Informe de Trabajo Profesional de Ingeniería en Informática

`svrmap`: *Identificación de estrategias de distribución de contenido en Internet*

Tutor: Dr. Ing. Esteban Carisimo

Alumnos

Rodrigo Zapico, *(Padrón # 93.272)*
`rzapico@fi.uba.ar`

Martín I. Errázquin, *(Padrón # 98.017)*
`merrazquin@fi.uba.ar`

Facultad de Ingeniería, Universidad de Buenos Aires

1 de junio de 2021

Índice general

Índice de figuras

Resumen

svrmap es una aplicación que permite a los usuarios descubrir el origen del contenido que consumen. Al utilizar una app como Spotify, Instagram o Mercado Pago, ¿Qué podemos hacer con nuestros teléfonos para saber hacia dónde viajan nuestros datos? Hasta ahora nada. Los usuarios carecen de recursos para conocer con qué plataformas interactúan sus Apps, y su impacto en la UX y la privacidad. Readaptando el concepto de VPNs, desarrollamos una App en Android que recolecta información de red para detectar los proveedores de contenido con los que interactúa cada una de las Apps. A través de esta App hemos podido observar (1) el extendido uso del paradigma de multi-CDNs, (2) las estrategias de distribución de contenido de medios periodísticos locales e internacionales (3) el rol de los ISPs en la distribución de contenido.

Capítulo 1

Introducción

1.1. Motivación

Ante la centralidad de los dispositivos móviles, y con un creciente ecosistema de aplicaciones móviles, este trabajo se enfoca en crear una herramienta que permita a los usuarios investigar **tres ejes fundamentales**: *(i)* el uso de infraestructuras de gran escala (cloud providers o CDNs) en la distribución del contenido móvil *(ii)* la creciente discusión por la privacidad de los usuarios al utilizar las aplicaciones *(iii)* la centralización de la infraestructura usada para distribuir el contenido.

En concreto, este trabajo plantea desarrollar una aplicación móvil que le permita al usuario, y también a la comunidad científica, explorar las interacciones del resto de sus aplicaciones con las infraestructuras que le brindan los servicios. A pesar que en este trabajo sólo se abordan una serie de análisis preliminares para comprobar el funcionamiento de la herramienta, su desarrollo está motivado en un marco de proyecto más grande. El objetivo final de este proyecto es que la aplicación habilite investigar las siguientes preguntas científicas, comprendidas en los tres ejes principales que estructuraron el desarrollo de la aplicación.

Primero, enfocándonos en el uso de infraestructuras de gran escala (cloud providers o CDNs) en la distribución del contenido móvil, svrmap permite determinar cuáles son las infraestructuras públicas (CDNs) más utilizadas por las aplicaciones móviles. Asimismo, ante el creciente uso del paradigma multi-CDN (ver Sección 1.2) esta herramienta posibilitará verificar la hipótesis que plantea que *el esquema multi-CDN prevalece en la distribución de contenido*. En caso de que esta hipótesis sea verificada, la aplicación permitiría estudiar si, dada una aplicación, el conjunto de CDNs utilizadas varía en función del ISP o la ubicación geográfica donde se encuentre el usuario. A su vez, la aplicación permitirá estudiar si la variación en el uso de CDNs responde a estados de la red, como puede ser la congestión o la distribución de cargas. También la fluctuación temporal del

uso de las CDNs por parte de una aplicación puede corresponderse con los costos asociados al uso de estas plataformas, permitiendo estudiar los aspectos comerciales detrás de la distribución del contenido. Finalmente, la aplicación también permitirá descubrir cuál es el rol de los ISPs alojando caches como política para reducir los costos del ISP y mejorar la experiencia de los usuarios.

Segundo, centrando nuestra atención en la creciente discusión por la privacidad de los usuarios al utilizar las aplicaciones, `svrmap` permitirá abordar esta discusión utilizando información recopilada del uso de las aplicaciones. Ciertos marcos normativos, como por ejemplo GDPR, establecen que el contenido de los usuarios debe ser servido desde infraestructuras físicamente alojadas dentro de la jurisdicción donde se encuentra el usuario. En estos casos, `svrmap` podría aportar información de las infraestructuras utilizadas para facilitar la verificación del cumplimiento del marco normativo. Más allá de los aspectos normativos, los grupos y usuarios interesados en la privacidad al momento de uso de las aplicaciones podrán identificar cuál es el paradero de sus datos personales al usar una aplicación.

Tercero, enfocándonos en la presencia de actores preponderantes en la oferta de servicios en la nube, `svrmap` permitirá estudiar temas vinculados a la concentración de este mercado. En primer lugar, como fue anteriormente mencionado, es importante contar con esta aplicación para poder determinar si realmente existe una concentración de las plataformas cloud. De ser así, `svrmap` podría identificar riesgos, como lo es el impacto de una falla en un esquema centralizado (paradigma único punto de falla), ante la diversificación del mercado. En el caso de una hipotética falla en estos puntos tan predominantes podría causarse una falla a gran escala de los servicios que corren sobre estas plataformas. De ser así, esto tendría grandes repercusiones económicas para los proveedores de servicios y también para las empresas que contratan estos servicios. Desde el punto de vista de la economía, la concentración de mercado también puede ser estudiada como la falta de competencia y su impacto en el costo de los servicios.

1.2. Contexto, paradigmas tecnológicos y estado del arte

Internet ha pasado a tener un papel central en la vida de las personas, donde el crecimiento de la población mundial conectada a la red se incrementó del 10 % al 51 % entre 2005 y 2020 [1]. Más aún, la irrupción de los teléfonos móviles inteligentes (smartphones) ha reconfigurado el acceso a la red. La aparición de esta tecnología ha permitido la portabilidad de la conexión a Internet y la capacidad de estar constantemente en línea. A

su vez, los smartphones han democratizado y ampliado el acceso a Internet permitiendo incorporar nuevas familias a la Internet en lugares donde antes no existía cobertura de redes de acceso fijas o en familias sin posibilidades de poder adquirir una computadora [2]. A la par de la rápida proliferación de smartphones, un numeroso ecosistema de aplicaciones ha surgido para satisfacer las demandas de los usuarios de teléfonos móviles [3]. Generalmente, estas aplicaciones se basan en el concepto de la disponibilidad permanente de acceso a Internet, y a partir de tal premisa, brindan el contenido, permiten accesos, o incluso introducen publicidades, como parte de la estrategia de monetización de las aplicaciones.

La arquitectura de los servicios brindados a través de Internet, independientemente del tipo de dispositivo que se utilice, se basa en estrategias de distribución de contenido a través de infraestructuras específicas, denominadas Redes de Distribución de Contenido (del inglés Content Delivery Networks, CDNs) o también conocidos como Cloud Providers. Las CDNs son sistemas distribuidos desde los cuales se brinda el contenido a los usuarios, diseñados para maximizar los parámetros de experiencia de usuario (latencia y throughput), minimizar la congestión (load balancing, multiple exit/peering points) y maximizar la resiliencia y disponibilidad (avoid single point of failure, protección ante DDoS) [4]. El diseño de las CDNs se centra en dos paradigmas, *enter deep* o *bring home* [5], los cuales tienen como objetivo esparcir estas infraestructuras en diferentes ubicaciones con la finalidad de brindar el contenido a los usuarios desde la proximidad. La premisa por la cual estas infraestructuras buscan ubicarse próximas a los usuarios es el hecho de poder reducir la latencia, la cual es un parámetro importante a la hora de evaluar la calidad de experiencia y la tasa de descarga [6]. En particular, la idea de aproximar estos sistemas busca reducir el tiempo de propagación en el medio, que depende exclusivamente de parámetros físicos como lo es la distancia. La escala de las CDNs más populares, por ejemplo Google o Akamai [4, 7, 8], comprende instancias del sistema distribuido a lo largo de ciudades en los cinco continentes.

El ascenso en el uso de CDNs para la distribución efectiva del contenido ha convertido a este paradigma en un estándar, impulsando la creación de un creciente número de CDNs. Las CDNs pueden ser clasificadas en CDNs privadas y CDNs públicas, de acuerdo con quién hace uso de esta infraestructura. Las CDNs privadas son infraestructuras desarrolladas por un proveedor de contenidos, por ejemplo Facebook o Netflix, para servir su propio contenido. En cambio, las CDNs públicas pertenecen a compañías que prestan el servicio de poner su infraestructura disponible a terceros para que desde allí sirvan su contenido, como es el ejemplo de Akamai. Las CDNs públicas emergen ante la necesidad de los desarrolladores de contenido y proveedores de servicios, que requieren del uso de estas infraestructuras, pero no les es económicamente factible desarrollar

su propia infraestructura. Frente a este escenario, en los últimos años se ha detectado una tendencia donde los proveedores de contenidos, una vez que se han establecido en el mercado, desarrollan sus propias CDNs privadas [7, 8]. Posiblemente esta tendencia haya estado relacionada a la reducción de los costos del equipamiento y el crecimiento de la conectividad de la red [9]. Sin embargo, más recientemente han surgido ciertos casos en la dirección opuesta, como por ejemplo Spotify [10], que han abandonado el uso de sus CDNs privadas para optar por estrategias totalmente basadas en CDN públicas.

