar varios dispositivos mediante esta tecnología.

Wifi Aware nos proporciona una implementación de sesión la cual permite identificarnos unívocamente en la red. Esta tecnología obliga a que esta sesión sea creada en el momento en el que se abre la aplicación, ya que si se hace posteriormente no funcionará.

Para llevar a cabo esta funcionalidad, la implementación de Wifi Aware en Android proporciona una interfaz denominada “DiscoverySessionCallback”. Implementando esta interfaz, podemos establecer los diferentes comportamientos que puedan tener tanto el Publisher como el Subscriber. Debido a que alguna de sus funcionalidades son compartidas, como puede ser el establecimiento de la conexión, decidimos crear una clase abstracta la cual agregará las funcionalidades comunes denominada ”OwnDiscoverySessionCallback”.

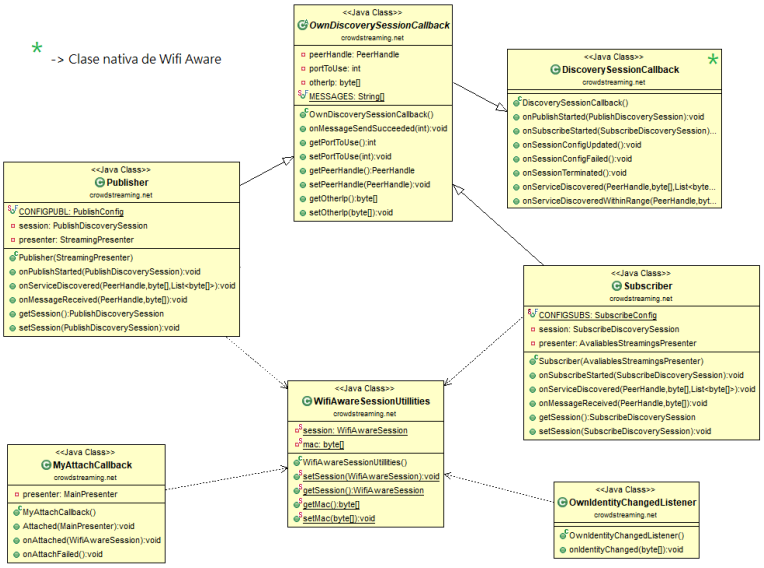



Figura 6.6: Diagrama de las clases relacionadas con Wifi Aware

Esta clase se encarga de guardar la información de las conexiones, tales como las IPs de los dispositivos conectados al nuestro además del puerto usado para el envío de datos. A la hora de comunicarnos, Wifi Aware nos proporciona un identificador llamado "PeerHandle", el cual es utilizado para identificar los dispositivos y mandar mensajes entre estos. Para el establecimiento de la conexión, hemos decidido crear una clase abstracta llamada "AbstractConnection" la cual extendemos según el rol del dispositivo. Esta clase también estará almacenada en la sesión.

Para establecer la conexión tuvimos que diferenciar el comportamiento del Publisher del comportamiento del Subscriber. Para ello creamos las clases "PublisherConnection" y "SubscriberConnection" las cuales extienden de "AbstractConnection". Esta última contiene la funcionalidad compartida de ambas clases y es la encargada de proporcionar los *sockets* por los cuales se envía el flujo de vídeo y audio. El Publisher establece el puerto y el protocolo y el Subscriber se conecta a este.

La implementación de Wifi Aware en Android también nos proporciona una clase abstracta llamada "NetworkCallback", la cual nos permite saber de manera explicita, cuando un dispositivo ha perdido la conexión con nosotros o ha cambiado alguna característica de la conexión. Este tendrá almacenada la conexión para notificar si pasa cualquiera de estos cambios y actuar en consecuencia.

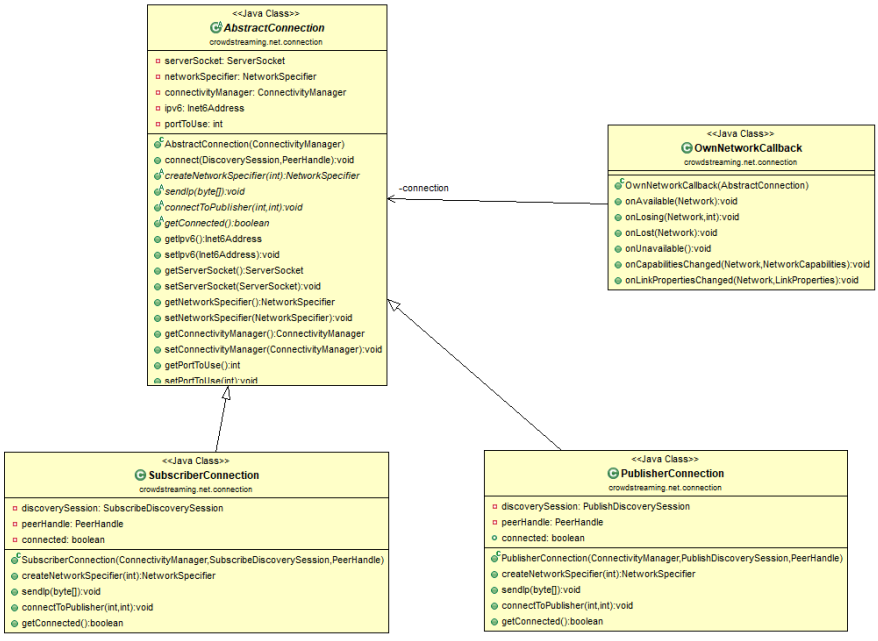



Figura 6.7: Diagrama de las clases de conexión

En cuanto a la interfaz gráfica, dispondremos de una “MainActivity”, en la cual se creara la sesión comentada anteriormente, donde se encuentra un menú desplegable y la lógica de intercambio de vistas. La vista que nos proporcionara la funcionalidad correspondiente al Publisher es “StreamActivity” en la cual se podrá previsualizar lo que está grabando la cámara además de encargarse del envió del contenido multimedia al Subscriber.

Para este *stream*, puesto que no estamos usando ningún protocolo de control de flujo de vídeo, esta vista se encarga de crear un *socket* utilizando la IP y el puerto proporcionados por la clase “Publisher”. Una vez hecho esto, la vista utiliza este *socket* para enviar el contenido obtenido mediante la cámara. Para esto usamos la API nativa de la cámara de Android

Por otro lado tenemos ”AvaliablesStreamingsFragment”, la cual nos muestra todas los *streams* cercanos disponibles para visualizar. Finalmente, cuando un usuario selecciona uno de estos, se le redireccionará a la vista “WatchStreamingActivity” la cual es la encargada de visualizar el contenido multimedia que está recibiendo.

