Modeling Realistic Bit Rates of D2D Communications between Android Devices

Clément Bertier, Marcelo Dias de Amorim, Farid Benbadis, Vania Conan

Modeling Realistic Bit Rates of D2D Communications between Android Devices

Clément Bertier[★,◇], Marcelo Dias de Amorim[★], Farid Benbadis[◇], and Vania Conan[◇]

[★]LIP6/CNRS – Sorbonne Université [◇]Thales SIX GTS France

ABSTRACT

Although D2D communications have been extensively investigated in the literature, relatively few works have focused on understanding the capacity of direct links in a real setup. In this paper, we propose an empirical characterization of the currently available high-speed D2D technologies in Android, namely *Wi-Fi P2P* and *Google Nearby*. To this end, we developed a custom Android application called *Ocat* which interacts with the available D2D APIs and measures the link's goodput. From the experimental campaign, we derive several useful observations. Concerning communication capacity, the goodput between Android devices ranges between 320 Mbits/s when nodes are within 20 meters of each other and 0.1 Mbits/s when the distance grows to 300 meters. Based on the experimental measurements, we propose a model of the upper-bound goodput as a function of the distance between two devices. Using the wireless signal strength as a link measurement, we combine it with the two-ray ground-reflection model to infer the goodput and obtain a good fit for the characterization of D2D links between Android devices. Our findings provide a reality check in regards to actual direct data-exchange capabilities of Android devices and can help assess system performance of D2D applications.

CCS CONCEPTS

• **Networks → Network performance modeling**; **Network performance analysis**.

KEYWORDS

D2D, model, experimentation, Android, measurements, RSSI.

ACM Reference Format:

Clément Bertier[★,◇], Marcelo Dias de Amorim[★], Farid Benbadis[◇], and Vania Conan[◇]. 2019. Modeling Realistic Bit Rates of D2D Communications between Android Devices. In *Proceedings of The 22nd ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'19).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Albeit significant advances in the design of protocols and algorithms for device-to-device (D2D) communications, the research community still lacks experimental works that focus on finely understanding D2D links *in real setups*. Several fundamental questions concerning the expected performance of direct links remain open: "*what is the expected transfer rate between two smartphones if they are 25m apart?*" or "*how far can these two smartphones communicate reliably?*". As a result, there are gaps in the state of the art when it comes to modeling D2D exchanges. For example, when simulating D2D communications in the *The ONE* simulator, the user has to select a maximum communication range as well as a constant throughput between the devices. Due to the scarcity of studies quantifying these parameters, authors often adopt rough estimates for the physical layer bit rate used by the wireless medium [24].

In this paper, we characterize the link quality of device-to-device communications based on empirical measurements of Wi-Fi Direct. We model the upper-bound of D2D bit rate as a function of the distance, thus shedding light on the applicability of D2D solutions. We expect to help protocol and algorithm designers better select their parameters in their simulations.

Studies tackling the performance of Wi-Fi links in infrastructure mode exist [20]; however, these studies differ from ours in several ways. Firstly, D2D links differ from traditional infrastructured communications because of hardware differences. By considering off-the-shelf smartphones using the latest OS updates (Android 8+ at the time of the writing of this paper), we provide up-to-date results. Secondly, to the best of our knowledge, this is the first characterization of a D2D link based on Google's Nearby Connections API. Thirdly, our results bring a number of new insightful observations for smartphone-based direct communications which notably differ from previous works in the literature. In a nutshell, we explore the state of current tools which enable high-speed D2D communications in Android and establish a model of the upper-bound bit rate using off-the-shelf devices.

As a summary, our contributions are:

- We explore the currently available high-speed D2D APIs in stock Android. As some APIs are proprietary, their inner-workings are not always disclosed, which requires indirect analysis to assess their behaviors.
- We design Ocat, a measurement application that stores data transfer information for post-processing. We detail our experimental procedure to collect data by using Ocat on Android smartphones considering both Google Nearby and Wi-Fi P2P APIs. We vary the distance between the devices up to the distance beyond which the devices lose connectivity (around 300 m in our experiments).
- We propose a model, after a thorough analysis of the collected data, to estimate the upper-bound of the goodput between two devices as a function of the distance.

