PePa Ping: Android Tool to Take and Predict Periodic Passive Ping Measurements

Diego Madariaga, Gabriela Mendoza
{diego, gabriela}@niclabs.cl

NIC Chile Research Labs
Universidad de Chile

Abstract

Global increase in the use of mobile Internet service generates interest in mobile network studies to determine and forecast the QoS provided by mobile operators. This study present a method to take passive ping measurements in Android devices and proposes different methods to forecast two of the most important Internet QoS indicators obtained by the passive ping method: RTT and percentage of packets lost in TCP connections, based on other passive in-smartphone measurements.

1. Introducción

Durante los últimos años, el uso del servicio de Internet móvil ha ido en permanente crecimiento a nivel global, lo que se refleja en Chile, en donde durante el primer trimestre de 2017, el 75.8 % del total de accesos a Internet, fue realizado por medio de dispositivos móviles [Sub17]. Dada la gran magnitud del uso actual de Internet móvil, es importante realizar estudios para determinar la calidad de este servicio entregado por los operadores de telefonía móvil, así como el desarrollo de métodos predictivos para la calidad de este servicio, que permitan anticipar posibles degradaciones en la calidad de servicio del Internet móvil en un tiempo determinado.

Dentro de los indicadores más utilizados para conocer la calidad de la conexión a Internet desde un dispositivo, destacan el tiempo de ida y vuelta (RTT), latencia, jitter, cantidad de paquetes perdidos y velocidad de subida y bajada de datos, los que típicamente se reportan en las pruebas de velocidad realizadas en sitios populares como *SpeedTest*[1] o *Fast*[2], donde estos indicadores son calculados en una conexión *end to end*, tomando medidas respecto al camino entre el dispositivo y el servidor de prueba, lo cual no representa necesariamente la calidad general del servicio, debido a que la prueba reporta el comportamiento de una única conexión de las múltiples que pueden presentarse de forma simultánea. Además, dichas pruebas son realizadas en un momento específico y capturan la calidad en ese instante, lo que dificulta registrar de forma constante la calidad de servicio del Internet durante un intervalo de tiempo mayor, ya que esto implicaría una permanente sobrecarga del dispositivo debido a las constantes pruebas, lo cual sería especialmente perjudicial en dispositivos móviles, ya que al tratarse de pruebas activas que introducen paquetes en la red, los planes de datos de Internet se verían fuertemente afectados, debido a que estos suelen ser limitados.

Por lo tanto, el trabajo en desarrollo presentado, muestra una herramienta desarrollada para dispositivos Android que permite de forma pasiva recolectar información importante acerca del estado de las conexiones TCP establecidas, capturando periódicamente valores de RTT y cantidad de paquetes perdidos. Además, se propone la realización de modelos de predicción basado en el análisis de series temporales y basados en la resolución de problemas de regresión que permitan estimar estos valores para un dispositivo en específico, tomando como base mediciones anteriores recolectadas y otros indicadores pasivos obtenidos, tales como intensidad de señal recibida y eventos de des-

In: Proceedings of the IV School o Systems and Networks (SSN 2018), Valdivia, Chile, October 29-31, 2018. Published at http://ceur-ws.org

[1] https://speedtest.net
[2] https://fast.com

conexión de la red.

2. Trabajo relacionado

Las mediciones de latencia, jitter y cantidad de paquetes perdidos suelen llevarse a cabo por medio de la herramienta de software conocida como *ping*, la cual realiza sus mediciones enviando paquetes ICMP (Internet Control Message Protocol) de tipo *echo request* y esperando la llegada de paquetes ICMP de tipo *echo reply* como respuesta. A pesar de su común uso, *ping* no reporta medidas precisas, ni refleja necesariamente la calidad de servicio recibida, principalmente debido a que se basa en el uso de paquetes ICMP (utilizados solo con fines de control y diagnóstico), y pueden ser afectados por diferentes mecanismos de control de tráfico que los paquetes TCP. Además, al igual que las pruebas de velocidad mencionadas anteriormente, *ping* reporta el estado de una única conexión y no el estado general de la conectividad a Internet.