En esta vista usamos la librería mencionada anteriormente, libVLC, la cual nos permite reproducir cualquier tipo de flujo multimedia mediante una URI. No obstante, dado que no estamos usando ningún protocolo de control de sesiones de vídeo en nuestra implementación, implementamos un buffer a través de un proxy HTTP local, el cual nos permitía transformar el flujo del *socket* a un flujo legible para la librería libVLC. Aunque esto provoca un pequeño retraso en el *stream*, garantiza que no se produzcan cortes.

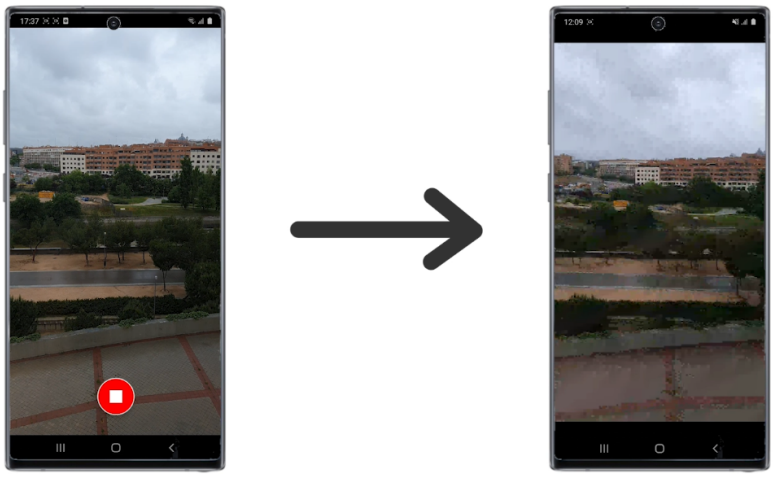


Figura 6.8: Ejemplo de *streams*

Finalmente, puesto que la aplicación hace uso de funcionalidades de Android

las cuales requieren permisos explícitos por parte de los usuarios, estos se pedirán una vez se abre la aplicación y en caso de que el usuario no los conceda, se le mostrará un mensaje de que no podrá usar la aplicación correctamente. Los permisos necesarios, además de la ubicación, son la cámara y micrófono a la hora de retransmitir, y el almacenamiento para la implementación del proxy y mas adelante para guardar los *streams* en el dispositivo del usuario.

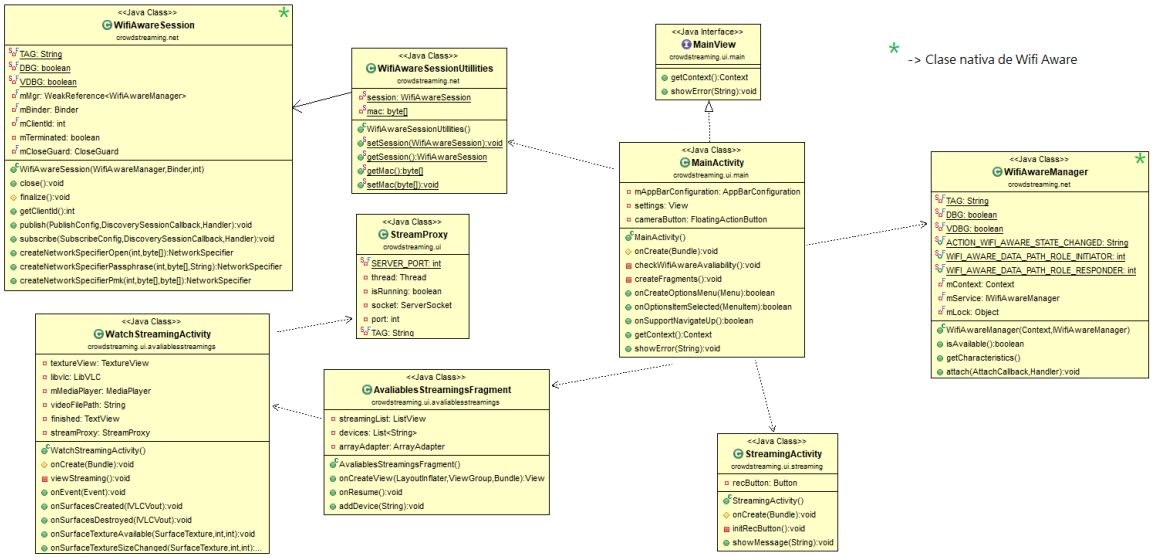



Figura 6.9: Diagrama de clases de las vistas

## 6.7. Almacenamiento de los *streams*

Debido a que repartimos los dispositivos entre los integrantes del equipo antes de la situación del estado de alarma causado por el COVID-19, ninguno de los miembros tenia la capacidad de probar las funcionalidades referentes a los nodos completos. Esto hizo que tuviéramos que avanzar en otras características menos relevantes mientras buscábamos una solución.

La funcionalidad por la que empezamos fue el almacenamiento de los *streams*, debido a que pensábamos que dentro de las características que podíamos hacer, era la que mas valor aportaba al proyecto.

### 6.7.1. Diseño y codificación

Para implementar esta funcionalidad, llevamos a cabo varios cambios en la clase "Publisher" la cual se encargaba de emitir el flujo de vídeo y audio y apenas cambios en la clase "Subscriber" encargada de la parte de recepción del flujo de vídeo y audio.

Esta característica se podrá activar o desactivar desde el menú de ajustes, ya que puede no interesar almacenar el vídeo, pudiendo causar problemas de seguridad en el escenario Witness.

Dado que el Subscriber ya tenia implementado un buffer a través del proxy HTTP local, el cual iba almacenando los datos recibidos para poder realizar el *stream* con mayor seguridad, lo más sencillo era usar este mismo para almacenar estos datos en un archivo de vídeo. Si el usuario elige la opción de no guardar el vídeo, este se guardara en una caché temporal la cual no puede ser accedida por el usuario de forma trivial.

Por otro lado, en la parte del Publisher, tuvimos que realizar cambios mas significativos. Dado que en la primera parte de la implementación, su dispositivo no comenzaba a enviar datos hasta que se conectaba al menos un usuario, por lo tanto, no disponía del flujo de datos desde el principio.