This work aims to be a stepping-stone on how to accurately model D2D communications between Android devices in real-life.

MSWiM'19, Nov 25–29 2019, Miami Beach, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

While there are still limited efforts to leverage D2D communications as an applicable data-exchange solution [7], we are hopeful results provided in this work entice the community to do so.

The rest of this paper is organized as follows. In Section 2, we explain the different available APIs, while in Section 3 we introduce Ocat, our measurement application for Android devices. We detail the experimental work in Section 4; in particular, we provide a thorough analysis of the RSSI measurements in D2D links and how to model them accurately. In Section 5, we give a reality-check based on our goodput analysis and how we perform the goodput-to-distance modeling. We postpone the related work to Section 6, so that the reader has enough material to best capture our contributions. Finally, we conclude the paper and identify ideas for future work in Section 7.

2 STOCK ANDROID HIGH-SPEED D2D APIS

To establish the upper-bound of D2D bit rate according to distance, we carry out field measurements. To this end, we first take a look at currently available D2D APIs in Android. As of now, there are two supported APIs for high-speed D2D in stock Android, namely Wi-Fi P2P [11] and Google Nearby Connections [10]. The former is an implementation of the Wi-Fi Direct standard [3], while the latter is a closed-source framework to send files, strings (e.g., URLs), or even data streams (e.g., VoIP) to surrounding devices.

2.1 Wi-Fi P2P

Wi-Fi P2P is the Android implementation of Wi-Fi Alliance's Wi-Fi Direct standard. In short, peer-to-peer (P2P) devices communicate through *P2P Groups*, which are dynamically formed by electing a device as the P2P Group Owner, which embodies the role of the access point. A P2P device can concurrently act as a P2P Group Owner and as a P2P Client of another group.

While the standard's inner-workings have been thoroughly reviewed in the literature [3], the Android implementation reveals technical restrictions. For instance, while the standard does not state any limitations in the number of clients in a P2P group, we were not able to maintain more than one Wi-Fi P2P connection using this API. Once two P2P devices have detected each other, one (or both) can initiate a Wi-Fi P2P connection, as shown in Figure 1. This connection needs to be approved by the user(s) via a pop-up notification. The Wi-Fi Direct standard imposes the connection to be set-up through the Wi-Fi Protected Setup protocol, and IP addresses are then configured through DHCP. After the assignment of IP addresses to both devices, a socket has to be opened to establish a two-way communication link between the two devices. In our implementation (see Section 3), we use TCP.

2.2 Nearby

Google Nearby Connections appeared in 2017 as a framework meant to completely abstract network-related complexities of D2D data exchange so that developers can focus on application features rather than communications technicalities. For the rest of this paper, we refer to this API as *Nearby*. Other than the application level functions and callbacks available to developers, technical specification of the inner-workings of Nearby are close to none.

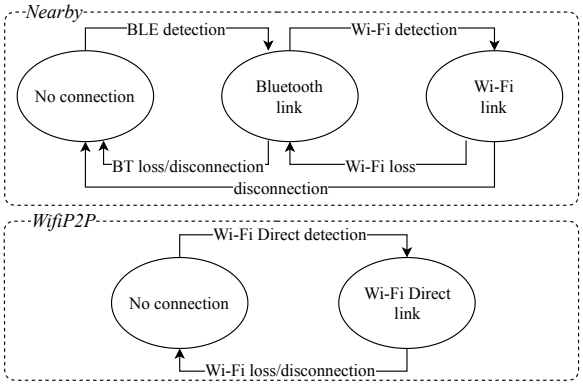


Figure 1: Summary of the connection process of both APIs.

The framework supports two types of topology: cluster and star. Cluster topology acts in a completely decentralized fashion, where any device can accept/start a connection from/with any other device. A star topology is more restrictive, as a clear client/server design has to be made prior to the connection. In a star, servers can accept all incoming connections but cannot initiate a connection, and a client can only be connected to a single server.