La gran oferta de CDN públicas, de fácil uso y acceso, a precios accesibles y con gran cobertura, permite que los proveedores de contenido hayan desarrollado complejas estrategias de distribución de contenido, las cuales incluyen múltiples CDNs. Por ejemplo, cualquier servicio brindado a través de Internet depende de gran variedad de servicios complementarios, brindados por terceros, quienes deciden desde qué CDNs servirlos [11]. A su vez, en cualquier sitio web, se requiere de otros servicios tales como DNS, certificados de autenticación, y elementos provistos (ej: bibliotecas CSS o JS) desarrollados, mantenidos y brindados por terceros. La estrategia de distribución de cada uno de estos servicios complementarios, está a cargo de su desarrollador. Este complejo sistema de dependencias genera que el acceso a cualquier servicio brindado a través de Internet dependa de datos provistos desde múltiples CDNs. Sumado a la dependencia de múltiples proveedores externos, en muchos casos son los mismos desarrolladores de una aplicación o servicio, quienes deciden combinar múltiples CDNs (multi-CDNs) para hacer la distribución del contenido [12]. Estas estrategias de multi-CDNs pueden clasificarse en dos grandes grupos: dependientes o independientes de la ubicación del usuario. Las estrategias de multi-CDNs dependientes de la ubicación del usuario modifican las CDNs desde las cuales se sirve el contenido en función de cuál es la ubicación geográfica o el proveedor de Internet (del inglés Internet Service Provider, ISP) donde se encuentra el usuario. Esta adaptación de las plataformas utilizadas en función del usuario buscan maximizar la experiencia de usuario (del inglés User Experience, UX), y en algunos casos, reducir los costos de distribución. Por ejemplo, Microsoft y Apple utilizan diferentes CDNs en Norteamérica y Latinoamérica para brindar las actualizaciones de sus sistemas operativos, ya que cada una de las CDNs empleadas tienen distinta cobertura geográfica de las instancias de su sistema distribuido [12]. Las estrategias multi-CDNs independientes de la ubicación del usuario sirven distintos elementos o servicios que conforman la aplicación desde distintas CDNs. Por ejemplo, Netflix utiliza los servicios Amazon AWS para su front-end pero luego sirve los videos desde su propia CDN llamada Open Connect [13]. A pesar de la gran oferta de servicios de CDNs, surge el interrogante si el uso de CDNs no está llevando a la centralización de la infraestructura de Internet, donde la mayor parte de los servicios dependen de unas pocas plataformas.

El aumento del uso de Internet, y la potencial exposición de datos personales en el uso de servicios digitales, ha despertado iniciativas de múltiples sectores para proteger la identidad digital de los ciudadanos. La transformación digital (del inglés, Digital transformation) [14] ha reemplazado un vasto número de actividades presenciales por medio de servicios en la nube, incluyendo compras, banca [15] y gobierno [16], entre otras. A su vez, la irrupción de Internet ha dado lugar a nuevos servicios nativos, tales como las redes sociales. El aumento de uso de Internet por nuevas culturas y migración de servicios, ha aumentado la huella digital de los usuarios. Es decir, los usuarios voluntariamente o involuntariamente dejan rastros y proveen información personal al interactuar con cada uno de estos servicios. Más aún, muchos servicios en línea recolectan esta información para identificar a los usuarios y sus preferencias, y de esa manera luego ofrecerles publicidad orientada. Alarmados por la exposición de la privacidad de los usuarios, y corriendo el riesgo de que esta información privada sea filtrada o expuesta, múltiples corrientes de activistas, organizaciones sin fines de lucro e instituciones parlamentarias, han llevado a cabo iniciativas para proteger los datos personales de los usuarios. Entre los casos más destacados de los últimos años se encuentran las legislaciones GDPR (Unión Europea) [17], CCPA (California, EE.UU.) [18] y LGPD (Brasil) [19]. Aunque estas legislaciones han sentado precedentes en materia de protección, los usuarios aún carecen de métodos sencillos para conocer con cuáles plataformas están interactuando sus dispositivos, y en concreto, de dónde van y de dónde vienen los datos. Más aún, la comunidad científica ha estudiado la aplicación de estos marcos regulatorios [20]. Sin embargo, por el momento estos análisis se restringen a estudios específicos, sin dotar de herramientas a la ciudadanía para la inspección del cumplimiento normativo.

1.3. Contribución

Este trabajo tiene el objetivo de crear una aplicación para dispositivos móviles con Sistema Operativo Android que permita identificar las estrategias de distribución de contenido en las aplicaciones móviles. Esta aplicación busca permitirle al usuario determinar el volumen de datos intercambiado con cada CDN mediante sencillas visualizaciones en la pantalla. Esta herramienta innovadora tiene como objetivo potenciar análisis sobre la distribución de contenido beneficiando a la comunidad científica, legisladores, reguladores, organizaciones interesadas en los derechos digitales y usuarios.

1.4. Transferencia a otras comunidades y al resto de la sociedad

El objetivo de este proyecto es brindar una herramienta, en forma de aplicación móvil, que permita determinar y visualizar el uso de plataformas en la nube para la distribución de contenido. Aunque se requiere de conocimientos técnicos específicos, como los conceptos de Redes de Distribución de Contenidos (CDNs) y Sistemas Autónomos (ASes), esta herramienta pretende expandir su impacto más allá de las comunidades técnicas. A continuación resumimos las comunidades en las que `svrmap` puede contribuir.

Comunidad de las ciencias informáticas dedicada al estudio de Internet. Esta herramienta permitirá estudiar el uso de CDNs públicas y privadas en la distribución del contenido. Mediante estos análisis se podrá identificar las compañías que cuentan con infraestructuras propias para la distribución del contenido. También permitirá estudiar cual es la prevalencia de las CDNs públicas y si alguna de ellas domina el mercado. Por último permitirá estudiar los métodos de distribución de contenido basados en múlti-CDNs

Comunidad de las ciencias sociales dedicada al estudio de la tecnología. Las ciencias sociales enfocadas en el estudio de la tecnología abarcan un amplio espectro, incluyendo el impacto de la tecnología en la vida de las sociedades, la brecha digital, el ecosistema digital, entre otros. Para investigadores enfocados en el estudio de Internet, y por ende familiarizados cuestiones técnicas, se beneficiarán de contar con nuevos datos técnicos de las aplicaciones. Por ejemplo, se podrá estudiar si el uso de la infraestructura para distribuir el contenido de las Apps en los países latinoamericanos es mediante proveedores de servicios extranjeros. También se podrá estudiar las diferencias que existen en los esquemas de distribución de contenido entre empresas consolidadas y empresas emergentes.

Comunidad de operadores de Internet. Los operadores de Internet son los técnicos e ingenieros que configuran y diseñan las redes de Internet de los proveedores de Internet (del inglés Internet Service Provider, ISP). En la mayor parte de los casos, la distribución del contenido está completamente administrada por los generadores de contenido, como lo son las Apps. Ante este escenario, los operadores tienen pocas herramientas para incidir en la distribución, pero deben adaptar la capacidad de sus redes para poder satisfacer las demandas de sus usuarios. A través de esta herramienta, los operadores podrán identificar las estrategias multi-CDN implementadas por las App al servir a usuarios dentro de su ISP. Con este conocimiento, los operadores contarán con mayor información para el diseño y operación de sus redes.

Legisladores. Tal como fue mencionado en la Sección 1.2, en los últimos años se han creado múltiples marcos regulatorios para resguardar la privacidad de los usuarios de Internet. El uso de `svrmap` permitiría a los legisladores crear o mejorar marcos normativos en función de las prácticas de distribución de contenido utilizadas por los proveedores de contenidos. Un ejemplo sería determinar cuál es el paradero o los intermediarios utilizados en la recolección de datos personales de los usuarios.

Reguladores. Esta herramienta permitirá que los reguladores (con incumbencia en telecomunicaciones (ENACOM), de actividades comerciales (Defensa de la competencia), etc.) puedan verificar el cumplimiento del marco regulatorio, en condiciones donde se fijen pautas para la distribución del contenido. Un caso particular sería que el contenido sea servido desde CDNs físicamente presentes en el país. En esos casos `svrmap` podría identificar las CDNs desde las cuales se sirve el contenido, y luego verificar si esas plataformas cuentan con presencia en el territorio donde se aplica el marco regulatorio.

Organizaciones civiles en defensa de la privacidad digital. Un gran número de organizaciones civiles nacionales e internacionales, como por ejemplo Internet Society, Access Now y la Asociación por los Derechos Civiles (ADC), tienen como parte de su alcance promover la defensa de la privacidad digital. Estas organizaciones podrían beneficiarse de `svrmap` para estudiar la dinámica en la cual interactúan las Apps y si esto implica algún riesgo para los usuarios.

Público general. Finalmente, suponemos que aquellos usuarios interesados en conocer aspectos relacionados a la privacidad, como también el impacto de la distribución de contenido en la perfomance, podrán verse interesados en utilizar `svrmap`.

Capítulo 2

Desarrollo de svrmap

2.1. Abordaje del problema y decisiones de diseño

El objetivo principal de `svrmap` es la identificación de los servidores con que se comunican las distintas aplicaciones del dispositivo. Se decidió desarrollarla para los Sistemas Operativos (SO) Android y en lenguaje Java. Con el fin de proteger la privacidad de los usuarios, el análisis del tráfico interceptado se limitó a las cabeceras de los paquetes, ignorando el contenido de las comunicaciones.

El primer paso consistió en analizar diferentes alternativas que permitieran capturar el tráfico generado en un dispositivo con Sistema Operativo Android. Sin embargo, los Sistemas Operativos Android presentan un gran número de restricciones en los permisos de acceso al sistema, principalmente vinculados con proteger aspectos de la privacidad [21, 22]. Frente a este escenario, y con la intención de que la captura de tráfico fuera por medio de un aplicación corriendo en el teléfono, se optó por una implementación mediante un servicio VPN (*VPN Service*).

Aunque el servicio VPN permite interceptar los paquetes en el dispositivo móvil, su finalidad es radicalmente diferente a la planteada para `svrmap`, ya que este servicio propone hacer un reenvío de los paquetes a un servidor externo [23]. Sin embargo, si se utilizara un servidor remoto para el reenvío de paquetes, esto alteraría la observación de las interacciones con las CDNs. Esto se debe a que, en un servicio VPN, es el servidor remoto el que interactúa con las CDNs, en lugar del dispositivo móvil. Recordando que las aplicaciones ajustan su estrategia de distribución de contenido en función de donde se encuentra el dispositivo que realizó la petición (ver Sección 1.2), `svrmap` solo capturaría la estrategia de distribución de contenido de las Apps para la ubicación (geográficas y topológica) del servidor remoto. En otras palabras, implementar el servidor remoto no le permitiría al usuario identificar las estrategias de distribución de contenido para el proveedor en el cual que se encuentra conectado. Entonces, la solución a este problema

se dio por medio de reinterpretar el concepto de VPN, dando lugar a una estructura denominada *AutoVPN*. Este concepto se basa en que el servidor remoto VPN sea el mismo dispositivo, evitando que se altere el punto de observación de la estrategia de distribución de contenido. Para poder llevar a cabo la *AutoVPN* se tuvieron que resolver ciertos aspectos específicos que serán posteriormente detallados (ver Sección 2.2.3.2).