En esta segunda versión implementamos dos hebras, una grababa el vídeo de la cámara desde el principio, y la otra comenzaba el *stream* una vez le llegaba la petición por parte del Subscriber. Durante las primeras pruebas no conseguimos retransmitir vídeo pero si audio, tras investigarlo durante un tiempo, nos dimos cuenta de que no mandábamos la cabecera del fichero con los metadatos necesarios para su correcta reproducción. Para solucionar este problema, analizamos los metadatos de los vídeos creados y descubrimos que tenían una cabecera común dado el formato del vídeo (MPEG-2). Esta cabecera se le envía al Subscriber una vez se inicia el *stream* para su correcta reproducción.

## 6.8. Nodo Gateway

Para implementar esta funcionalidad, como comentábamos en la sección 4.1.9, decidimos usar la biblioteca de código abierto NetCipher. Lo primero que probamos, fue intentar acceder a una pagina de la red TOR desde la propia aplicación. Nos dimos cuenta de que, para que esto funcionara, el usuario necesitaría tener instalada la aplicación orbot [50], la cual es una VPN que te permite navegar por dicha red.

Una vez descubrimos esta limitación, decidimos explorar otras opciones menos restrictivas. Vimos mas sencillo que el usuario se instalara la aplicación "TOR brow-

ser" [49], la cual te permite navegar por la red TOR. Aparte de esto, la aplicación le mostraría una lista de organismos que disponían del repositorio "SecureDrop" el cual obtuvimos mediante la API oficial de "SecureDrop" [51].

### 6.8.1. Diseño y codificación

En la siguiente figura se muestra el diseño de las clases necesarias para llevar a cabo esta funcionalidad. Para ello disponemos de la clase "GatewayListData", la cual almacena la información de cada uno de los organismos que disponen de un respositorio SecureDrop, conteniendo su enlace TOR, título y logo. Por otro lado tenemos el "GatewayFragment", el cual tiene una lista de organismos, el cual se encarga de mostrarlos a través de la clase "GatewayAdapter".

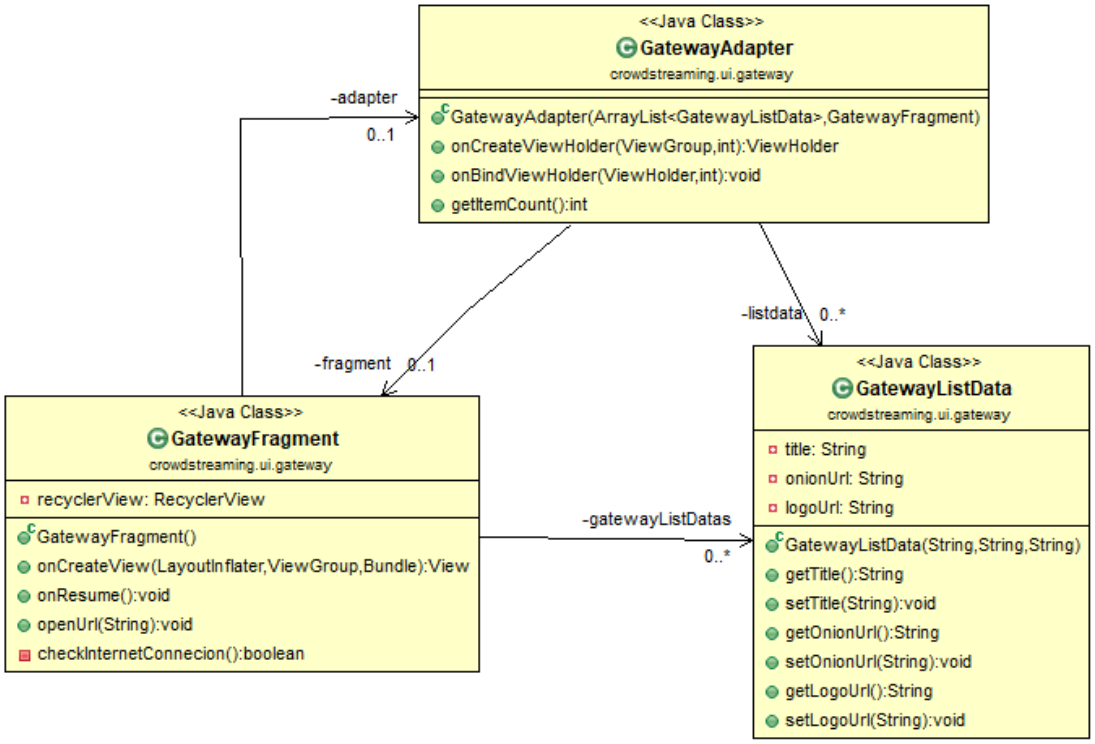



Figura 6.10: Diagrama de clases del nodo gateway

A continuación mostramos el resultado de la interfaz gráfica para esta funcionalidad, donde se puede ver la lista de organismos


Figura 6.11: Interfaz gráfica nodo gateway

## 6.9. Galería de *streams*

Dado que tras haber realizado las funcionalidades anteriores seguíamos sin disponer de al menos 3 dispositivos móviles, decidimos implementar una característica extra la cual permitiera al usuario disponer de una galería. En esta se puede seleccionar los *streams* almacenados anteriormente y visualizarlas.

### 6.9.1. Diseño y codificación

A continuación se muestra el diagrama de clases de la galería, el cual tiene una arquitectura similar al nodo gateway. La clase "GalleryFragment" dispone de una lista de fotos, en la cual se guarda la ruta de la imagen y su miniatura. Para conseguir la miniatura de los vídeos usamos la biblioteca de código abierto "FFmpegMediaMetadataRetriever".