Nearby uses the commonly available wireless technologies found in off-the-shelf smartphones, namely Bluetooth Low Energy (BLE), Bluetooth, and Wi-Fi. In this paper, we consider the star topology as it is the only one to use Wi-Fi (the cluster topology uses Bluetooth), and because we aim to establish the upper-bound of D2D bit rates.

Using the `adb` software [9] and a spectrum analyzer, we unveil that the connection process is actually three-fold, as summarized in Figure 1. First, a BLE beacon is sent to notify all users within communication range that a server is available (*advertiser*) or a client is looking for a server to connect to (*discoverer*). Once two devices have detected each other, they first establish a Bluetooth connection in order to begin the data transfer as soon as possible. During this Bluetooth data transfer, the two devices attempt to establish a Wi-Fi connection; if they succeed, they automatically switch the data transfer over to the Wi-Fi link. If the Wi-Fi link drops due to poor signal, they fall back to the Bluetooth connection. The choices of the Wi-Fi standard and of the carrier frequency are always set by the server (advertiser) which acts as an AP.

After the establishment of the D2D link, the file transfer can start. Nearby triggers regular callbacks to notify the user-space of internal events, such as a status update on the transfer of the file.

2.3 Goodput measurement

We focus on Wi-Fi P2P and Nearby because they both enable higher transfer speeds than BLE and Bluetooth. Unfortunately, there is no integrated way in neither Wi-Fi P2P nor Nearby APIs to obtain the throughput of the D2D wireless link (i.e., the data exchange rate between the two Wi-Fi interfaces). Thus, instead of obtaining the throughput, we consider the *goodput* in the remainder of this paper, defined as the bit rate at the application level.

To obtain the goodput, we look at the behavior of the APIs. Nearby triggers a callback every time a *chunk* is received. A chunk is a application-level data block of $\leq 2^{16}$ bytes (the Nearby API

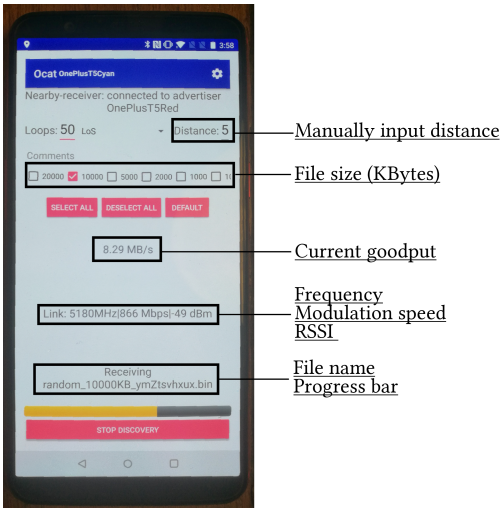


Figure 2: Ocat user interface.

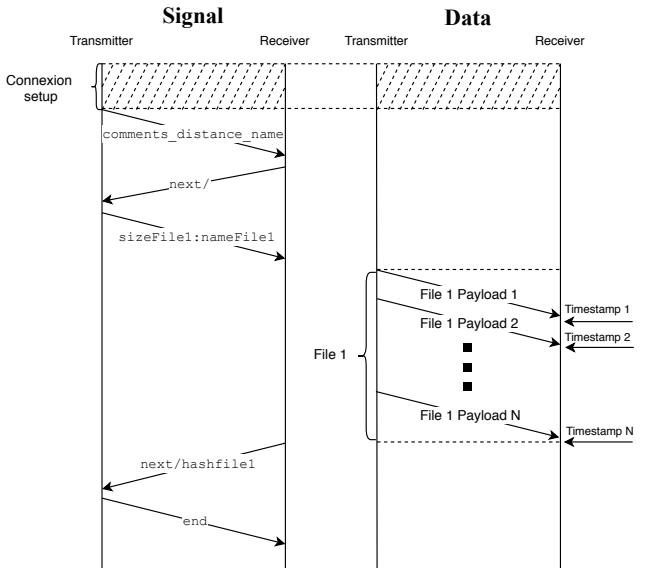


Figure 3: Representation of the synchronization protocol implemented in Ocat.

enforces this size). To obtain the mean goodput after the file has been sent, we collect timestamps at the arrival of each chunk. For both Nearby and Wi-Fi P2P APIs, we define the mean goodput as:

$$\text{Mean Goodput} = \frac{\sum_{n=2}^{N} \text{sizeChunk}(n)}{\text{Timestamp N} - \text{Timestamp 1}}. \quad (1)$$

We start at $n = 2$ because we cannot obtain the timestamp before the arrival of the first chunk.

