SGF: A Crowdsourced Large-scale Event Dataset

Jens Heuschkel
TK / TU Darmstadt
Darmstadt, Germany
heuschkel@tk.tu-darmstadt.de

Alexander Frömmgen
KOM / TU Darmstadt
Darmstadt, Germany
alexander.froemmgen@kom.tu-darmstadt.de

ABSTRACT

This paper presents a crowdsourced dataset of a large-scale event with more than 1000 measuring participants. The detailed dataset consists of various location data and network measurements of all national carrier collected during a four-day event. The concentrated samples for this short time period enable detailed analysis, e.g., by correlating movement patterns and experienced network conditions.

CCS CONCEPTS

• **Networks** → **Network measurement**;

KEYWORDS

crowdsourcing, dataset, large-scale event

ACM Reference Format:

Jens Heuschkel and Alexander Frömmgen. 2018. SGF: A Crowdsourced Large-scale Event Dataset. In *MMSys'18: 9th ACM Multimedia Systems Conference, June 12–15, 2018, Amsterdam, Netherlands*. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3204949.3208120>

1 INTRODUCTION

Today, most people use one or even multiple mobile devices with an Internet connection [1]. This trend changed our communication behavior towards Internet based services like WhatsApp. Internet service providers (ISP) react to the increased demand and scale their systems to serve the daily needs of their customers. At big events, however, when thousands of people are in a small area, people report on experienced performance degradations of their Internet connection.

Existing datasets from large scale events are typically concentrated on a single carrier or passive measurements of the users in the network infrastructure. Erman et al. [3] analyzed the Super Bowl from AT&T's perspective and provided a detailed analysis of the performance and the user behavior. Shafiq et al. [6] describe provider observations of two crowded events. They present low layer metrics, but provide only limited insights into the actual performance available at the end device. Additionally, they are restricted to the perspective of a few network operators.

We wanted to investigate such a large-scale event in detail. A comprehensive dataset of large-scale events enables various analysis, e.g., of the experienced internet connection performance and the dependent applications, such as instant messaging, web performance

and multimedia systems. For a detailed analysis and understanding how infrastructure behaves and how people experience a potential performance degradation, a large dataset is required. For getting a wide spread over the existing ISP, access technologies and mobile devices, we used a crowdsourcing application to measure with the participants personal mobile devices. Nikraves et al. [5] used crowdsourced measurements with samples from all over the world to provide valuable insights into the general network performance. They showed that crowdsourcing is a reasonable tool to get large data sets. As another example, Xu et al. [7] used crowdsourcing to investigate a cellular network in Singapore in daily life situations. In this paper, we focus on the dataset of a large-scale event, as partly analysed in [4].

The remainder of this paper is structured as follows: Section 2 presents the measurement app, the venue, the data anonymization efforts and the limitations of the data. Then, we present the dataset and the used measurement method structured by location (Section 3), network performance (Section 4), and other measurements (Section 5). Section 6 discusses illustrating examples for using the dataset. Finally, Section 7 concludes the paper.

2 MEASUREMENT SETUP

In the following, we present the general considerations of the measurement process:

Venue: Schlossgrabenfest. For the measurements, we chose a local music festival named Schlossgrabenfest¹. It is the biggest music festival in the state of Hesse in Germany. During the festival, the city center of Darmstadt, a city with approximately 150000 inhabitants, is highly crowded. The event takes four days, has more than 110000 visitors per day and is restricted to a specific area with entrance control. Figure 1 shows the area of the festival with purple borders. Since many young people stay at the outside of the area in a park, celebrating their own party, we included this area in our measurement (shown by the red borders in Figure 1). The area of interest is bordered by 49.872740°N and 49.880131°N latitude as well as 8.650635°E and 8.657088°E longitude.

Application: Research4Refill. For the measurements, we developed an application for the Android platform (version 4.0) and distributed it over the Google Play Store. As we were focusing on a very large dataset with as many participants as possible, we limited the amount of collected data to preserve privacy and avoid scaring potential users. The app tracks the location (based on GPS, network signals, and signals of additional placed Bluetooth Beacons), the WiFi and cellular network state. It actively executes HTTP, DNS, and traceroute measurements. Additionally, we asked all users to fill

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

MMSys'18, June 12–15, 2018, Amsterdam, Netherlands

© 2018 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-5192-8/18/06.