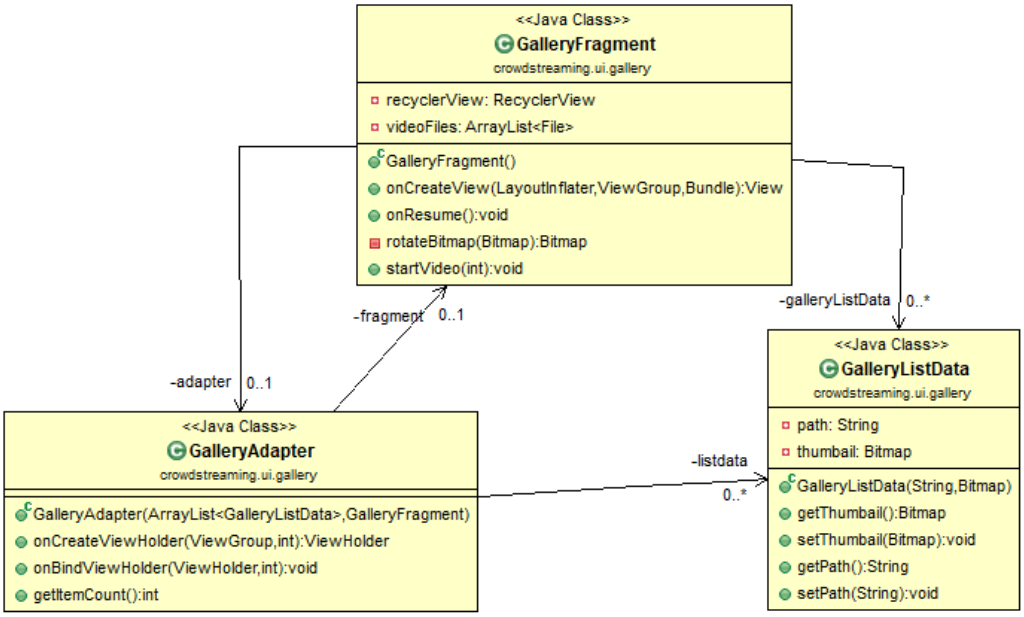



Figura 6.12: Diagrama de clases de la galería


Figura 6.13: Interfaz gráfica galería

## 6.10. Nodo tránsito simple oculto

Una vez disponíamos todos los dispositivos, seguimos la hoja de ruta planeada. Comenzamos con la implementación de un nodo de transito simple. En esta primera variante, el nodo de transito se conectará a al primer *stream* que encuentre, la cual no visualizará, pero compartirá con los nodos cercanos. También, existirá la opción de elegir el *stream* que comparte, en caso de no querer compartir la primera que encuentre.

Esta funcionalidad esta orientada la gente que simplemente quiere hacer de repetidor sin necesidad de visualizar el *stream*, pudiendo bloquear la pantalla del móvil o utilizar otras aplicaciones mientras comparte con otros dispositivos.

### 6.10.1. Diseño y codificación

Para implementar esta nueva funcionalidad, no nos hizo falta añadir ninguna vista nueva, ya que desde la propia vista de *streams* disponibles el usuario haría uso de esta.

En la primera variante una vez se inicia la vista, el dispositivo actuará como repetidor tomando uno de los *streams* cercanos, de manera aleatoria. Si este finaliza, empezará a compartir otro.

La segunda variante, permite al usuario elegir el *stream* que desea compartir. Si esta finaliza, se le mostrará la lista de *streams* cercanos para que así puede seleccionar alguno de los existentes.

Dado que la funcionalidad tanto del Subscriber como del Publisher estaba implementada anteriormente, la lógica a añadir en este nuevo nodo no era de gran complejidad, ya que un nodo tránsito implementa ambas funcionalidades al mismo tiempo. Esto solo supondría unos cambios a la hora de seleccionar el *stream* y que este no se reproduzca, por lo que la totalidad de estos cambios fueron en clases ya implementadas sin la necesidad de añadir clases nuevas.

Uno de los problemas que se nos planteo a la hora de implementar esta funcionalidad fue la posible existencia de ciclos en el proceso de compartir *streams*, es decir, que a un nodo le llegue un vídeo que ya está compartiendo. La funcionalidad implementada actualmente solo permite compartir un *stream* por lo que el problema de los bucles no se iba a dar en ningún caso. No obstante esta es una restricción que se debe tener en cuenta en posibles ampliaciones de este proyecto.

El año pasado los alumnos del TFG device to device streaming, predecesor a es-

te, se encontraron con problemas a la hora de realizar esto con Wifi Direct, ya que el único nodo que puede tener más de una conexión abierta era el Group Owner, único capaz de actuar como nodo transito. Esto conllevaba grandes limitaciones a la hora de usar la aplicación, reduciendo considerablemente su capacidad. Con Wifi Aware, hemos podido solventar estos problemas de manera relativamente fácil y accesible para el usuario.

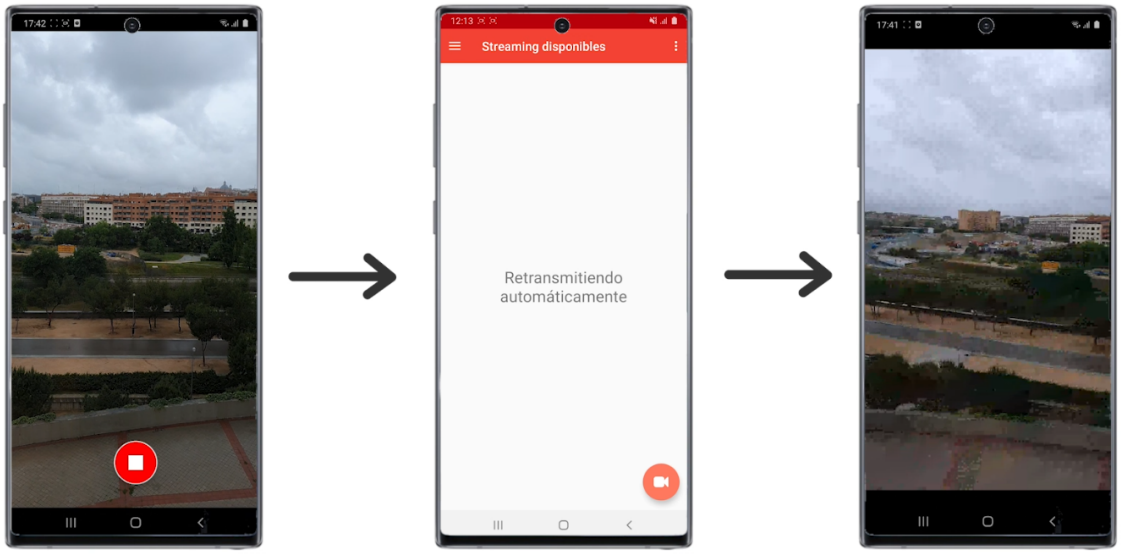



Figura 6.14: Ejemplo de *stream* con un nodo tránsito oculto

## 6.11. Nodo tránsito simple

### 6.11.1. Diseño y codificación

Al igual que la funcionalidad anterior, al estar implementado todo lo necesario para llevar a cabo esta, solo requería unos pequeños cambios para juntar estas funcionalidades. No hubo que añadir vistas ni clases nuevas, la lógica se implementaría desde la vista de la visualización.