Cómo solución a estos problemas, tanto operadores de redes como investigadores han adoptado métodos pasivos para medir la calidad de servicio presente en las conexiones de Internet establecidas, pudiendo analizar todos los flujos de datos sin una sobrecarga de tráfico que pueda afectar las mediciones tomadas, llevando a cabo estudios para estimar la latencia de las conexiones de Internet de forma pasiva, analizando los tiempos de salida y entrada de paquetes TCP [Jia02] o analizando la opción *timestamp* en las cabeceras TCP[3], siguiendo el mecanismo introducido en [Jac92]. El método de *ping* pasivo periódico presentado en este trabajo, difiere de los mencionados en el hecho de que reporta los valores de RTT y porcentaje de paquetes perdidos que el mismo sistema ha calculado, por medio del acceso a la información almacenada en los sockets de cada conexión. Además, la herramienta reporta cada intervalos de 10 segundos acerca de las conexiones activas durante ese intervalo de tiempo.

Con respecto a la predicción de indicadores de calidad en dispositivos móviles destaca la predicción de calidad de experiencia a partir de mediciones del estado de la red [Agg14] y a partir de mediciones dentro de los mismos dispositivos [Cas17], utilizando técnicas propias del aprendizaje de máquinas como análisis de series temporales y la implementación de clasificadores. De este mismo modo, se han implementado métodos para predecir, a partir del desempeño de la red, valores de calidad de experiencia para *streaming* de video [Ket10] y para Voz sobre protocolo de Internet (VoIP) [Cha16]. La principal diferencia del trabajo presentado, es que propone la obtención y predicción de indicadores de calidad de servicio y no de calidad de experiencia, por medio del uso de mediciones obtenidas dentro del mismo dispositivo móvil.

[3]http://www.pollere.net/pping.html

3. Aplicación Android PePa Ping

Dado que la gran mayoría de las soluciones para estimar tiempo de ida y vuelta (RTT) y cantidad de paquetes perdidos se basan en la interceptación de interfaces de red y en el acceso al contenido de los *headers* IP y TCP de cada paquete, sus implementaciones en entornos Linux requieren permisos de superusuario (*root*), lo que dificulta su implementación en dispositivos Android (basados en Linux) ya que para acceder a dichos permisos, el usuario debe realizar un proceso de *rooteado* del teléfono, interviniendo el estado de fábrica del dispositivo, perdiendo su garantía y pudiendo incluso quedar inutilizable en caso de alguna falla.

Ya que para realizar análisis sobre las conexiones a Internet es indispensable el acceso al tráfico de red, y considerando las limitaciones mencionadas previamente para dispositivos Android, durante este trabajo se desarrolló la aplicación PePa Ping que accede al tráfico de Internet de una forma distinta, haciendo uso de la API incluida en la versión 4.0 de Android para establecer una conexión con una red privada virtual[4]. A través de este método, se crea una interfaz de red virtual y se provee a la aplicación de un descriptor de archivo, donde cada lectura entrega un paquete IP que va desde el dispositivo hacia la red, y cada escritura introduce un paquete IP como proveniente del exterior. En un uso tradicional, la aplicación se conecta mediante un túnel VPN a un servidor VPN y le envía los paquetes leídos desde el descriptor de archivos. Posteriormente, los paquetes enviados desde el servidor VPN son leídos del túnel VPN y escritos en el descriptor de archivos, completando así el flujo de paquetes IP.

La herramienta desarrollada, utiliza el servicio de VPN de una manera diferente, sin enviar los paquetes IP leídos hacia un servidor VPN exterior, sino que manejándolos en el mismo celular, en un servidor VPN interno, cuyo funcionamiento se ilustra en la Figura 1.