<https://doi.org/10.1145/3204949.3208120>

¹The official website (in German): <http://www.schlossgrabenfest.de/2015/>

Figure 1: Map showing the festival area (purple) and the measurement area (red)²

in a short questionnaire. Thus, the app only requires minimal rights and does not track the actual transferred user data.

We advertised the application in local print media, social media (e.g., Facebook), the university campus and on a video screen during the festival. We appeal the people to help us understanding their network problems by running the application and contribute their data. An additional incentive was a free drink we offered for users who ran the app for a at least four hours. We found more than 1000 users willing to give up some privacy for a free beer.

Anonymization. Each data point can be assigned to a unique device. Each device has a randomly generated UUID as identification without any correlation to the international mobile equipment identity (IMEI). We thought about changing the id for the devices every day, but we believe this will not increase the privacy of the individuals, but would hinder the performance data correlation over more than one day. The GPS traces as well as cellular network station details are restricted to the measurement area (shown by the red

²Source of the map: <https://maps.wikimedia.org/#17/49.87471/8.65393>

Table 1: location

accuracy	accuracy of the data point in meter
bearing	bearing of a data point in degrees or 0 if there was no compass available ³
latitude	latitude in degree
logitude	longitude in degree
provider	location provider as one of the strings 'network', 'gps' or 'fused'

borders in Figure 1) to protect the privacy of the attendances. The shown area includes some university but no residential buildings to ensure anonymity. With this pieces in place, we ensure that no individual person can be traced to their home. On the other hand, it's possible to join the data with each other and find the source device to every data point, enabling comprehensive data analysis

Format. We provide the data as SQL dump export file at [2]. The file is tested to be importable into a MySQL Community Server version 5.6.27. After importing the data, the server provides the following 11 tables:

(1) `location` containing the location data measures by the location provided by Android, (2) `bluetooth` containing the bluetooth measurements, (3) `tcp_bandwidth`, and (4) `udp_bandwidth` containing the bandwidth measurements, (5) `http_loadtime` containing the http loadtime measurements, (6) `dns` containing the dns request measurements, (7) `tracerout`, and (8) `tracerout_hops` containing the tracerout measurements, (9) `traffic_stats` containing traffic information provided by Android, (10) `device_info` containing device information, and (11) `battery` containing battery information.

Every database table has an `id` field containing a long integer as primary key. Most database tables have a `deviceId` field, identifying a single mobile device from the database table `device_info`. Also most database tables have a timestamp field, containing the date time of the measured value. We present the structure of every table and explain the meaning of the columns in the respective sections.

Limitations. We assume that most participants are tech-savvy and students. The app might influence the participants, as they stay longer to gain a free drink, or only install the app if they plan to stay long. Additionally, the battery was drained much faster than in normal operation. Many users didn't consider this fact and ran out of battery early at the evening or shut down the measurements.

3 LOCATION MEASUREMENTS

In this section, we present our location measurements. We collected 1196875 location measurements during the event. As 276 out of 1401 devices only returned network locations. We assume that these users manually turned off GPS. Due to privacy concerns, we removed samples outside the area of interest, as presented in section 2 and shown in fig. 1

Network & GPS. For measuring the location data, we used the location pro-vider API of Android. We did not only save the location, but also the accuracy, bearing and the used location provider. There

³Note that a proper initialized compass is unlikely

are three types of location providers represented in our measurements: (1) The GPS provider provides data from the GPS system. (2) The network provider provides location data based on the seen cellular networks and WiFi access points. (3) The fused provider provides diffuse location information, where we have no further information where the location comes from. The measurements are located in the database table `location` with the fields presented in table 1.

Bluetooth Beacons. Beside the GPS and network location provider, the app regularly searches for Bluetooth Beacons. We placed 50 Bluetooth Low Energy Beacons⁴ at the venue. In the database table `beacons` we listed the beacons deployed by us with respective locations in the area. We were surprised that most people turned off Bluetooth immediately and complained about this measurement. For reading the beacon properties, we used the open source library *AltBeacon*⁵ recommended by the beacon manufacturer. The measurements are located in the table `bluetooth` with the fields presented in table 2.