Figura 6.15: Ejemplo de *stream* con un nodo tránsito simple

## 6.12. Publisher completo

Debido a la tardanza de la entrega de los dispositivos y a la crisis del COVID-19, esta fue la ultima funcionalidad que pudimos implementar. Con esta, conseguimos que un dispositivo retransmitiera a varios dispositivos adyacentes simultáneamente, la cual también fue una funcionalidad que ocasionó problemas a los alumnos del año pasado. Esta se ha podido implementar de una forma mas accesible para el usuario y sencilla a la hora de programar gracias a Wifi Aware.

### 6.12.1. Diseño y codificación

Para poder tratar la implementación llevada a cabo primero, debemos explicar mas detalladamente una de las características fundamentales de Wifi Aware en Android, llamada "PeerHandle". "PeerHandle" es una clase que nos proporciona la implementación de Wifi Aware en Android, la cual permite identificar un dispositivo de forma univoca, añadido a la IP o a la MAC del dispositivo. Este identificador es proporcionado en forma de objeto una vez se descubre un dispositivo. Por ello, cuando retransmitimos de un dispositivo a otro, para realizar la conexión y el intercambio de mensajes, debemos usar este.

En nuestra primera versión solamente almacenábamos el "PeerHandle" que estaba conectado actualmente a nuestro *stream*, en esta nueva versión, es necesario almacenar los "PeerHandle" de todos los dispositivos que se conectan a nuestro *stream*.

De la misma manera que en las dos funcionalidades anteriores, el cambio no fue muy significativo, ya que a la hora de retransmitir en vez de hacerlo por un solo *socket*, lo tendríamos que hacer por múltiples *sockets*, por lo tanto, el cambio llevado a cabo, fue almacenar estos en una lista. A la hora de retransmitir, recorríamos esta lista y mandábamos la información por los *sockets*.

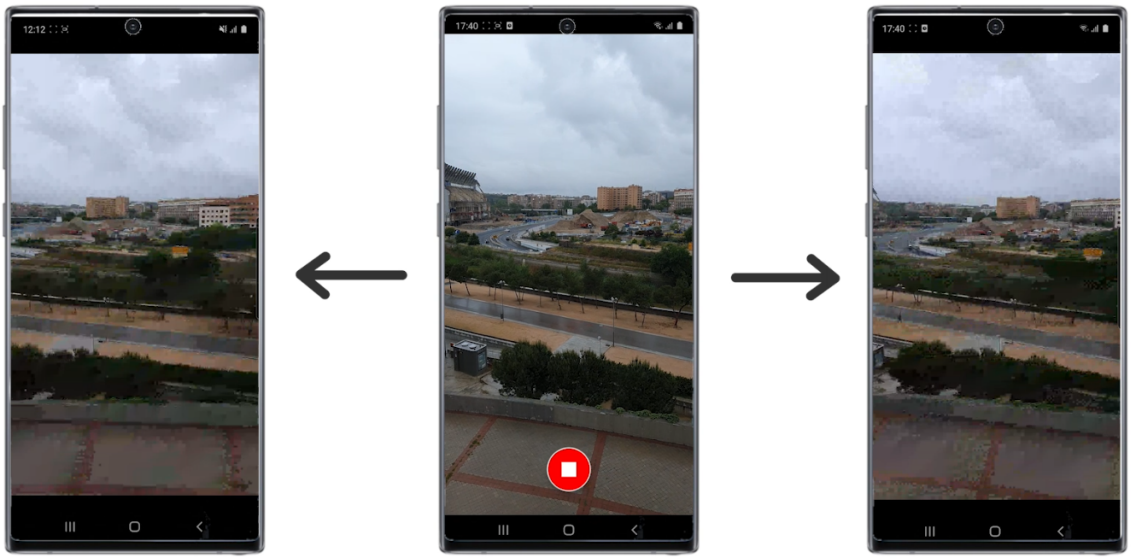


Figura 6.16: Ejemplo de *stream* de un Publisher completo

# 7. Conclusiones y trabajo futuro

La idea de este proyecto era implementar la transmisión de flujos de video y audio por una red ad-hoc de dispositivos móviles (habitualmente llamada MANET), partiendo de los resultado obtenidos en un TFG del año pasado "Device to device streaming en dispositivos móviles" [3]. Como trabajo futuro, propusieron hacer un red sin infraestructura construida sobre Wifi Aware por la cual poder transmitir vídeo y audio.

Llevar a cabo este propósito al completo es una idea muy ambiciosa, nosotros por nuestra parte, hemos conseguido implementar una primera versión de esta con la funcionalidad suficiente para demostrar su viabilidad. No solo eso, si no que también hemos logrado poder solventar varios problemas que tuvieron en el proyecto predecesor a este.

Uno de los mayores desafíos del año pasado como hemos explicado en la sección 3.2.1, era la implementación de un nodo de tránsito de una manera sencilla y sin demasiadas restricciones. Este problema no lo consiguieron solventar debido a que la implementación de Wifi Direct en Android, no tenia implementadas las partes optativas de la especificación de Wifi Direct que se necesitaban para hacer esto posible.

Wifi Aware, al suponer una mejora con respecto a Wifi Direct, nos ha dado la funcionalidad suficiente para poder solventar estos problemas de una manera relativamente sencilla. No obstante, aun suponiendo una mejora, también nos hemos encontrado una serie de desafíos.

El primer problema ha sido la poca documentación sobre como desarrollar con esta tecnología, la cual aún estando bien especificada, no era del todo suficiente. Este problema se pudo solventar gracias a los pocos proyectos en Github que se podían encontrar sobre el tema, los cuales sirvieron como base inicial sobre la que construir nuestra aplicación.

Otro gran problema que nos ha surgido también por usar Wifi Aware, ha sido la tardanza a la hora de conseguir dispositivos compatibles con esta tecnología, lo cual nos retrasó hasta marzo para poder empezar a codificar. Este problema, junto al estado de alarma provocado por el COVID-19, ha provocado que tuviéramos que prescindir de algunos requisitos y centrarnos en los que daban mas valor a la aplicación.

Por estos motivos, sumados al hecho de no haber podido modificar la librería "libstreaming", la cual implementa un protocolo de control sobre el flujo de vídeo, tuvimos que prescindir de este mecanismo. Esta librería sirvió como base a los alumnos del proyecto anterior, sin el cual su desarrollo hubiese sido mucho mas complicado. La

falta de un protocolo de control, puede conllevar problemas de seguridad y de pérdida de paquetes, quedando esto como tarea pendiente para la continuación de este proyecto en un futuro.