Este servidor VPN interno se encuentra programado en el lenguaje C++, y es introducido en el código de la aplicación Android gracias al framework Java Native Interface, que permite la interacción entre código Java y código escrito en C, C++ o assembler. Así, el servidor interno recibe los paquetes IP y envía

[4]https://developer.android.com/reference/android/net/VpnService

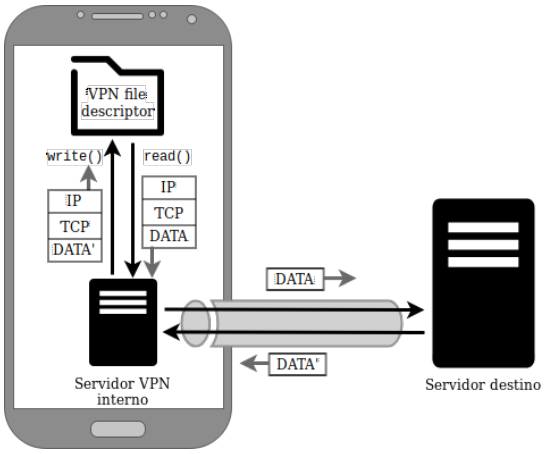

Figura 1: Diagrama de flujo de paquetes utilizando servidor VPN interno.

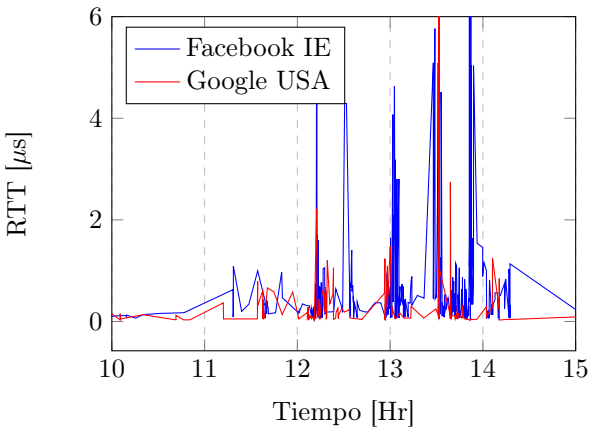

Figura 2: RTT registrado por la aplicación PePa Ping durante 5 horas

sus datos por sockets de tipo `SOCK_DGRAM` (UDP) y `SOCK_STREAM` (TCP), que reciben directamente los datos a enviar y se encargan de la encapsulación de estos en headers TCP/UDP e IP. Asimismo, al leer de estos sockets, no se obtienen paquetes IP, sino solo el área de datos de estos (*payload*) y ya que el descriptor de archivos de la red virtual necesita que en él se escriban paquetes IP, el servidor interno coloca los headers correspondientes sobre los datos leídos desde los sockets antes de escribirlos en el descriptor de archivo. Lo anterior implica que para las conexiones TCP, el servidor interno se encargue de manejar, entre otras cosas, el registro de los bytes enviados por el socket, con el fin de que los headers TCP puestos sobre los datos leídos desde el socket presenten el valor correcto en el campo ACK.

Es importante mencionar que, a diferencia de Java, C permite la creación de sockets en donde se provee una comunicación a nivel de protocolo de red, pudiendo enviar y recibir directamente paquetes IP. Sin embargo, este método requiere la creación de sockets del tipo `SOCKET_RAW`[5], lo que requiere que la aplicación sea ejecutada con permisos de superusuario, lo que es una dificultad mayor en dispositivos Android, como se explicó con anterioridad en la sección 2.

La aplicación mantiene dos tablas de hash, para almacenar las conexiones TCP y UDP, cuyas *llaves* corresponden a la unión de los valores `puerto_origen`, `ip_destino` y `puerto_destino`, y cuyos *valores* son objetos de la clase `TcpConnection` implementada, que almacena al socket conectado entre el dispositivo y la dirección `ip_destino:puerto_destino`. Además, se utiliza un método de sondeo (*polling*) para organizar la lectura de los sockets, mediante la llamada de sistema `epoll`, en donde los descriptores de archivo de los sockets son registrados para que se levante un evento cada vez que alguno esté disponible para ser leído (cuando se tenga respuesta a los datos enviados anteriormente).