Table 2: `bluetooth`

<code>bluetoothAddress</code>	hardware address of the placed beacon
<code>bluetoothName</code>	visible name of the placed beacon ⁶
<code>distance</code>	distance to the beacon calculated by the library
<code>id1</code>	first identifier ⁷
<code>id2</code>	second identifier
<code>id3</code>	third identifier
<code>manufacturer</code>	two byte integer with an id ⁸
<code>rsi</code>	signal strength
<code>txPower</code>	transmitted power calibration value

4 NETWORK MEASUREMENTS

In this section, we present our main part of the measurements. We measured several network characteristics leveraging the Android API. The results are structured in the following five paragraphs: Bandwidth, HTTP Loadtime, DNS Request, Traceroute and Traffic Stats.

Bandwidth. During the event, we actively measured the available bandwidth of the devices. For one measurement cycle, we measured first the upload and download rate with TCP. Also we measured the latency (round-trip time) for a four byte packet. Right after that, we measured the upload channel with UDP. For measuring the throughput, we uploaded respectively downloaded a

randomly generated data blob and took the time provided by `System.nanoTime()`⁹. The TCP measurements are located in the database table `tcp_bandwidth` with the fields presented in table 3. The UDP measurements are located in the database table `udp_bandwidth` with the fields presented in table 4.

Table 3: `tcp_bandwidth`

<code>errorMsg</code>	error message (if any)
<code>exception</code>	exception type (if any)
<code>fourBytesLatency</code>	measured latency for four bytes in nano seconds
<code>localport</code>	local port number to which this socket is bound ¹⁰
<code>port</code>	target port
<code>openSocket</code>	time the system needed to create the Socket object ¹¹
<code>randomValueRead</code>	random value generated for the latency measurement
<code>randomValueSend</code>	random value received from the server
<code>receiving</code>	results of the download throughput measurement ¹²
<code>sending</code>	result of the upload throughput measurement ¹³

Table 4: `udp_bandwidth`

<code>udpId</code>	randomly generated id for one measuring flow
<code>errorMsg</code>	error message if an exception occurs
<code>time</code>	measured time for sending 50 kb randomly generated data in nanoseconds

HTTP Loadtime. We regularly measured the time for HTTP GET requests. As targets, we used a pool of popular German localized web pages (e.g., `google.de` or `facebook.de`), where a sufficient load balancing is ensured. We measured how long it takes to open the connection¹⁴ and the load completion time, but canceled the request after 30 seconds. The HTTP loadtime measurements are located in the database table `http_loadtime` with the fields presented in table 5.

⁹ [https://developer.android.com/reference/java/lang/System.html#nanoTime\(\)](https://developer.android.com/reference/java/lang/System.html#nanoTime())

¹⁰ Sometimes the vendor implementation returns 0 for this function.

¹¹ <https://developer.android.com/reference/java/net/Socket.html>

¹² The receiving measurement results are structured as follows:
(totalTransferredSize);(totalTime);(kbps);
list of ((time for chunk);(chunk size));

¹³ The sending measurement results are structured as follows:
(totalTransferredSize);(firstReceivedByte);
(timestampBeginReceiving);
(timestampAfterFirstByte);
(timestampEndReceiving);(kbps);
list of ((time for chunk);(chunk size));

¹⁴ [https://developer.android.com/reference/java/net/URLConnection.html#openConnection\(\)](https://developer.android.com/reference/java/net/URLConnection.html#openConnection())

⁴ <http://www.beaconinside.com/>

⁵ <https://github.com/AltBeacon/android-beacon-library>

⁶ This can be matched to the database table `beacons`

⁷ As far as we know, all three parts are needed for a distinct identification.