Debido a la falta de tiempo, tampoco se pudieron implementar las restricciones referidas al escenario Witness reflejadas en la sección 1.3, las cuales se centraban en el anonimato y la seguridad de los usuarios. También quedaría como pendiente la restricción del escenario Humanitarian, por la cual se daba la opción de difuminar las caras de las personas en las emisiones.

Por otro lado, durante el desarrollo, se nos ocurrió una funcionalidad que puede tener bastante interés para el futuro, en la cual el nodo tránsito hace de repetidor múltiple. Este nodo no tendría ninguna restricción en cuanto al número de emisiones compartidas simultáneamente, es decir, en vez de seleccionar la primera emisión que encuentre, compartirá todas las encontradas. Esta funcionalidad puede conllevar nuevos problemas como el problema de ciclos, en el que a un nodo, le puede llegar una emisión que ya está compartiendo, dejando pendiente también la investigación sobre como solventar este problema. Además, también haría faltar estudiar los posibles problemas de rendimiento en la red en este caso, ya que un repetidor podría estar sometido a una gran carga de flujos de datos.

Otras tareas pendientes que no han sido especificadas en los requisitos, pero que podrían mejorar la aplicación en su estado actual serían: poder poner un título a tu emisión, poder elegir con que cámara transmitir el vídeo (frontal y trasera), poder ver el numero de personas que están visualizando tu emisión y tener cifrada la emisión.

Aun así, aunque falte funcionalidad por desarrollar, hemos conseguido una aplicación intuitiva y útil que demuestra la viabilidad del concepto propuesto en este trabajo fin grado, mejorando la aplicación del proyecto predecesor a este. También, hemos conseguido dejar una buena base inicial, habiendo realizado un gran trabajo de investigación sobre el que asentar una continuación a este proyecto.

# 8. Conclusions and future work

The idea of this project was to implement the streaming of video and audio over an ad-hoc network of mobile devices (usually called MANET), based on the results obtained in a TFG last year "Device to device streaming on mobile devices" [3]. As future work, they proposed to make a network without infrastructure built on Wifi Aware to broadcast video and audio

To carry out all this is a very ambitious idea, we have managed to implement a first version of this with enough functionality to demonstrate its viability. Not only that, but we have also been able to solve several problems they had in the previous project.

One of the biggest challenges encountered last year, as explained in the section 3.2.1', was the implementation of a transit node in a simple way and without too many restrictions. This problem was not solved because the implementation of Wifi Direct on Android did not have the optional parts of the Wifi Direct specification implemented that were needed to make this possible.

Wifi Aware, being an improvement over Wifi Direct, has given us enough functionality to be able to solve these problems in a simple way. However, while this is an improvement, we have also encountered a number of challenges.

The first problem has been the little documentation on how to develop with Wifi Aware, which although is well specified, was not enough. This problem could be solved thanks to the few projects in Github that could be found on the subject, which served as an starting point to build our application.

Another big problem that we found using Wifi Aware, has been the delay in getting devices compatible with this technology, which delayed us until March to start coding. This problem, together with the state of alarm caused by COVID-19, meant that we had to dispense with some requirements and focus on those that gave most value to the application.

Within this, we found the fact that we could not modify the library libstreaming to have a control protocol with which to manage the flow of video and audio. The lack of this protocol, can lead to security problems and loss of packages, leaving this as a pending task for the continuation of this project in the future.

Due to lack of time, the restrictions on the Witness scenario reflected in the section 1.3, which focused on anonymity and user security, could not be implemented either. The Humanitarian scenario restriction, whereby the option of blurring people's faces in the broadcasts would also remain pending.

On the other hand, during the development, we came up with a functionality that may be quite interesting for the future, in which the transit node acts as a multiple repeater. This node would not have any restriction on the number of shared emissions simultaneously, that is, instead of selecting the first emission it finds, it will share all the streams found. This functionality may lead to new problems such as the cycle problem, in which a node may receive an emission that is already shared, leaving pending also the investigation on how to solve this problem. In addition, it would also be necessary to study the possible performance problems in the network in this case, since a repeater could be subject to a big load of data flows.

Other pending tasks that have not been specified in the requirements, but that could improve the application in its current state would be: to be able to title your broadcast, to be able to choose with which camera to transmit the video (front and back), to be able to see the number of people who are watching your broadcast and to have the broadcast encrypted.

Even so, although there is still some functionality to be developed, we have achieved an intuitive and useful application that demonstrates the viability of the concept proposed in this project, improving the application of the predecessor project to this one. Also, we have managed to leave a good initial basis, having done a great deal of research on which to base a continuation of this project.

# 9. Aportación de cada participante

Puesto que íbamos a trabajar con varias tecnologías y herramientas con las que no habíamos tratado previamente, decidimos tener una primera fase de investigación. Durante esta, nos reuníamos varias veces por semana y tratábamos entre todos, los diferentes enfoques que encontrábamos sobre estas tecnologías.

Una vez realizado todo este primer proceso de investigación y al no disponer todavía de los dispositivos, nos centramos en realizar la especificación de requisitos. Durante este proceso, al igual que en el anterior, nos reuníamos periódicamente para discutir las diferentes funcionalidades que queríamos para la aplicación.

Finalmente, a la hora de desarrollar, puesto que el desarrollo se veía limitado al número de dispositivos que teníamos para realizar pruebas, además del confinamiento producido por el estado de alarma, nos reuníamos semanalmente mediante vídeo conferencias y usando Teamviewer para compartir el código que se estaba programando. De esta manera, discutíamos las posibles implementaciones y el diseño de la aplicación, para así, posteriormente, mientras una persona programaba, proponer diferentes soluciones para solventar los diferentes conflictos que se iban dando.

Como conclusión, se podría decir que todos los integrantes de este proyecto, hemos estado involucrados por igual en el desarrollo de este. Tomamos esta decisión como la más apropiada, debido a la dificultad del proyecto y a la situación excepcional del estado de alarma.

A continuación, mostramos en detalle un desglose de las tareas de cada participante, como se puede ver, cada integrante ha tenido las mismas tareas.