2.4 Network-related data collection

To better understand D2D characteristics, we intend to collect any accessible network-related information or statistics, when using either APIs. As we have not found any straightforward means to obtain extra information using Nearby and Wi-Fi P2P APIs, we require the use of a third API supported in stock Android that solves the issue at least for the Nearby case. This API, called WifiManager [12], is built to handle regular client Wi-Fi connections in Android. We exploit the fact that once Nearby establishes a D2D link through Wi-Fi, this link acts as a standard Wi-Fi connection; thus, it becomes possible to use WifiManager to query the Wi-Fi interface to obtain extra information. Thanks to this workaround, we obtain the received signal strength indication (RSSI, in dBm), the link speed (in Mbit/s), and the used frequency (in MHz). Unfortunately, this technique exclusively works on the client side of the connection, due to the AP-side information being inaccessible in Android for privacy issues. While there arguably could be complex software solutions such as installing a custom Android firmware on the device, this falls outside of the scope of this paper since we focus on stock Android. We used in our experiments OnePlus 5T smartphones, which uses Wi-Fi 5 (802.11ac) in both Nearby and Wi-Fi P2P (contrarily to other brands that use Wi-Fi 4 for Nearby).

3 EXPERIMENTAL PROCEDURE

We designed and implemented Ocat (Opportunistic communications assessment tool), a mobile application whose purpose is to measure the connectivity characteristics between two or more devices using (so far) either Nearby or Wi-Fi P2P.

3.1 Ocat

In Figure 2, we show the user interface of the application. The process consists of generating random files of 10 MBytes, transmitting them between the devices, and storing the gathered information in a log file for post-processing. Recall that we only collect information once the data reaches the application level at the receiver side, as we cannot access network-related information at the OS level.

We unfortunately noticed that Nearby prematurely notified the transmitter user-space of transfer completion, even though significant part of the file was still being transferred. In Figure 3, we show the protocol exchanges implemented in Ocat. The purpose of the control plane is mainly to keep the synchronization between the transmitter and the receiver. This synchronization scheme allowed us to enforce strict transmission rules so that two files are never sent concurrently, which is paramount to fine-grained measurement of file transfer rates, thus solving the premature file transmission notification issue.

To ensure that files arrive correctly, the receiver sends a hash of the file to the transmitter along with the `next` command in order to request the next file in the queue. If the hash is correct, the transmitter sends the next file; otherwise, it sends the same file again (up to 5 times in our experiments).

3.2 Empirical goodput as a function of distance

We now have the necessary background to investigate the relationship between distance and goodput. In this second set of experiments, we laid on the ground a measuring tape which we used as a mean to measure the exact distance d between the transmitter and receiver devices. We put the smartphones on tripods, both of them

Figure 4: Photography of the experimental procedure. Both smartphones run the Ocat application, and we move one of the tripods according to a chosen stepping along the measuring tape (white line on the ground).

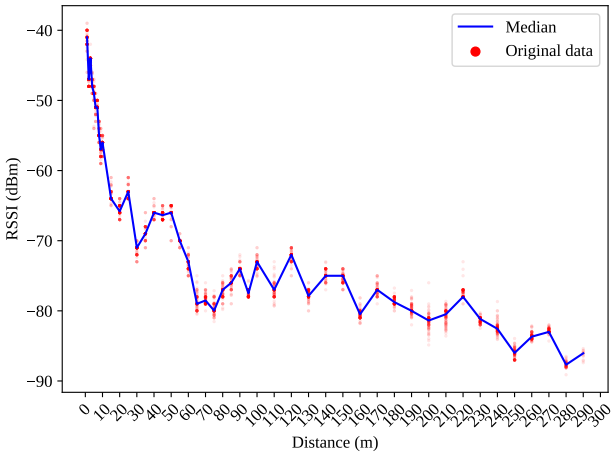


Figure 5: RSSI to distance measurements using the OnePlus 5T devices.

set at the height of 1.3 m. We start by putting them 1 m apart, as seen in Figure 4, and then we gradually increase the distance. Using Ocat, we connect two devices using either Wi-Fi P2P or Nearby (implemented as an option in the app). With an active D2D connection, the transmitter sends 100 randomly generated files to the receiver. Once all files have been sent, we move the receiver to the next mark and start the next round of transmissions. We repeat the process until the devices are too far to establish a connection.