⁸ A list of registered manufacturer codes may be found at: <https://www.bluetooth.org/en-us/specification/assigned-numbers/company-identifiers>

Table 5: http_loadtime

errorMsg	exception type (if any)
isIOException	boolean indicating if the occurred exception was an IOException
tillFinished	time ⁹ until the page is completely loaded including the connection
tillOpenCon	time ⁹ until a connection to the server is established
url	target URL of the request
headerString	returned HTTP header as string

Table 6: dns

rwUrl	real world URL used for the request
rwIp	resolved IPs for the real world URL
rwTime	needed time for the real world URL DNS request, including connection establishment ⁹
rndUrl	random generated URL used for the request
rndIp	resolved IP for the random generated URL (if any)
rndTime	needed time for the random generated URL DNS request, including connection establishment ⁹

DNS Request. We regularly triggered DNS requests and measured the time for getting an answer from the DNS Server. Since DNS is an often used resource in current networks, most ISPs provide a caching infrastructure in the edge networks and also the Android operating system provides a DNS cache. For the measurements, we use a pool of popular German localized web pages and we assumed that most DNS requests we make are handled by this caching infrastructure. To overcome this we requested randomly generated non-existing URLs, enforcing a cache bypass. We noted that some ISPs answered with IPs from navigation help pages of ISPs (e.g., 62.157.140.133). The results are stored in the database table `dns_measurement` with the fields presented in table 6.

Traceroute. Since the traceroute tool is not available on unrooted Android platforms, we decide to use ICMP echo with an increasing TTL for every packet to emulate traceroute. We are aware that it's possible that every ICMP package could take a different route. This enables analysis how often different routes are taken by packets. As far as we see, this is a very rare effect and packets take surprisingly often the same route. As traceroute targets we had a pool of popular German localized web pages and picked one randomly from there. The traceroute measurements are located in the database table `traceroute` with the fields presented in table 7. In a second database table `traceroute_hops`, we store the single hops for every trace. The fields are presented in table 8.

Traffic Stats. We read the traffic stats provided by the Android API¹⁵. These data points are passive measurements done by the Android operating system. It mainly provides the received and sent

¹⁵<https://developer.android.com/reference/android/net/TrafficStats.html>

Table 7: traceroute

targetIp	IP for the picked target URL
targetUrl	picked target URL
rawData	complete console output of the ping tool

data amount. The measurements are located in the database table `traffic_stats` with the fields presented in table 9.

5 OTHER MEASUREMENTS

Along with the network measurements, we saved some data about the devices that are used for the measurements, leveraging the Android API. Additionally we asked the user to fill out a short survey.

Device Information. At the first start of our measurement application, we generated a random UUID for each device. Using this UUID the device registered with some context information about the device hardware and software provided by the Android API¹⁷. The database table `device_info` is organized according to the Android API calls. The fields are presented in table 10.

Battery Measurements. We regularly measured battery stats, like the current and total capacity to estimate our energy consumption. We used the Android API¹⁹ for reading these values. The database table `battery` is structured as presented in table 11.

User Survey. We ask the participants to optionally fill out a short survey about themselves. We asked them about their age, gender and if they would pay – and if yes how much – for a better internet connection at the event. The database table `user_survey` is structured as presented in table 12.

6 EXAMPLES

In [4], we used the presented dataset to analyze network load characteristics at a large-scale event. We analyzed the movement patterns of the users, showing constant movement even during the crowded times. Our additionally used Bluetooth beacon based location service allowed us to even trace users who did not provide GPS locations. We show that there is a significant performance degradation during the festival regarding DNS and HTTP failures as well as load times. Illustrating this, Figure 2 shows the average HTTP load times and

¹⁶since device boot

¹⁷<https://developer.android.com/reference/android/os/Build.html>

¹⁸It's not documented by the API reference, but we assume its the build time of the system image.

¹⁹<https://developer.android.com/reference/android/os/BatteryManager.html>

Table 8: traceroute_hops

traceId	primary key of the traceroute table identifying a single trace
hopId	hop index (0 based) of one trace
ip	IP of the hop
url	URL (if any) of the current hop, generated by DNS reverse lookup

Table 9: traffic_stats

mobileRxBytes	number of bytes received across mobile networks ¹⁶
mobileRxPackets	number of packets received across mobile networks ¹⁶
mobileTxBytes	number of bytes transmitted across mobile networks ¹⁶
mobileTxPackets	number of packets transmitted across mobile networks ¹⁶
totalRxBytes	number of bytes received ¹⁶
totalRxPackets	number of packets received ¹⁶
totalTxBytes	number of bytes transmitted ¹⁶
totalTxPackets	number of packets transmitted ¹⁶