## 9.1. Francisco Calero Velasco

### 9.1.1. Investigación

- Estudio de herramientas y lenguajes para aplicaciones Android: Android Studio, Java, XML y Apis de Android.
- Estudio de tecnologías para realizar conexiones peer-to-peer: WiFi Direct, Wi-Fi Aware, Apple Wireless Direct Link, Bluetooth...
- Investigación sobre aplicaciones parecidas: chitchat, shareIt, serval, MANET voice Chat, GoTenna...
- Investigación sobre las restricciones de seguridad según el escenario de la aplicación.
- Estudio de tecnologías para la retransmisión multimedia: HTTP, RTSP, RTP, UDP, TCP...
- Selección de tecnologías tales como las librerías y APIs disponibles.

### 9.1.2. Especificación y desarrollo

- Especificación a alto nivel.
- Especificación detallada de las historias de usuario.
- Diseño diagramas UML.
- Diseño Mockups.
- Desarrollo del prototipo.
- Desarrollo de la aplicación.

### 9.1.3. Memoria

- Resumen.
- Introducción.
- Estado del arte y antecedentes.
- Elección de tecnologías.
- Especificación de requisitos.

- Desarrollo.
- Conclusiones y trabajo futuro.
- Traducción de la introducción y conclusión al inglés.

## 9.2. Sergio Manzanaro Caraballo

### 9.2.1. Investigación

- Estudio de herramientas y lenguajes para aplicaciones Android: Android Studio, Java, XML y Apis de Android.
- Estudio de tecnologías para realizar conexiones peer-to-peer: WiFi Direct, Wi-Fi Aware, Apple Wireless Direct Link, Bluetooth...
- Investigación sobre aplicaciones parecidas: chitchat, shareIt, serval, MANET voice Chat, GoTenna...
- Investigación sobre las restricciones de seguridad según el escenario de la aplicación.
- Estudio de tecnologías para la retransmisión multimedia: HTTP, RTSP, RTP, UDP, TCP...
- Selección de tecnologías tales como las librerías y APIs disponibles.

### 9.2.2. Especificación y desarrollo

- Especificación a alto nivel.
- Especificación detallada de las historias de usuario.
- Diseño diagramas UML.
- Diseño Mockups.
- Desarrollo del prototipo.
- Desarrollo de la aplicación.

### 9.2.3. Memoria

- Resumen.
- Introducción.
- Estado del arte y antecedentes.
- Elección de tecnologías.
- Especificación de requisitos.
- Desarrollo.
- Conclusiones y trabajo futuro.
- Traducción de la introducción y conclusión al inglés.

## 9.3. David Salido Camacho

### 9.3.1. Investigación

- Estudio de herramientas y lenguajes para aplicaciones Android: Android Studio, Java, XML y Apis de Android.
- Estudio de tecnologías para realizar conexiones peer-to-peer: WiFi Direct, Wi-Fi Aware, Apple Wireless Direct Link, Bluetooth...
- Investigación sobre aplicaciones parecidas: chitchat, shareIt, serval, MANET voice Chat, GoTenna...
- Investigación sobre las restricciones de seguridad según el escenario de la aplicación.
- Estudio de tecnologías para la retransmisión multimedia: HTTP, RTSP, RTP, UDP, TCP...
- Selección de tecnologías tales como las librerías y APIs disponibles.

### 9.3.2. Especificación y desarrollo

- Especificación a alto nivel.
- Especificación detallada de las historias de usuario.
- Diseño diagramas UML.

- Diseño Mockups.
- Desarrollo del prototipo.
- Desarrollo de la aplicación.

### 9.3.3. Memoria

- Resumen.
- Introducción.
- Estado del arte y antecedentes.
- Elección de tecnologías.
- Especificación de requisitos.
- Desarrollo.
- Conclusiones y trabajo futuro.
- Traducción de la introducción y conclusión al inglés.

# 9. Bibliografía


[1] *"Shuttleworth Foundation"* http://www.shuttleworthfoundation.org/ , último acceso 18-06-2020.

[2] Marta Juste, *"Las cifras de Internet: En España el 85 % de la población está conectada."* https://www.expansion.com/economia-digital/innovacion/2018/02/01/5a72e73a22601db2288b4658.html, 01-02-2018, último acceso 18-06-2020.

[3] Ivan Gulyk y Noel José Algora Igual, *"Device to Device streaming en dispositivos móviles"* https://eprints.ucm.es/56536/, 09-03-2020, último acceso 18-06-2020.

[4] Jován Pulgarín, *"¿Qué es un deepfake y por qué deberíamos saberlo?"* https://www.cnet.com/es/noticias/que-es-deepfake-como-hacer-peligro-videos-manipulacion-apps/, 24-09-2019, último acceso 18-06-2020.

[5] Saurabh Upadhyay y Sanjay Kumar Singh, *"Video Authentication- An Overview"* http://airccse.org/journal/ijcses/papers/1111ijcses06.pdf, 4-11-2011, último acceso 18-06-2020.

[6] Wikipedia, *"Group Domain of Interpretation"* https://es.wikipedia.org/wiki/Group$_Domain_of_Interpretation$, último acceso 18-06-2020.

[7] softwarelab, *"¿Qué es el Bluetooth y para qué sirve?"* https://softwarelab.org/es/bluetooth/, último acceso 18-06-2020.

[8] Carlos González, *"¿Qué es el WiFi? Así funciona la tecnología que lo conecta todo a Internet"* https://www.adslzone.net/reportajes/tecnologia/que-es-wifi-como-funciona/, 19-06-2019, último acceso 18-06-2020.

[9] Esteban, *"Qué es el WiFi Direct, para qué sirve y cómo configurarlo en tu Android"* https://elandroidelibre.elespanol.com/2019/05/que-es-el-wifi-direct-para-que-sirve-y-como-configurarlo-en-tu-android.html, 25-05-2019, último acceso 18-06-2020.

[10] Borja Simancas Delgado, *"WiFi Aware vendrá en Android O"* https://elandroidelibre.elespanol.com/2017/07/wifi-aware-android-o-informacion.html, 29-07-2017, último acceso 18-06-2020.

[11] Milan Stute, David Kreitschmann y Matthias Hollick, *"Apple Wireless Direct Link"* https://arxiv.org/pdf/1808.03156.pdf, 9-8-2018, último acceso 18-06-2020.

[12] Rubén Andrés, *"Qué es NFC Móvil, cómo funciona y qué puedes hacer con él"* https://computerhoy.com/noticias/life/que-es-nfc-movil-como-funciona-que-puedes-hacer-24207, 08-02-2017, último acceso 18-06-2020.