Otro requerimiento de diseño es el hecho de poder contar con información que indique la ubicación de los servidores utilizados para distribuir el contenido. Una alternativa de uso de DNS para la implementación de la aplicación hubiera permitido descubrir información de la ubicación geográfica de los servidores, utilizando conceptos de ingeniería inversa de las practicas de nomenclatura de la infraestructura [24]. Sin embargo, la alternativa de DNS fue descartada a raíz de las dificultades presentadas por la creciente adopción de DoH (*DNS over HTTPS*) [25]. Por lo tanto, usando el paradigma de *AutoVPN*, se optó por identificar la ubicación topológica de los servidores que sirven el contenido. Para esto, se usó el concepto de Número de Sistema Autónomo (del inglés Autonomous System Number, ASN) [26] que indica el Sistema Autónomo (del inglés Autonomous System, AS) donde se encuentra el servidor, por ejemplo Google, identificado por el ASN 15169[1]. La identificación de los ASes se implementó a través de un servicio provisto por RIPE [2] por medio de una API REST. Específicamente, este servicio determina el ASN en el cual se encuentra ubicada una dirección IP, que en el caso de `svrmap` corresponde a las direcciones IP de los servidores de las CDNs.

Finalmente, se debió resolver la persistencia de datos, su exportación y la presentación de información al usuario mediante una interfaz de usuario (del inglés User Interface, UI) que permita explorar estos datos.

2.2. Implementación

En esta sección se detallan las cuestiones relativas a las características de la aplicación y su estructuración:

- **Estructura general**. Descripción de alto nivel de los componentes que forman a la aplicación, agrupados por funcionalidades.

- **Dependencias externas**. Listado y descripción de las dependencias externas utilizadas por la aplicación, tanto bibliotecas importadas como APIs consumidas.

[1] Información pública de Google publicada en PeeringDB donde indica que la compañía se identifica con el ASN 15169. `https://www.peeringdb.com/net/433`

[2] Registro Regional de Internet para la región de Europa y Medio Oriente `https://www.ripe.net/`

- **Mecanismos internos**. Detalle de mecanismos internos utilizados, requeridos para el correcto funcionamiento de la aplicación.

2.2.1. Estructura general

La arquitectura de la aplicación, como se muestra en la Figura 2.1, está estructurada a partir de 3 componentes principales:

1. **Packet sniffer**. Una capa de *packet sniffing*, implementada como una AutoVPN, que permite que las otras aplicaciones mantengan su tráfico de datos mientras de forma transparente registra los paquetes enviados y recibidos.

2. **Enriquecimiento de datos**. Un módulo que, a partir de las interacciones capturadas, obtiene información de las IPs de los servidores de las CDNs y sus correspondientes ASNs, enriqueciendo la información del tráfico observado.

3. **Interfaz de usuario**. Una interfaz de usuario que permite visualizar la información recopilada desde distintas vistas.

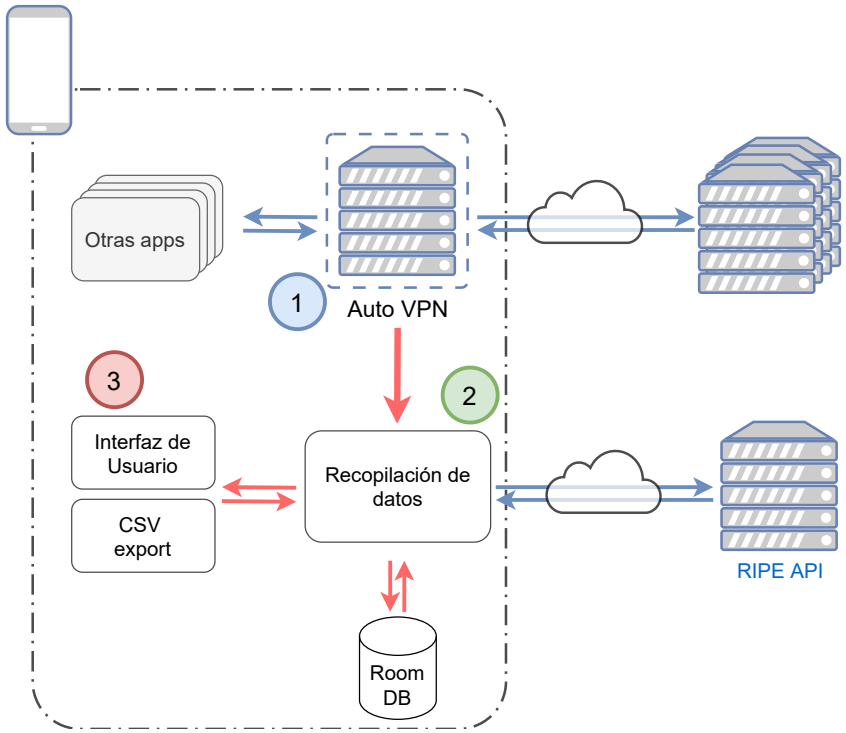


Figura 2.1: Arquitectura a nivel macro de la aplicación.

Enfocándonos en la estructura del proyecto a nivel de código, como se indica en la Figura 2.2, la división está diseñada a partir de conjuntos de responsabilidades. A

continuación se detalla cada uno de los directorios que componen este árbol y sus tareas asociadas.

```
.
|-- data/
|       `-- IpInfo.java
|-- models/
|       |-- Application.java
|       |-- ApplicationDetail.java
|       |-- ApplicationListItem.java
|       |-- ApplicationToAutonomousSystemRelation.java
|       |-- AutonomousSystem.java
|       |-- AutonomousSystemDetail.java
|       |-- AutonomousSystemListItem.java
|       `-- ExpandedApplicationToAutonomousSystemRelation.java
|-- persistence/
|       `-- db/
|       |-- Converters.java
|       |-- LocalDatabase.java
|       |-- daos/
|       |       |-- ApplicationDao.java
|       |       |-- ApplicationToAutonomousSystemRelationDao.java
|       |       `-- AutonomousSystemDao.java
|       `-- repositories/
|               |-- ApplicationRepository.java
|               `-- AutonomousSystemRepository.java
|-- routing/
|       |-- IncomingPacketHandler.java
|       |-- OutgoingPacketHandler.java
|       |-- TcpTask.java
|       `-- UdpTask.java
|-- utils/
|       |-- ByteUtils.java
|       |-- CSVExporter.java
|       |-- Constants.java
|       |-- RIPEApiParser.java
|       |-- StringFormatter.java
|       `-- VpnStoppedException.java
|-- views/
|       |-- appdetail/
|       |       |-- AppDetailFragment.java
|       |       `-- AppDetailViewModel.java
|       |-- applist/
|       |       |-- AppListFragment.java
|       |       |-- AppListViewAdapter.java
|       |       `-- AppListViewModel.java
|       |-- asdetail/
|       |       |-- AutonomousSystemDetailFragment.java
|       |       `-- AutonomousSystemDetailViewModel.java
|       |-- aslist/
|       |       |-- AutonomousSystemListFragment.java
|       |       |-- AutonomousSystemListViewAdapter.java
|       |       `-- AutonomousSystemListViewModel.java
|       `-- home/
|               `-- HomeFragment.java
|-- MainActivity.java
`-- SvrmapVpnService.java
```

Figura 2.2: Estructura de archivos del proyecto.

- `Data/`. Aquí se recopila, unifican y consolidan los datos de las direcciones IPs y sus correspondientes ASes, incluyendo la información recopilada de las transmisiones y la obtenida mediante la API de RIPE. Específicamente, es la Clase `IpInfo` la que lleva a cabo todas estas funcionalidades.

- `Models/`. En este punto se estructuran los datos obtenidos y se almacenan en las tablas de la base de datos. En particular, la información se incluye en 3 tablas

  - `Application`. Información sobre las aplicaciones observadas.
  - `AutonomousSystem`. Información sobre ASes a los que pertenecen las IPs destino observadas.
  - `ApplicationToAutonomousSystemRelation`. Transmisiones observadas entre aplicaciones e IPs. Incluye especialmente el volumen de datos enviados y recibidos.

- `Persistence/db/`. Definición de la base de datos, DAOs de Room y repositorios.

- `Routing/`. Aquí se describen las Clases relacionadas a mantener y procesar las transmisiones entre las otras apps y el exterior.

  - `OutgoingPacketHandler`. Administra las tareas (`tasks`) y control del envío de paquetes salientes.
  - `IncomingPacketHandler`. Recepción centralizada de paquetes entrantes.
  - `TcpTask`. Administra una transmisión TCP, incluyendo actualización del estado de la conexión (ACK, seqNum, etc.) y recopilación de datos.
  - `UdpTask`. Administra una transmisión UDP, incluyendo recopilación de datos.

- `Utils/`. Aquí se integran diversas funciones de utilidad, incluidas la función de exportar los datos a CSV, parseo de respuestas de la RIPE API, diversas modificaciones del formato de datos y definición de constantes de la aplicación.

- `Views/`. Elementos de la interfaz de usuario incluyendo Fragment, ViewAdapter y ViewModel para cada uno.

- `MainActivity`. Actividad principal sobre la que se basa la interfaz de la aplicación.

- `SvrmapVpnService`. Este es el servicio VPN propiamente dicho. Por cuestiones de permisos del Sistema Operativo Android, este servicio es el responsable de construir el mapeo entre conexiones y aplicaciones.

2.2.2. Dependencias externas

La aplicación también cuenta con dependencias de bibliotecas externas para su funcionamiento. A continuación, se describen las principales bibliotecas utilizadas para su desarrollo y la función que cumple cada una de ellas dentro de `svrmap`.