PePa Ping, registra cada 10 segundos el RTT y la cantidad de paquetes perdidos para cada uno de los sockets TCP activos durante los 10 segundos previos, accediendo a dichos valores calculados por el sistema, mediante la opción de socket `TCP_INFO`, realizando así mediciones pasivas de RTT y cantidad de paquetes perdidos de forma periódica. Estas mediciones son pasivas en cuanto no se añaden paquetes nuevos a la red sino que los paquetes son enviados por medio de los sockets creados por el servidor VPN interno.

4. Análisis preliminar y trabajo futuro

La Figura 2 muestra el RTT reportado por la aplicación PePa Ping en un dispositivo de prueba en uso constante de Internet móvil, respecto a conexiones TCP establecidas durante las 10:00 y 15:00 horas de un día laboral hacia servidores de Google en California, Estados Unidos y hacia servidores de Facebook en Dublín, Irlanda, en donde se observa que las mediciones de RTT presentan un comportamiento que varía en función de al menos dos variables: tiempo y ubicación geográfica del servidor de destino.

En adición a estas variables, se plantea la consideración de otras medidas pasivas tomadas por el dispositivo tales como intensidad de señal recibida, episodios de desconexión de la red y posición geográfica (datos disponibles por medio de aplicaciones como *Adkintun Mobile* [Bus13]), con el fin de utilizar estrategias de aprendizaje de máquinas enfocadas

[5] http://man7.org/linux/man-pages/man2/socket.2.html

tanto en el análisis de series temporales como en la resolución de problemas de regresión, para predecir a partir de las mediciones mencionadas tomadas dentro del dispositivo móvil, valores de RTT y porcentaje de paquetes perdidos en las conexiones TCP establecidas, de forma similar a lo realizado en trabajos para predecir la calidad de experiencia en dispositivos móviles en tiempo real [Cas17, Agg14].

Además, se considera como trabajo futuro la extensión de la aplicación desarrollada con el fin de entregar al usuario información en tiempo real de las conexiones establecidas por cada una de las aplicaciones instaladas. Para esto, es necesario agrupar las conexiones pertenecientes a una misma aplicación y calcular datos representativos de la calidad de servicio obtenida por dicha aplicación, con lo que sería posible darle a los usuarios una herramienta útil para conocer la calidad del Internet recibido.

Referencias

[Sub17] Subsecretaría de Telecomunicaciones. Penetración de Internet marca alza del 25 % y llega a los 16,7 millones de accesos. Online de http://www.subtel.gob.cl/penetracion-de-internet-marca-alza-del-25-y-llega-a-los-167-millones-de-accesos, September, 2017

[Agg14] V. Aggarwal, et al. Prometheus: Toward quality-of-experience estimation for mobile apps from passive network measurements. *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*. ACM, 2014. p. 18.

[Cas17] P. Casas, et al. Predicting QoE in cellular networks using machine learning and in-smartphone measurements. *Quality of Multimedia Experience (QoMEX), 2017 Ninth International Conference on*. IEEE, 2017. p. 1-6.

[Ket10] I. Ketykó, et al. QoE measurement of mobile YouTube video streaming. *Proceedings of the 3rd workshop on Mobile video delivery*. ACM, 2010. p. 27-32.

[Cha16] P. Charonyktakis, et al. On User-Centric Modular QoE Prediction for VoIP Based on Machine-Learning Algorithms. *IEEE Transactions on mobile computing 15*, no. 6, p. 1443-1456, 2016.

[Jia02] H. Jiang, et al. Passive estimation of TCP round-trip times. *ACM SIGCOMM Computer Communication Review*, 2002, vol. 32, no 3, p. 75-88.

[Jac92] V. Jacobson , R. Braden and D. Borman. TCP Extensions for High Performance, RFC 1323. *RFC Editor*, 1992.

[Bus13] J. Bustos-Jiménez, et al. How adkintunmobile measured the world. *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*. ACM, 2013. p. 1457-1462.