In practice, a plethora of parameters related to the wireless medium has to be considered, such as shadowing, refraction, or fading. To limit as much as we could the influence of these parameters, we conducted our experiments in a rural environment with little to no external interference (Figure 4). This is compliant with our goal of measuring the maximum reachable transfer speeds. We checked for interference using a spectrum analyzer by verifying that no signal above noise-level (-100dBm in our case) was detected on the 2.4 GHz and 5 GHz bands.

4 RSSI FOR GOODPUT ESTIMATION

As previously explained, we can only access the RSSI, as Android does not give access to the transmission power. The RSSI, in dBm, is calculated as:

$$\text{RSSI}(d) = P_t + G_t + G_r - L - PL(d), \tag{2}$$

where d is the distance between the transmitter and the receiver, P_t is the transmission power, G_t is the transmitter's antenna gain, G_r is the receiver's antenna gain, L is the loss of the system, and $PL(d)$ is the path loss according to distance d. For the off-the-shelf smartphones that we use in our experiments, most of these parameters are not disclosed.

We first assume that smartphones transmit at constant power, meaning that P_t is fixed. We simplify the equation by letting $A = P_t + G_t + G_r - L$. This is done since we can only measure the RSSI, hence making it impossible to differentiate the terms. We have then:

$$\text{RSSI}(d) = A - PL(d). \tag{3}$$

4.1 Modeling the RSSI

We present in Figure 5 the RSSI measurements between two OnePlus 5T following the measurement methodology presented in Section 3. While at short distances (1 m to 20 m), the strength of the signal seems to be monotonically decreasing, it is not the case throughout the entire experiment. For instance, from 30 m to 50 m there is an apparent increase in the signal strength even though we increased the distance between the devices. While counter-intuitive, this increase of signal in spite of the growth of distance is a behavior typically found in the rural environment [21].

We consider three different models, namely the free-space path loss propagation model [2], the log-distance path loss model [2], and the accurate version of the two-ray ground-reflection model (as opposed to the approximation often found in the literature) [19, 21]. As the model yielding the best results is the accurate version of the two-ray ground-reflection model, as we will see below, we present further details of this model in Appendix 7.

We estimate the parameters for each model through a best-fit approach. As we only have the RSSI at each receiver, we need to estimate the global gain of the system A (see Equation 3) for all models. To find which path loss model leads to the best fit, we perform a least-square curve fitting using *lmfit* [16]. We remind the reader that, depending on the PL model used, several parameters are used to find the best fit. These parameters are listed in Table 1, where η is the attenuation exponent of the log-distance model and ϵ is a parameter that depends on the material of the reflection surface and h_r/h_t the height of the receiver/transmitter.

We show in Figure 6 how the different models fit our measures. The free-space model leads to a pretty good fitting by using a global gain $A \approx 10.46$. The log-distance model seems to exhibit the

Table 1: Summary of the models and their parameters.

RSSI/PL Model	Param 1	Param 2	Param 3
Freespace	A	-	-
Log-Distance	A	η	-
Two-Ray ground-reflection	A	ϵ	h_r/h_t

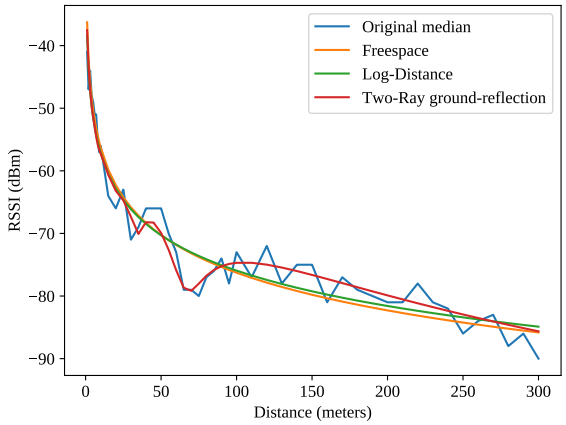


Figure 6: RSSI Models after least-square fitting. The two-ray ground ground-reflection model is the only model capturing the signal increase while augmenting the distance.