1. **PCap4J** [**27**]. Esta una biblioteca utilizada en la aplicación para llevar a cabo el parseo de las cabeceras de los paquetes entrantes y salientes. Esta tarea es necesaria para varias funciones, pero principalmente para identificar las direcciones IP de los servidores con los cuales se conecta cada aplicación.

2. **FastCSV** [**28**]. Esta biblioteca brinda la lectura y escritura de archivos CSV. Aunque existen múltiples alternativas para aplicaciones Java corriendo en Android, se optó por esta biblioteca, ya que es un módulo de bajo peso, lo cual tiene un impacto positivo para el *dump* de datos de las ejecuciones de `svrmap`.

3. **Volley** [**29**]. Esta es una biblioteca oficial para el intercambio de peticiones del protocolo HTTP, la cual fue utilizada en `svrmap` para interactuar con la API de RIPE.

4. **Room** [**30**]. Esta es una biblioteca oficial que provee una interfaz para un uso más idiomático/eficiente de una base de datos SQLite.

5. **RIPE API** [**31**]. Este es un servicio provisto por medio de una API REST, fue utilizando para generar el mapeo de direcciones IP en sus correspondientes ASNs.

2.2.3. Mecanismos internos

En las siguientes subsecciones se describen los mecanismos internos utilizados por la aplicación, definiendo un *mecanismo interno* como un conjunto de funciones realizado por una entidad o grupo de ellas, internas a la aplicación, que permite llevar adelante una tarea necesaria para el funcionamiento de `svrmap`. Estos son:

- **Routing**. El mecanismo de ruteo es el encargado de interceptar y redireccionar los paquetes IP, de forma de mantener la conectividad del dispositivo.

- **Packet forging**. El forjado de paquetes es un mecanismo necesario para sortear el fenómeno de *impedance mismatch* provocado por limitaciones del Sistema Operativo.

- **Estructuras de datos**. La estructuración de los datos es requerida para el almacenamiento de la información recopilada.

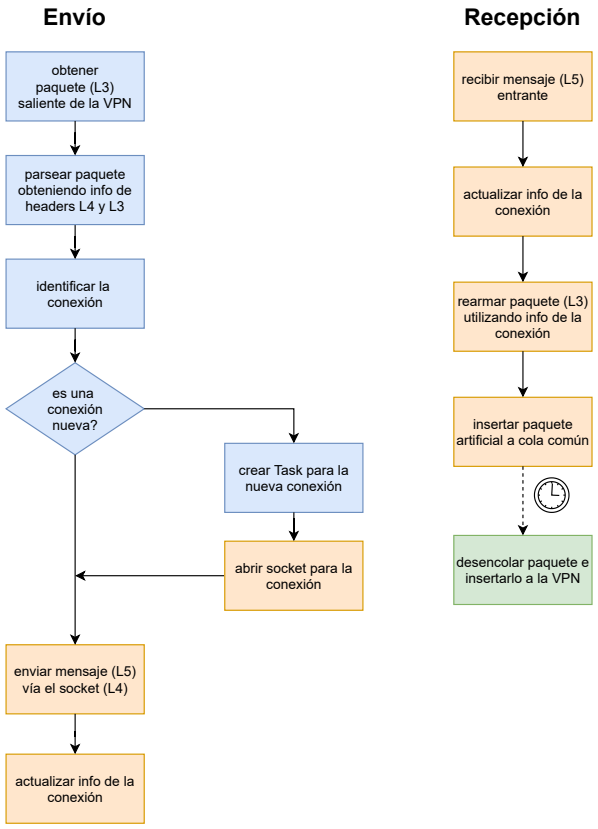


Figura 2.3: Esquema de ruteo. En *azul*, las acciones realizadas por el OutgoingPacketHandler, en *naranja* UDP/TCPTask, en *verde* IncomingPacketHandler. Las flechas punteadas indican acciones realizadas en forma asincrónica (ocurren eventualmente).

2.2.3.1. Ruteo (routing)

El principal componente de `svrmap` es el que permite interceptar los datos entrantes y salientes de las aplicaciones del dispositivo. Para poder llevar acabo esta tarea, `svrmap` crea una capa intermedia entre las aplicaciones y el módulo de envío de paquetes presente en el Sistema Operativo. Es importante remarcar que `svrmap` debe poder insertar esta capa intermedia de forma transparente para no alterar el normal funcionamiento de las aplicaciones y del Sistema Operativo.

Este componente esta formado por medio de tres entidades:

1. `VPNService`. Esta entidad provee dos descriptores de archivos (file descriptors), uno de lectura para capturar los datagramas IP salientes generados por las Apps y otro de escritura para enviarle datos a las Apps a partir de los datagramas IP recibidos en el dispositivo.

2. `Tasks`. Cada tarea (task) administra una conexión individual, incluyendo el envío y recepción de paquetes desde el exterior. En esta entidad se adaptan las cabeceras

de los protocolos de la pila TCP/IP para no generar ningún impacto en el funcionamiento de las aplicaciones. Las tareas se encargan de administrar conexiones TCP y envíos de datagramas UDP.

3. `Packet Handlers`. Esta entidad administra la creación de las tareas (tasks), y su comunicación con el `VPNService`. Esta administración tiene un esquema *many-to-one*, ya que el dispositivo corre múltiples tareas pero una única entidad `VPNService` administra todas las conexiones del dispostivo.

La Figura 2.3 ilustra el proceso de ruteo el cual se encuentra divido en los flujos de envío (izquierda) y recepción (derecha) de datagramas IP. Ambos presentan dificultades diferentes, debido a que en un sentido la relación es one-to-many y en el otro many-to-one.

El envío de información comienza con el *OutgoingPacketHandler* leyendo de la VPN un paquete saliente previamente capturado. A partir de parsearlo y leer sus headers de L3 (capa de red) y L4 (capa de transporte) se identifica la conexión, tras lo cual se puede establecer si es nueva o si el paquete pertenece a una preexistente. De ser nueva, se crea según el protocolo de L4 un TCPTask o un UDPTask asignado a la conexión, los cuales en su proceso de construcción incluyen la apertura de un socket del correspondiente protocolo hacia el par (ip, puerto) destino que es el que formalmente se comunicará con el servidor al que se intentaba conectar la app original. Una vez creada cada task se almacena, por lo que para conexiones preexistentes se reutiliza la creada permitiendo mantener el estado de la comunicación. Obtenida la task, nueva o no, se envía vía su socket el mensaje (L5 - capa de aplicación) contenido en el paquete capturado, actualizando luego el estado de la conexión en el caso de TCP.

El flujo de recepción de paquetes funciona en sentido inverso: comienza con el socket de un task recibiendo un mensaje (L5), tras lo cual primero se actualiza el estado de la conexión si corresponde y luego se construye el paquete IP artificial. Éste se crea en blanco y se le insertan el mensaje recibido (L5) y headers de L3 y L4 forjados en base al estado de conexión, con valores tales que la app original interprete el nuevo paquete como real y oriundo del servidor al que se quería comunicar. Ya listo, el paquete es insertado en una cola común para todas las tasks. El *IncomingPacketHandler*, que periódicamente desencola paquetes de la misma y los escribe en la VPN, eventualmente hará lo propio con el paquete en cuestión, cerrando así el circuito de envío y recepción de mensajes.

2.2.3.2. Packet forging

El hecho de colocar un `VPNService` entre las aplicaciones y la salida de los paquetes, y que este se comporte de manera transparente, genera desafíos técnicos para resolver. En primer lugar, se debe recordar que los servicios VPN están orientados a que el dispositivo

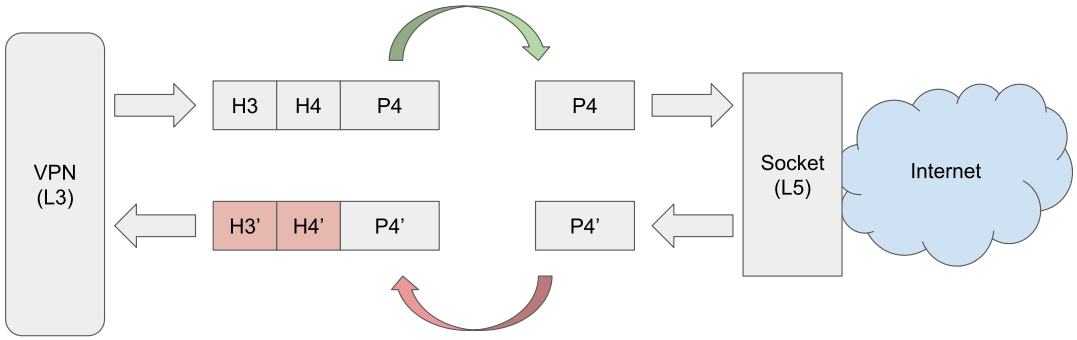


Figura 2.4: Ilustración del fenómeno de *impedance mismatch*.

se conecte con un servidor remoto, y este servidor sea quien se encargue, en nombre del usuario en el dispositivo, de contactarse con los proveedores de contenido. Sin embargo, como se explicó en la Sección 2.1, si se utilizara un servidor externo, se distorsionarían los datos recolectados. Por lo tanto, se optó por la opción de la *AutoVPN*, donde el servidor remoto corre en el mismo dispositivo.

Para lograr que este servidor remoto se ejecute dentro del mismo dispositivo hay que controlar en simultáneo flujos de datos a nivel de capa de red (L3) y de transporte (L4), como se indica en la Figura 2.4. Este fenómeno fue denominado *impedance mismatch*. Dicho manejo se lleva a cabo por la entidad definida como tarea (`Task`), que intermedia en cada una de las conexiones de cada aplicación con el exterior. Cuando una aplicación quiere establecer una comunicación al exterior, esto resultará en un datagrama IP que será interceptado por el VPNService. Aquí, una tarea se encarga de hacer la lectura del VPNService, donde los datos que obtendrá estarán encapsulados dentro de un datagrama IP. Esta intercepción permite finalmente capturar los datos de las cabeceras asociadas, lo cual es central para la identificación de las CDNs a contactar. Sin embargo, dado que las versiones actuales de Android no permiten el manejo de `rawSockets`[3], la tarea no podrá enviar tal y como está ese datagrama al exterior. Para esto, la tarea debe parsear el contenido del datagrama IP y establecer una nueva conexión al exterior por un nuevo socket. Este problema todavía persiste cuando regresa la respuesta generada por el servidor contactado, ya que ahora el `VPNService` aguarda por un datagrama IP para enviar a la aplicación. Por lo tanto, la tarea forjará un nuevo datagrama IP para

[3]raw socket: `https://man7.org/linux/man-pages/man7/raw.7.html`

generar la respuesta necesaria. Es importante que la tarea haya conservado las variables de estado de la conexión (por ejemplo ACK, SeqNumb, flags, etc.) para poder crear un paquete acorde a lo que la aplicación está esperando.