Table 10: device_info

board	name of the underlying board, like "hammerhead" for the Nexus 5
bootloader	system bootloader version number
brand	consumer-visible brand with the hardware will be associated
device	name of the industrial design
display	build ID string meant for displaying to the user
hardware	name of the hardware from the kernel command line or /proc
manufacturer	manufacturer of the hardware
model	end-user-visible name for the end product
osVersion	system property "os.version" provided by the Dalvik VM
product	name of the overall product
radioVersion	version of the radio firmware
time	build time ¹⁸
uuid	randomly generated UUID
version	user-visible SDK version of the framework

the number of successful and failed requests for one day. The impact of the event between 8 pm and 11 pm is apparent, since the failed requests increase dramatically and the load times are almost doubled.

The detailed investigation of the network performance shows that the performance degradation differs between network operators, network types, and locations. We found that the carriers with a low failure rate during the event had a higher average load time. Furthermore, we sketched an evaluation of the impact of the carrier,

Table 11: battery

capacity	remaining battery capacity as integer percentage of total capacity
totalCapacity	total battery capacity in μAh

Table 12: user_survey

age	the filled in age of the participant
gender	the filled in gender of the participant
payment	the filled in willingness to pay for a better internet connection in Euro

Figure 2: HTTP measurements during the event.

the access technology, and the user's mobility on the perceived performance. The public release of the dataset enables the research community to conduct more detailed analysis and use our results as foundation for the evaluation of future research.

7 CONCLUSION

In this paper, we discussed our crowdsourced measurements at a large-scale event. We measured location (GPS, network, and additional placed Bluetooth Beacons), the WiFi and cellular network state, and executed HTTP, DNS, and traceroute measurements. Also, we provide detailed documentation to understand the data and the storage structure.

Our measurement study shows that large crowdsourcing measurements are feasible. We convinced more than 1000 users to participate in our study. We noticed that people liked the idea to help us to understand their performance issues. A small fraction of participants complained (e.g., in the Play Store) about the increased data transmission and higher energy consumption due to the app. Our measurement setup does not allow us to distinguish between the induced energy consumption of our app and a potentially increased consumption due to the overloaded network.

For the future, we hope that the community will be able to use the data for additional in-detail analysis of certain aspects, e.g. to retrieve complex movement models, dependencies between the signal strength and the people density, or evaluate new technologies to deal with crowded events.

ACKNOWLEDGMENT

This work has been funded by the German Research Foundation (DFG) as part of project B2 and C2 within the Collaborative Research Center (CRC) 1053 – MAKI.

REFERENCES

- [1] Cisco visual networking index: Global mobile data traffic forecast update 2015–2020 white paper. <http://www.cisco.com/c/en/us/solutions/collateral/servi>

- ce-provider/visual-networking-index-vni/white_paper_c11-520862.html, 2016.
- [2] Sgf: A crowdsourced large-scale event dataset. In *url-https://doi.org/10.5281/zenodo.1220903*. Zenodo, Apr. 2018.
 - [3] J. Erman and K. K. Ramakrishnan. Understanding the super-sized traffic of the super bowl. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 353–360. ACM, 2013.
 - [4] A. Frömmgen, J. Heuschkel, P. Jahnke, F. Cuozzo, I. Schweizer, P. Eugster, M. Mühlhäuser, and A. Buchmann. Crowdsourcing measurements of mobile network performance and mobility during a large scale event. In *International Conference on Passive and Active Network Measurement*, pages 70–82. Springer, 2016.
 - [5] A. Nikraves, D. R. Choffnes, E. Katz-Bassett, Z. M. Mao, and M. Welsh. Mobile network performance from user devices: A longitudinal, multidimensional analysis. In *Passive and Active Measurement*, pages 12–22. Springer, 2014.
 - [6] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, S. Venkataraman, and J. Wang. A first look at cellular network performance during crowded events. In *ACM SIGMETRICS Performance Evaluation Review*, volume 41, pages 17–28. ACM, 2013.
 - [7] Y. Xu, Z. Wang, W. K. Leong, and B. Leong. An end-to-end measurement study of modern cellular data networks. In *Passive and Active Measurement*, pages 34–45. Springer, 2014.