same behavior, however, the best-fit yields a gain of $A \approx 8.3$ and the attenuation exponent is set to $\eta \approx 1.8$; the problem is that η is under the minimum value of 2 which corresponds to the propagation of a signal in free-space. Thus, we consider this fit to be invalid and ignore it since fixing this parameter to a minimum of 2 is equivalent to calculating the free-space model.

The two-ray ground-reflection model gives the best results. It can capture specific phenomena such as the signal increase observed between 30 m and 50 m (which is the consequence of constructive interference due to the ground reflection). The algorithm fixes the parameters $A \approx 9.19$ dB and $\epsilon = 1.009$. It is not a surprise that the algorithm gave $h_r = h_s \approx 1.38m$, which corresponds to the height of the phones on top of the tripods.

4.2 RSSI and Goodput

To derive a distance-to-goodput model, we need to establish if a clear correlation between a given RSSI value and a goodput exists. We know from the Wi-Fi 5 standard that, for each value of RSSI, a particular modulation is chosen in order to achieve the best trade-off between throughput and error resilience [6]. Typically, modulations reaching the highest throughputs are quite sensitive to noise and are unsuitable for long-distance communications. If we were in a noisy environment, the RSSI measurements should be correlated to a throughput through a rate-adaptation algorithm, but since this is not the case, the sole RSSI is enough for our upper-bound model.

Recall that, in Android, the modulation used by the Wi-Fi interface is not disclosed; still, it can be inferred from the link speed

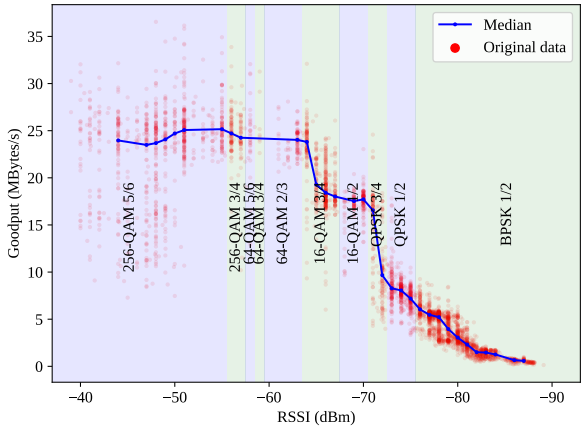


Figure 7: Empirical RSSI-to-goodput relationship with the indication of the modulation used.

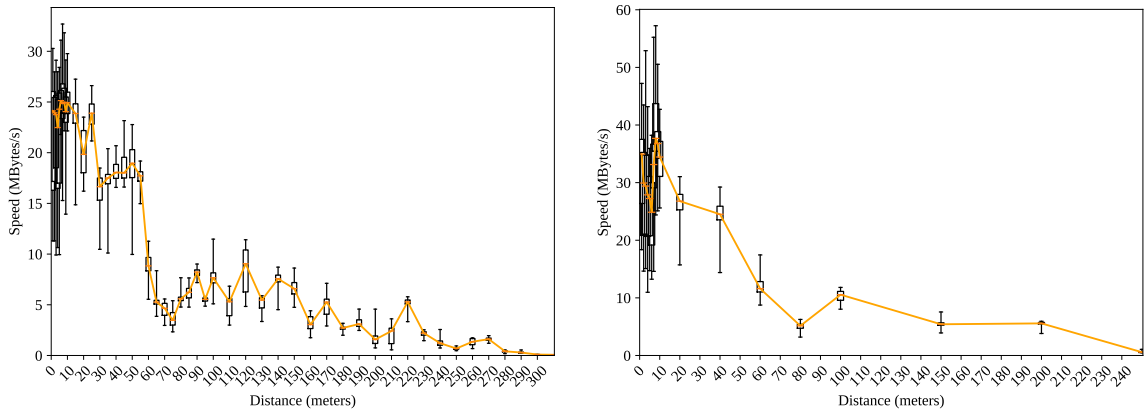
Table 2: Android Wi-Fi 5 theoretical maximum throughput using 80MHz bandwidth & 2*2 MIMO, 400ns GI