2.2.3.3. Estructuras de datos

Los datos de las cabeceras recopilados al interceptar los datagramas IP son estructurados a partir de las premisas de diseño. En primer lugar, para mantener el funcionamiento de las aplicaciones luego de haber insertado el `VPNservice`, es necesario almacenar temporalmente los datos que corresponden a la conexión, incluyendo direcciones y puertos de origen y destino. Recordando que el objetivo de la aplicación es poder determinar las CDNs, y más precisamente los ASes correspondientes a estas plataformas, se deben recopilar datos que incluyan las direcciones IP contactadas, sus ASes y la aplicación que la contactó. El almacenamiento de estas estructuras se hace por medio de tres tablas: *(i)* datos de aplicaciones (Tabla Apps), *(ii)* Sistemas Autónomos (Tabla AutonomousSystems), y *(iii)* una relación *many-to-many* entre ambos (Tabla AppToAutonomousSystemRelation).

Tabla Apps

- `uid (primary key, int)`: identificador de la aplicación a nivel de SO.
- `name (string)`: nombre con que la aplicación está registrada en el dispositivo.

Tabla AutonomousSystems

- `asn (primary key, int)`: Número de Sistema Autónomo en donde se encuentra la dirección IP de un servidor con el cual se contacta una aplicación.
- `rir (string)`: RIR al que pertenece el ASN contactado por la App.
- `owner (string)`: Nombre de la organización a la que el RIR le delegó el ASN contactado.
- `country (string)`: Código de país (ISO 3361-2) del país donde se encuentra radicada la organización que opera el ASN.
- `registration_date (date)`: Fecha de registro del sistema.

Tabla AppToAutonomousSystemRelation

- `session_id (primary key, int)`: Número de identificación de la ejecución actual.

- `app_uid (primary key, foreign key→uid, int)`: uid de la aplicación que interviene en la conexión.
- `asn (primary key, foreign key→asn, int)`: ASN donde se encuentra la dirección IP de un servidor con el cual se contacta una aplicación.
- `ip_address (primary key, string)`: Dirección IP destino de la conexión.
- `total_bytes_out (int)` Cantidad total de bytes enviados.
- `total_bytes_in (int)`: Cantidad total de bytes recibidos.

2.3. Vistas de la aplicación

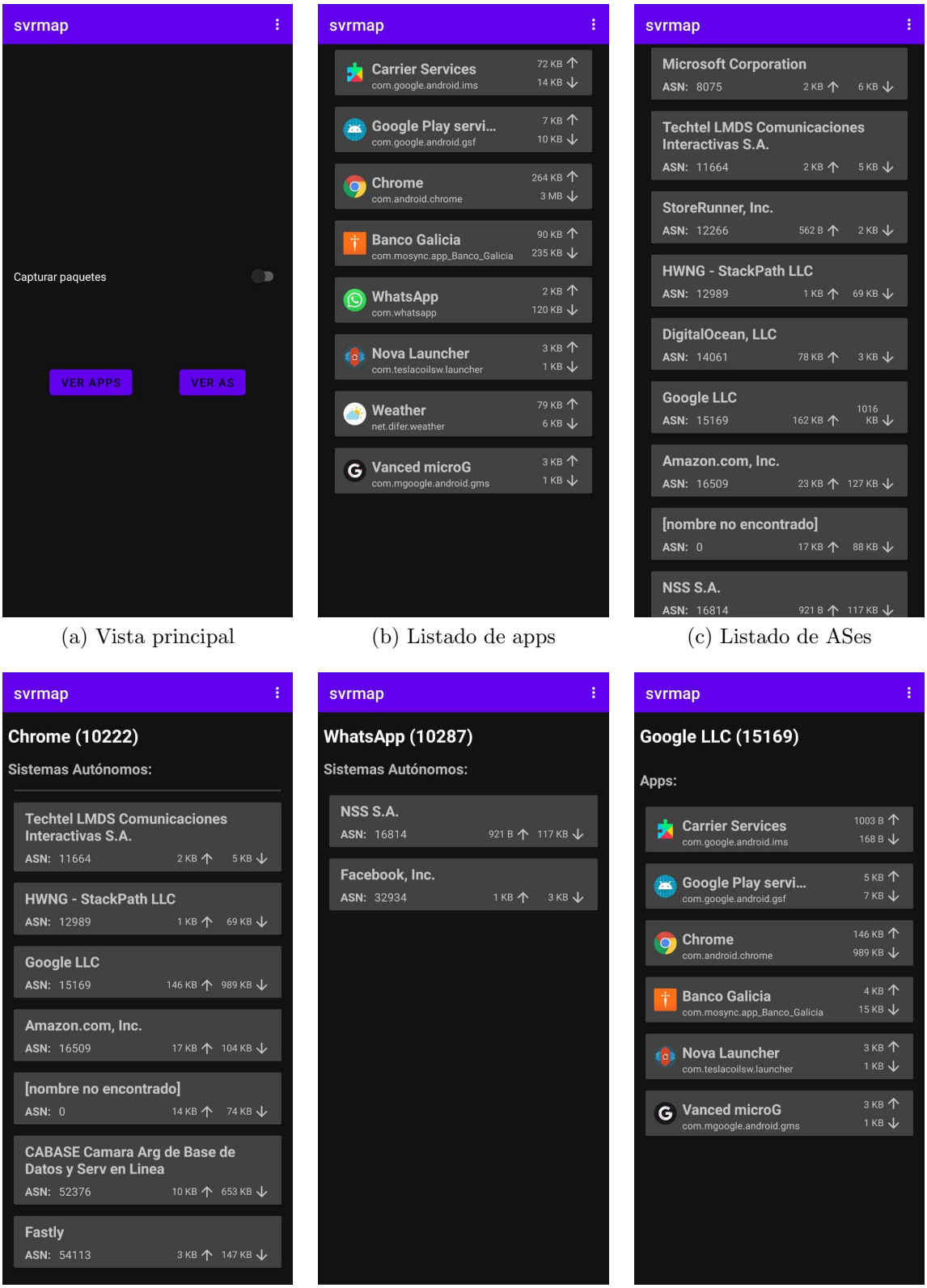
La Figura 2.5 muestra seis vistas de `svrmap`, la pantalla de inicio para comenzar o detener la captura (Fig. 2.5a), diferentes formas de acceso a los datos (por aplicación (Fig. 2.5b), por Sistema Autónomo (Fig. 2.5c)), dos ejemplos de vistas por aplicación (Fig. 2.5d y Fig. 2.5e) y una vista por Sistema Autónomo (Fig. 2.5f). Comparando la lista de aplicaciones (Fig. 2.5b) y de ASes (Fig. 2.5c), se observa que para un listado reducido de aplicaciones, existe un listado extenso de ASes que proveen contenido. En la aplicación se provee una interacción tal que por medio de las apps listadas en la Figura 2.5b, se puede acceder al detalle de las interacciones de cada una, tal como se muestra para Chrome y WhatsApp, en las Figuras 2.5d y 2.5e, respectivamente. En la Figura 2.5f se muestra una visualización equivalente, pero centrada en un AS en particular, la cual es accedida tocando cualquier AS en la Figura 2.5c.

El caso de WhatsApp, presentado en la Figura 2.5e), muestra varios aspectos importantes para destacar, y donde `svrmap` permite visibilizarlos. En primer lugar, los datos muestran una marcada integración vertical. En este servicio prestado por Facebook, existe una integración vertical en la distribución de contenido de Whatsapp, donde es servido casi exclusivamente a través del AS de Facebook (AS32934). En segundo lugar, `svrmap` revela la presencia de caches locales ubicados en iPlan/NSS (AS16814), proveedor desde el cual se generó la captura.

2.4. Limitaciones

Para concluir este Capítulo, se presentan las principales limitaciones halladas durante el desarrollo de `svrmap`.

Despliegue de DoH. Entre los objetivos planteados por `svrmap` estaba capturar las consultas DNS generadas por el dispositivo, ya que un gran número de proveedores de



(a) Vista principal (b) Listado de apps (c) Listado de ASes

(d) Chrome (e) WhatsApp (f) Apps servidas por GGC

Figura 2.5: Ejemplos de vistas de `srvmap`.

contenido hacen uso de DNS para manejar la la asignación de recursos de la CDNs [32]. Sin embargo, en las etapas iniciales del proyecto se detectaron interferencias generadas por DoH que eran difíciles de resolver de manera sencilla. Para evitar estos inconvenientes, y dado que la incipiente pero enérgica adopción de DoH podía comprometer la recolección de esta información, se decidió posponer este aspecto de la aplicación para trabajos futuros.

Volley. La biblioteca Volley fue utilizada para interactuar con la API de RIPE , aunque presentó problemas en los arribos y la intercepción de paquetes. En primer lugar, el funcionamiento de Volley es asincrónico y sin ninguna clase de parámetro para controlar los tiempos de arribos. Esta limitación genera que los datos de RIPE puedan arribar una vez concluida la corrida de exploración, demorando la presentación de los datos.