[13] *"TCP (Transmission Control Protocol): retrato del protocolo de transporte"* https://www.ionos.es/digitalguide/servidores/know-how/que-es-tcp-transport-control-protocol/,02-03-20, último acceso 18-06-2020.

[14] *"UDP: ¿Qué es el protocolo UDP?"* https://www.ionos.es/digitalguide/servidores/know-how/udp-user-datagram-protocol/, 05-06-20, último acceso 18-06-2020.

[15] Fabian Avila *"¿Qué es y para que sirve el Streaming?"* https://eventovirtual.co/que-es-y-para-que-sirve-el-streaming/ , último acceso 18-06-2020.

[16] *"RTP"* https://tools.ietf.org/html/rfc3550 , último acceso 18-06-2020.

[17] *"RTCP"* https://tools.ietf.org/html/rfc4961 , último acceso 18-06-2020.

[18] Nestor Ángeles, *"RTSP"* https://tecnosinergia.zendesk.com/hc/es/community/posts/115000357991-Protocolo-RTSP-Para-que-sirve- , último acceso 18-06-2020.

[19] Rubén Velasco, *"ShareIt"* www.redeszone.net/2018/08/11/wi-fi-direct-aplicaciones-gratis-usar-protocolo/ , último acceso 18-06-2020.

[20] Karmakargopi, *"ChitChat"* https://github.com/karmakargopi/ChitChat , último acceso 18-06-2020.

[21] *"Serval project"* http://www.servalproject.org/ , último acceso 18-06-2020.

[22] *"Manet voice chat"* https://play.google.com/store/apps/details?id=org.span.ptthl=es , último acceso 18-06-2020.

[23] Ajay Kumar, *"GoTenna"* https://www.pcmag.com/reviews/gotenna , último acceso 18-06-2020.

[24] *"ObscuraCam"* https://guardianproject.info/apps/org.witness.sscphase1/ , último acceso 18-06-2020.

[25] *"ProofMode"* https://guardianproject.info/apps/org.witness.proofmode/ , último acceso 18-06-2020.

[26] *"Ripple"* https://guardianproject.info/apps/info.guardianproject.ripple/ , último acceso 18-06-2020.

[27] *"EyeWitness"* https://www.eyewitness.global/ , último acceso 18-06-2020.

[28] *"SvcAuth"* https://nsl.cs.sfu.ca/wiki/index.php/svcAuth , último acceso 18-06-2020.

[29] Yubal, *"Red tor"* https://www.xataka.com/basics/red-tor-que-como-funciona-como-se-usa , último acceso 18-06-2020.

[30] *"Twitter Api"* https://developer.twitter.com/en/docs , último acceso 18-06-2020.

[31] *"Netcipher"* https://guardianproject.info/es/code/netcipher/ , último acceso 18-06-2020.

[32] *"SecureDrop"* https://securedrop.org/ , último acceso 18-06-2020.

[33] *"Guardian Project"* https://guardianproject.info/es/ , último acceso 18-06-2020.

[34] *"Andorid"* https://www.android.com/intl/es_es/ , último acceso 18-06-2020.

[35] *"Andorid Studio"* https://developer.android.com/docs , último acceso 18-06-2020.

[36] *"Java"* https://www.java.com/es/download/faq/whatis_java.xml , último acceso 18-06-2020.

[37] *"LibVLC"* https://wiki.videolan.org/LibVLC/ , último acceso 18-06-2020.

[38] Fyhertz, *"Libstreaming"* https://github.com/fyhertz/libstreaming , último acceso 18-06-2020.

[39] *"Git"* https://git-scm.com/ , último acceso 18-06-2020.

[40] *"Github"* https://github.com/ , último acceso 18-06-2020.

[41] *"Trello"* https://trello.com/ , último acceso 18-06-2020.

[42] Marco Antonio Gomez Martín y Pedro Pablo Gomez Martín, *"Texis"* http://gaia.fdi.ucm.es/research/texis// , último acceso 18-06-2020.

[43] *"Latex"* https://www.latex-project.org/ , último acceso 18-06-2020.

[44] *"Overleaf"* https://www.overleaf.com/ , último acceso 18-06-2020.

[45] *"Qué es Google Drive"* http://gapps.upaep.mx/inicio/googledocs/google-drive/que-es-google-drive , último acceso 18-06-2020.

[46] *"Discord"* https://discord.com/ , último acceso 18-06-2020.

[47] Andrea Núñez, *"Qué es TeamViewer"* https://www.ticbeat.com/tecnologias/que-es-teamviewer-y-como-funciona/ , último acceso 18-06-2020.

[48] *"SCTP"* https://www.ionos.es/digitalguide/servidores/know-how/sctp/ , último acceso 18-06-2020.

[49] *"Tor browser"* https://play.google.com/store/apps/details?id=org.torproject.torbrowserhl=es , último acceso 18-06-2020.

[50] *"Orbot"* https://play.google.com/store/apps/details?id=org.torproject.androidhl=es , último acceso 18-06-2020.

[51] *"Api secureDrop"* https://securedrop.org/api/v1/directory/ , último acceso 18-06-2020.

[52] Eric *"NAN"* https://github.com/anagramrice/NAN , último acceso 18-06-2020.

[53] *"Documentación de Wifi Aware en Android:"* https://developer.android.com/guide/topics/connectivity/wifi-aware , último acceso 18-06-2020.

[54] *"Kanban"* https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban , último acceso 18-06-2020.

[55] Borja Simancas Delgado, *"Foto Wifi Aware"* https://elandroidelibre.elespanol.com/2017/07/wifi-aware-android-o-informacion.html , último acceso 18-06-2020.

[56] University of Zurich, *"Foto P2P streaming"* http://www.csg.uzh.ch/csg/en/publications/software/p2p-streaming.html?fontsize=big , último acceso 18-06-2020.

[57] *"Foto ejemplo obscuracam"* https://obscuracam.es.aptoide.com/app , último acceso 18-06-2020.

[58] *"Foto funcionamiento SecureDrop"* https://github.com/freedomofpress/securedrop/issues/274 , último acceso 18-06-2020.

[59] *"Foto ciclo activity"* https://aep-sdks.gitbook.io/docs/using-mobile-extensions/mobile-core/lifecycle/lifecycle-extension-in-android , último acceso 18-06-2020.

[60] *"Foto ciclo fragment"* https://developer.android.com/guide/components/fragments?hl=es , último acceso 18-06-2020.