El segundo es que Volley funciona como un proceso aparte de forma que el consumo de la API de RIPE, desde el punto de vista del VPNService, se ve como tráfico de una aplicación más el cual interceptará. Esto provoca un ciclo infinito que detiene la aplicación:

1. Se intercepta tráfico a una IP de una app.
2. Se recopila la IP y se emite una consulta a RIPE API por información sobre la misma.
3. Esta consulta se emite en la forma de Volley enviando un paquete a la IP del servidor de RIPE API.
4. Como es en sí un paquete saliente de una aplicación que no es svrmap, es interceptado por el VPNService, volviendo al paso (1).

Como esta limitación está dada por construcción de Volley no es posible de corregir en forma correcta sino sólo salvar: en este caso se decidió utilizar un *bypass* donde los paquetes emitidos por Volley no son interceptados. Si bien exploraciones realizadas durante el desarrollo del proyecto mostraron que sólo `svrmap` utilizaba Volley, este tipo de medida admite el riesgo de no recopilar tráfico de otras aplicaciones en caso de que utilicen dicha librería para manejar su tráfico. La solución correcta y definitiva a esta limitación es el desarrollo de un stack propio para la realización de consultas a la API de RIPE sin requerir de un proceso externo.

Compatibilidad con diferentes versiones del SO. La aplicación requiere de una versión Android 8.0 (API 26) o superior debido al manejo de permisos del Sistema Operativo para las tareas que se encuentran corriendo en el *foreground*.

Capítulo 3

Caso de estudio: Distribución de contenido en aplicaciones de diarios

Este Capítulo presenta los resultados obtenidos al analizar las estrategias de distribución de contenidos de las aplicaciones de diferentes diarios, realizado en base a información recopilada utilizando `svrmap`.

3.1. Descripción del análisis

Condiciones Generales de los Experimentos

Dispositivo	Modelo SO	Nokia 7.1 Android 10
Captura	Condiciones Diarios	Realizado durante la tarde (17:00hs), con una duración de 5 minutos La Nación (AR), Página 12 (AR), Le Monde (FR), El País (ES) y La Repubblica (IT)

Condiciones Particulares de los Experimentos

		Experimento #1	Experimento #2
ISP	Nombre	iPlan	Personal (Telecom-Fibertel)
	ASN	16814	7303
	Red de acceso	Fija (Fibra Óptica)	Móvil (4G)
	Conectividad	WiFi	Red celular

Cuadro 3.1: Descripción de condiciones para captura de datos. Los experimentos consistieron en utilizar las apps correspondientes a los diarios mencionados, primero conectado mediante la red de WiFi, y luego utilizando datos móviles.

Para poder estudiar el impacto y la usabilidad de `svrmap`, se decidió llevar adelante un caso de estudio enfocado en investigar las estrategias de distribución de contenido

de diarios de diferentes países. Aunque para poder generar conclusiones firmes respecto del funcionamiento de estas aplicaciones es necesario llevar a cabo experimentos a gran escala, el reducido estudio que se llevó a cabo muestra indicios suficientes respecto a la contribución de `svrmap` y la prevalencia de estrategias multi-CDNs.

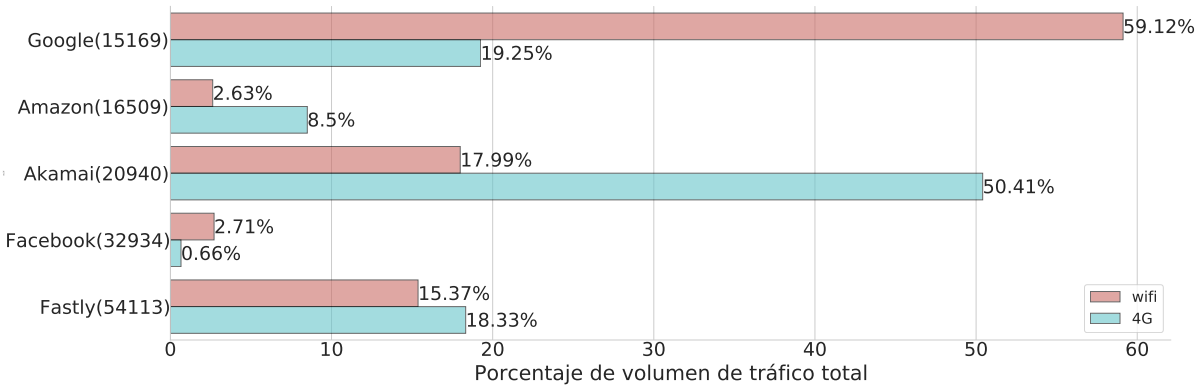
El objetivo de los experimentos estuvo centrado en estudiar el impacto de la ubicación topológica del usuario en la estrategia de distribución de contenido. La Tabla 3.1 resume las principales variables controladas en cada uno de los dos experimentos ejecutados. En ambos experimentos se repitieron los diarios inspeccionados, siendo estos La Nación (AR), Página 12 (AR), Le Monde (FR), El País (ES) y La Repubblica (IT). La selección de diarios utilizados incluyó diversos medios nacionales e internacionales para tener un panorama más amplio de las estrategias de distribución de contenido en diferentes puntos del planeta. Más aún, la ubicación de los puntos de observación se encuentra en la zona donde estos medios tienen su mayor caudal de público, lo que genera un interés de analizar la estrategias de distribución de contenido para usuarios en zonas remotas. Además, otras variables que permanecieron constantes en ambos experimentos fueron la hora del análisis, el tiempo transcurrido durante la navegación y los parámetros del dispositivo. La principal diferencia en ambos casos fue la red de acceso mediante la cual el dispositivo accedió al contenido. El primer experimento fue llevado a cabo desde la red de iPlan (AS16814), el cual prestaba conectividad residencial por fibra óptica y enlazando al dispositivo por medio de un Access Point WiFi. En cambio, en el segundo experimento se usó la red móvil 4G del operador Personal, parte del grupo Telecom-Fibertel (AS7303).

3.2. Prevalencia de las grandes CDNs

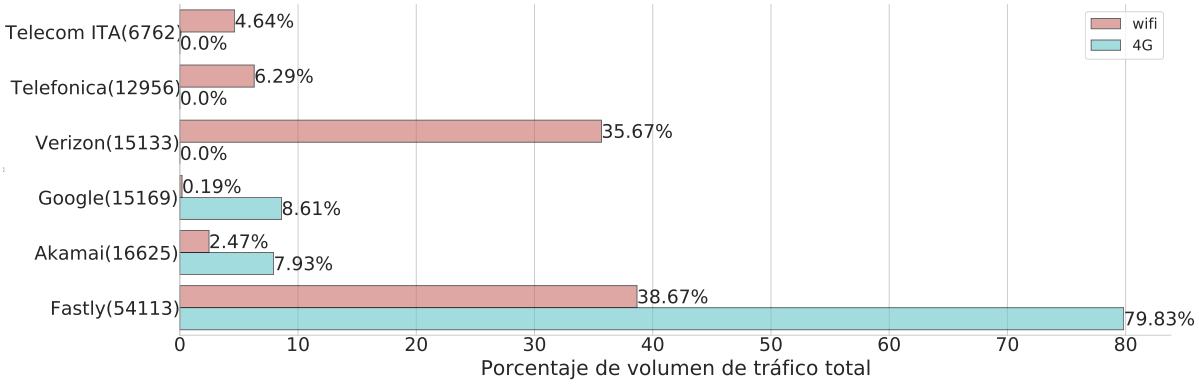
El primer análisis de los datos recolectados en los experimentos se centró en estudiar la prevalencia de las grandes CDNs (por ejemplo Amazon AWS, Google Cloud, etc) en las estrategias de distribución de contenido de los diarios. Nuestra hipótesis se basó en que la ubicuidad de estas CDNs hará que también sean las principales fuentes de distribución de contenido para los diarios.

La Figura 3.1 presenta los datos recibidos desde cada CDN (identificada cada una a través de su Número de Sistema Autónomo), para las aplicaciones del diario El País (Fig. 3.1a) y Le Monde (Fig. 3.1b). En esta figura se comparan los cambios de patrones de tráfico observados en el experimento con redes WiFi o redes móviles. Aunque los niveles de tráfico fluctúan en función del proveedor de acceso utilizado, en ambos experimentos para ambas aplicaciones se observa una significativa presencia de las principales CDNs, incluyendo a Fastly, Edgecast-Verizon, Akamai, Google y Amazon, entre otros.

En ambos casos se representan sólo los ASes cuya contribución para alguno de los



(a) El País



(b) Le Monde

Figura 3.1: Porcentaje de volumen de tráfico registrado para las aplicaciones de El País (Fig. 3.1a) y Le Monde (Fig. 3.1b). Las barras comparan el tráfico recibido desde cada CDN según el proveedor utilizado. En ambos casos se han descartado los ASes que no contribuyeran en al menos 3 % en algún caso.

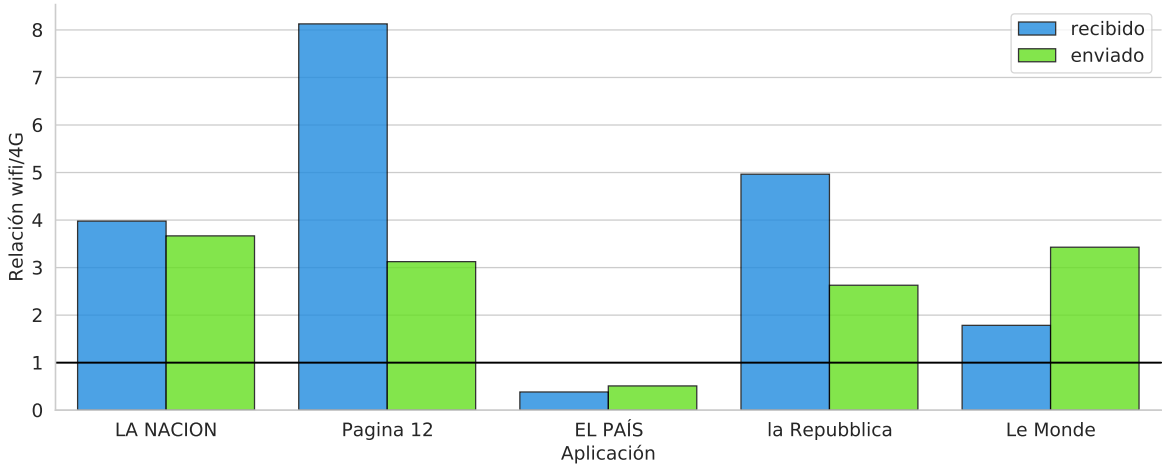


Figura 3.2: Relación entre volumen de tráfico manejado por las aplicaciones entre redes. La línea negra delimita aquellas que aumentan su volumen de las que lo reducen.

tipos de red fuera de al menos 3 %. Esto implicó eliminar 5 de los 10 ASes con representación en alguna de las redes para El País (Fig. 3.1a) y 19 de los 25 para Le Monde (Fig. 3.1b), mostrando cómo un grupo reducido del total de proveedores representan casi la totalidad del tráfico: 97.82 % (WiFi), 97.15 % (4G) para El País y 87.93 % (WiFi), 96.37 % (4G) para Le Monde.

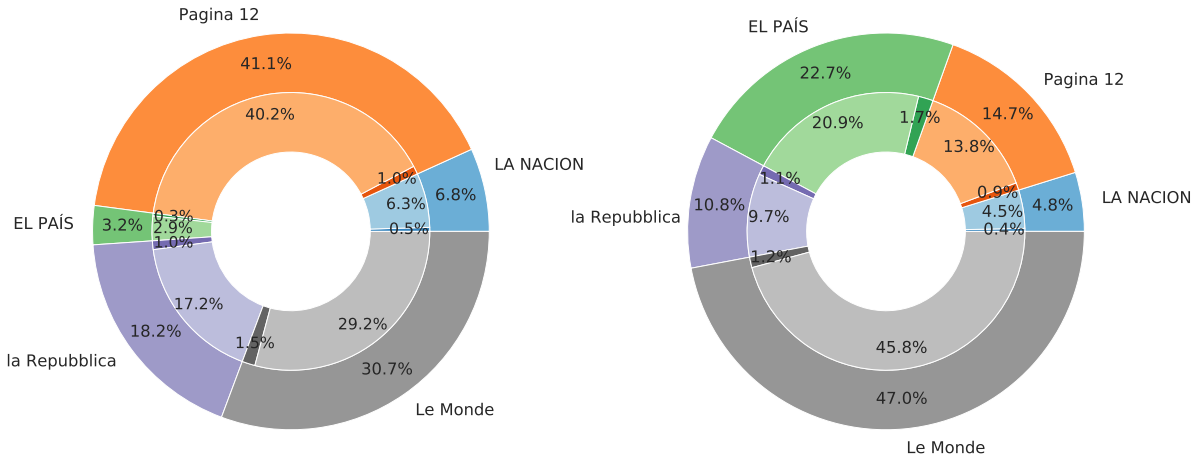


Figura 3.3: Fracción del volumen de tráfico total relevado, por aplicación y para red doméstica (iPlan, AS16814, WiFi) a la izquierda y red móvil (Personal, AS7303, 4G) a la derecha. Arcos internos indican el tráfico en entrante (claro) y saliente (oscuro). Todos los porcentajes son valores absolutos.

3.3. Alteraciones en los volúmenes de tráfico

Otro de los aspectos que se quiere evaluar es si el volumen de tráfico (medido en Bytes) cambia según el proveedor desde el cual se accede. Específicamente, nuestro interés se centra en comprender si los proveedores de contenido modifican la estrategia de distribución, incluyendo CDNs utilizadas y cantidad de datos enviados, dependiendo de si el usuario se encuentra conectado por medio de una red móvil. Es probable que en estos escenarios de redes con menores prestaciones, el proveedor de contenido reduzca el volumen de datos para no comprometer el tiempo de descarga.

La Figura 3.2 muestra la fracción de bytes recibidos por cada aplicación dependiendo de la red de acceso utilizada. Esta fracción tiene como numerador el volumen de datos recibidos por la red WiFi y como denominador la misma medición pero para la red 4G. En todos los casos, a excepción de El País, observamos que la descarga de datos (presentada en color azul) fue superior al utilizar la red de acceso WiFi, ya que la fracción siempre fue mayor a 1. Llamativamente, para el caso de Página/12 la fracción ronda el valor de 10, indicando que el volumen de datos recibido a través de WiFi es casi un orden de magnitud mayor que el mismo tráfico recibido a través de la red 4G.

Aunque estos resultados confirman la suposición de que la tasa de transferencia disminuye para usuarios conectados por 4G, es necesario mencionar ciertas limitaciones de los experimentos. En primer lugar, la interacción con las aplicaciones fue ejecutada de forma manual, generando que la duración de cada interacción no fuera totalmente idéntica en cada caso. Estas alteraciones pueden haber introducido ruido en las mediciones. En segundo lugar, debido a la cantidad limitada de puntos de observación, es decir redes de acceso desde donde se ejecutaron los experimentos, no podemos concluir si este fenómeno de fluctuación del tráfico se debe a la tecnología de la red de acceso o simplemente el proveedor.

La figura 3.3 muestra la representatividad de cada aplicación según el tipo de red en términos de porcentaje del volumen de tráfico total relevado. Además, cada uno de ellos se encuentra desglosado en tráfico entrante y saliente. De estos arcos internos podemos observar una asimetría común a todas las aplicaciones entre volúmenes de tráfico, notando que en todos los casos para una aplicación dada el volumen de tráfico entrante es alrededor de un orden de magnitud mayor al saliente. Dirigiendo la atención hacia los arcos externos vemos que tanto La Nación como La Repubblica están por debajo del 20 % del volumen total (el esperado en una distribución uniforme) para ambos tipos de red mientras que Le Monde se encuentra siempre muy por encima. En los casos de El País y Página/12 vemos un cambio drástico en su fracción de volumen total, respaldada a su vez por lo ya visto en la figura 3.2.

3.4. Incidencia de red de acceso sobre proveedores

Finalmente queremos investigar las CDNs involucradas en la distribución de contenido de cada medio periodístico, desde los dos puntos de observación (WiFi y 4G). Aunque en las secciones previas ya se observaron cambios en las CDNs implementadas según el punto donde se encuentra el usuario, ahora queremos contrastar cuáles siempre están presentes, y cuáles varían según la ubicación.

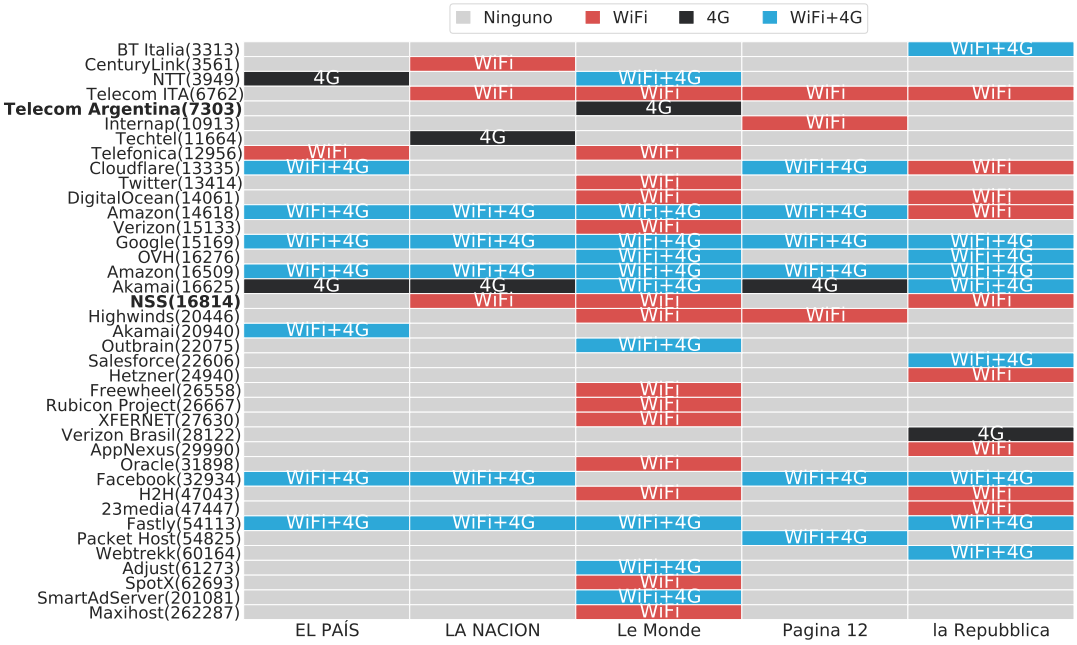


Figura 3.4: Presencia de cada AS por app, por tipo de red.

La Figura 3.4 presenta un mapa de calor, donde las columnas representan cada una de las aplicaciones de los diarios, y las filas indican las CDNs con las cuales se contactaron estas aplicaciones[1]. En esta figura se utilizan cuatro colores: gris (sin interacción), rojo (WiFi/iPlan), negro (4G/Personal) y azul (ambos) para indicar en qué experimento fue detectada la interacción de la App con la CDNs. En la figura podemos identificar un grupo de CDNs de notoria prevalencia, ya que están presentes para casi todas las aplicaciones, y en la mayoría de los casos fueron observadas en ambos experimentos, desde 4G y WiFi. Este grupo está integrado por Akamai (AS20940), Amazon (AS16509 y AS14618), Google (AS15169), Fastly (AS54113) y Facebook (AS32934). Notablemente, Facebook aparece muy frecuentemente aunque no brinde servicios *cloud*, lo que se debe a que estas aplicaciones cuentan con contenido embebido de la red de Facebook, como por ejemplo, login con Facebook, comentarios, etc. Otro aspecto relevante en el gráfico es la presencia en las filas de Telecom Argentina (AS7303) y NSS (AS16814), indicados en

[1]Por motivos de representación gráfica, sólo se incluyeron los ASes de las CDNs que contribuyeron con más del 0.5 % del tráfico para alguno de los dos modos

letra *negrita*. Estos son los ASes desde donde se realizaron los experimentos, y aparecen también como proveedores de contenido. Su presencia se debe a una técnica denominada como *in-network cache* [33, 34], donde los proveedores de Internet alojan en sus propias redes caches de las CDNs. De esta manera los ISPs y las CDNs reducen los costos por la distribución de contenido.

Capítulo 4

Trabajo futuro

A continuación se enumeran las principales direcciones que podrían abordarse para continuar este trabajo en las próximas etapas

1. **Centralización de datos**. El esquema de almacenamiento utilizado en este desarrollo genera que los datos se almacenen exclusivamente de forma local. Entendiendo la importancia de exportar la información, `svrmap` provee la función de exportar los datos a CSV. Sin embargo, este esquema de almacenamiento y distribución de datos es ineficiente para exploraciones de la red a gran escala. En trabajos futuros se podría implementar un sistema de recopilación centralizada de la información (conservando la anonimidad de los usuarios) para, por ejemplo, analizar patrones de comportamiento de las aplicaciones en base a la ubicación geográfica de sus usuarios. Este reconfiguración del sistema de almacenamiento daría una visión macroscópica de la infraestructura subyacente.

2. **Eliminación de dependencias**. En este trabajo se priorizó desarrollar una prueba de concepto generando un mínimo producto viable. Esta decisión de diseño llevó a optar por soluciones preexistentes cada vez que fuera posible, limitando esfuerzos en posibles mejoras de la aplicación. Generalmente, esto no presentó ningún problema, pero en el caso de realizar *calls* HTTP(S) a sistemas externos, sería una mejor alternativa implementar un stack propio, para poder proteger adecuadamente los sockets correspondientes a dichas *calls*. Por el momento el único sistema externo con el que se interactúa es la API REST de RIPE, pero en un futuro podría causar problemas (en el caso, por ejemplo, de implementarse el sistema centralizado mencionado anteriormente).

3. **Desarrollo para iOS**. De ser posible implementar `svrmap` en el sistema operativo móvil de Apple, iOS, se abarcaría casi la totalidad de dispositivos móviles de

hoy en día, por lo que dicha posibilidad resulta particularmente interesante como desarrollo futuro.

4. **Mejorar propuesta de valor para usuarios**. Aunque la utilidad de `svrmap` es principalmente orientada a tareas de investigación científica, esta aplicación podría ser expandida para el público en general. En este momento, para comprender la información recolectada y presentada se requiere de profundos fundamentos técnicos del funcionamiento de Internet. En tareas futuras se podrían generar esquemas y métricas que le permitan a los usuario comprender el desempeño de sus ISPs en la cadena de distribución de contenidos.

Capítulo 5

Conclusiones

En este trabajo profesional se construyó una prueba de concepto para el sistema `srvmap` con el objetivo de estudiar las estrategias de distribución de contenido de las aplicaciones de Android. Este trabajo permitió verificar la hipótesis de que es posible implementar en un dispositivo con Sistema Operativo Android una aplicación que en el mismo dispositivo determine los patrones de tráfico y las CDNs con las que se contacta cada aplicación. Esta facultad permitió estudiar la infraestructura subyacente de las aplicaciones que utilizamos diariamente, al menos en dispositivos que utilizan Android.

Consideramos que `srvmap` se presenta como una herramienta de alto impacto para la investigación científica, facilitando la recopilación de datos en dispositivos móviles. Este beneficio es sustancial ya que previo a `svrmap`, esta era una tarea difícil o incluso imposible. Particularmente, si se resuelve la falta de incentivos para usuarios comunes y se implementa un servicio de recopilación anónima de datos de utilización de los mismos, se podría crear una plataforma colaborativa. Esta tendría como objetivo brindar información del comportamiento a gran escala de aplicaciones móviles, lo cual podría potenciar la investigación en un área.

En conclusión, consideramos que el proyecto fue un éxito, con numerosas vías de desarrollo a futuro (como se discutió en el Capítulo 4), y que podría permitir un estudio de la infraestructura de la red desde el punto de vista de los dispositivos móviles con un nivel de profundidad difícilmente posible con herramientas preexistentes.

Bibliografía

[1] Union Internacional de Telecomunicaciones. Estadisticas. `https://www.itu.int/en/ITU-D/Statistics/Pages/stat/default.aspx`. 4

[2] Union Internacional de Telecomunicaciones. Measuring digital development: Facts and figures 2019. `https://www.itu.int/en/ITU-D/Statistics/Documents/facts/FactsFigures2019.pdf`. 5

[3] Stuart McIlroy, Nasir Ali, and Ahmed E Hassan. Fresh apps: an empirical study of frequently-updated mobile apps in the google play store. *Empirical Software Engineering*, 21(3):1346–1370, 2016. 5

[4] Matt Calder, Xun Fan, Zi Hu, Ethan Katz-Bassett, John Heidemann, and Ramesh Govindan. Mapping the expansion of google's serving infrastructure. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, page 313–326, New York, NY, USA, 2013. Association for Computing Machinery. 5

[5] Cheng Huang, Angela Wang, Jin Li, and Keith W Ross. Measuring and evaluating large-scale cdns. In *ACM IMC*, volume 8, pages 15–29, 2008. 5

[6] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The macroscopic behavior of the tcp congestion avoidance algorithm. *ACM SIGCOMM Computer Communication Review*, 27(3):67–82, 1997. 5

[7] Esteban Carisimo, Carlos Selmo, J Ignacio Alvarez-Hamelin, and Amogh Dhamdhere. Studying the evolution of content providers in ipv4 and ipv6 internet cores. *Computer Communications*, 145:54–65, 2019. 5, 6

[8] Esteban Carisimo, Carlos Selmo, J Ignacio Alvarez-Hamelin, and Amogh Dhamdhere. Studying the evolution of content providers in the internet core. In *2018 Network Traffic Measurement and Analysis Conference (TMA)*, pages 1–8. IEEE, 2018. 5, 6

[9] Amogh Dhamdhere and Constantine Dovrolis. The internet is flat: Modeling the transition from a transit hierarchy to a peering mesh. In *Proceedings of the 6th*

International COnference, Co-NEXT '10, New York, NY, USA, 2010. Association for Computing Machinery. 6

[10] Spotify. How spotify aligned cdn services for a lightning fast streaming experience. https://engineering.atspotify.com/2020/02/24/how-spotify-aligned-cdn-services-for-a-lightning-fast-streaming-experience/. 6

[11] Aqsa Kashaf, Vyas Sekar, and Yuvraj Agarwal. Analyzing third party service dependencies in modern web services: Have we learned from the mirai-dyn incident? In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, page 634–647, New York, NY, USA, 2020. Association for Computing Machinery. 6

[12] Rachee Singh, Arun Dunna, and Phillipa Gill. Characterizing the deployment and performance of multi-cdns. In *Proceedings of the Internet Measurement Conference 2018*, pages 168–174, 2018. 6

[13] Netflix. How netflix works with isps around the globe to deliver a great viewing experience. https://about.netflix.com/en/news/how-netflix-works-with-isps-around-the-globe-to-deliver-a-great-viewing-experien 6

[14] Christian Matt, Thomas Hess, and Alexander Benlian. Digital transformation strategies. *Business & information systems engineering*, 57(5):339–343, 2015. 7

[15] Carmen Cuesta, Macarena Ruesta, David Tuesta, Pablo Urbiola, et al. The digital transformation of the banking industry. *BBVA research*, pages 1–10, 2015. 7

[16] Ines Mergel, Noella Edelmann, and Nathalie Haug. Defining digital transformation: Results from expert interviews. *Government information quarterly*, 36(4):101385, 2019. 7

[17] European Union. Gdpr: User-friendly guide to general data protection regulation. https://www.gdpreu.org. 7

[18] State of California Department of Justice. California consumer privacy act (ccpa). https://oag.ca.gov/privacy/ccpa. 7

[19] Lei geral de proteÇÃo de dados (lgpd). https://www.lgpdbrasil.com.br. 7

[20] Costas Iordanou, Georgios Smaragdakis, Ingmar Poese, and Nikolaos Laoutaris. Tracing cross border web tracking. In *Proceedings of the Internet Measurement Conference 2018*, pages 329–342, 2018. 7

[21] Android security overview. `https://source.android.com/security`. 11

[22] Android features - application security. `https://source.android.com/security/app-sandbox`. 11

[23] Guía oficial de uso de VPNService. `https://developer.android.com/guide/topics/connectivity/vpn#service`. 11

[24] Matthew Luckie, Bradley Huffaker, and k claffy. Learning regexes to extract router names from hostnames. In *Proceedings of the Internet Measurement Conference*, pages 337–350, 2019. 12

[25] P. Hoffman and P. McManus. DNS Queries over HTTPS (DoH). RFC 8484, IETF, October 2018. 12

[26] Autonomous system numbers. `https://www.arin.net/resources/guide/asn`. 12

[27] Pcap4j. `https://www.pcap4j.org`. 16

[28] Fastcsv. `https://github.com/osiegmar/FastCSV`. 16

[29] Documentación oficial de la librería volley. `https://developer.android.com/training/volley`. 16

[30] Documentación oficial de la librería room. `https://developer.android.com/training/data-storage/room`. 16

[31] Documentación oficial de la RIPEstat API. `https://stat.ripe.net/docs/data_api`. 16

[32] Ao-Jan Su, David R Choffnes, Aleksandar Kuzmanovic, and Fabian E Bustamante. Drafting behind akamai: Inferring network conditions based on cdn redirections. *IEEE/ACM transactions on networking*, 17(6):1752–1765, 2009. 23

[33] Ioannis Psaras, Wei Koong Chai, and George Pavlou. Probabilistic in-network caching for information-centric networks. In *Proceedings of the second edition of the ICN workshop on Information-centric networking*, pages 55–60, 2012. 31

[34] Kideok Cho, Munyoung Lee, Kunwoo Park, Ted Taekyoung Kwon, Yanghee Choi, and Sangheon Pack. Wave: Popularity-based and collaborative in-network caching for content-oriented networks. In *2012 Proceedings IEEE INFOCOM Workshops*, pages 316–321. IEEE, 2012. 31