Bit rate	Modulation	Redundancy	Max RSSI	Min RSSI
866	256-QAM	5/6	-	-55
780	256-QAM	3/4	-56	-57
650	64-QAM	5/6	-58	-58
585	64-QAM	3/4	-59	-59
520	64-QAM	2/3	-60	-63
390	16-QAM	3/4	-64	-67
260	16-QAM	1/2	-68	-70
195	QPSK	3/4	-71	-72
130	QPSK	1/2	-73	-75
65	BPSK	1/2	-76	-

if the number of MIMO antennas is known. We compiled Table 2 from the Wi-Fi 5 standard to better understand the behavior of the throughput. The BPSK modulation does not have a minimum RSSI in the tested devices as it continues to operate until the signal is lost, and the same logic applies to the 256-QAM which operates no matter how high the RSSI is.

The units we consider are dBm for the RSSI and MBytes/s for the goodput. In Section 2.3, we calculated a mean goodput per file, and we now apply the same idea to the RSSI. The RSSI given by WifiManager is not guaranteed to be up-to-date. We mitigate this issue by weighting the RSSI of each chunk using the transmission time of the chunk to establish the mean RSSI of the file.

As the choice of a specific RSSI value is not left to the user, we cannot guarantee that all possible values of RSSI are covered. However, by moving the tripods back and forth several times, we were able to cover a wide range of possible values, from -40 dBm to -87 dBm. In Figure 7, we present plot the mean goodput per file sent, as a function of the mean RSSI during the file transmission. Each red dot on the figure represents a sample (a file transmitted), and we additionally superimposed the RSSI range of all modulations to give a physical layer perspective to the reader. The blue line



(a) Nearby (b) Wi-Fi P2P

Figure 8: Goodput measurements from OnePlus 5T to OnePlus 5T connections with both APIs. The box-plots' whiskers include 95% of the data.

represents the median values of all values of RSSI, given that at least 50 samples over the same RSSI are available.

We note that the median does not significantly change in the range -40 dBm to -62 dBm despite five different modulations. Since prior works have found a clear correlation between modulation and energy consumption in smartphones [23], this reveals a potential energy consumption issue. As the goodput using the modulations between 256-QAM 5/6 and 64-QAM 2/3 are unequivocally equivalent, it seems that using the 64-QAM 2/3 modulation from -40 dBm to -62 dBm could be a better alternative as we know that it consumes less energy. Also, there are two significant drops in the measured median goodput, at precisely -63 dBm and -72 dBm. While it would seem intuitive to think that they correspond to a modulation change, they are not correlated to a particular modulation.

We can also observe a large spread around the median goodput from -40 to -50dBm. For instance, the goodput varies from 7 MBytes/s to 35 MBytes/s with a median around 24.5 MBytes/s.

The goodput as a function of RSSI, i.e. the results presented in Figure 7 can be easily approximated by a linear piecewise function $\widehat{\text{goodput}}(x)$, where x is the RSSI:

$$\widehat{\text{goodput}}(x) = \begin{cases} 24.26 & x \geq -64, \\ -0.36x + 42.35 & -64 < x \leq -71, \\ -0.79x + 66.44 & -71 < x \leq -82, \\ -0.24x + 21.13 & -82 < x. \end{cases} \quad (4)$$

Now that we defined our model of the goodput as a function of the RSSI, we need to extend it to become as a function of the distance. This is the topic of the next section.

5 GOODPUT VS. DISTANCE

5.1 Analysis of empirical observations

We were able to maintain a D2D connection up to 280 m distance using Wi-Fi P2P and up to 310 m using Nearby. This difference is explained by the fact that Wi-Fi P2P is solely based on Wi-Fi 5 (5 GHz band) while Nearby may fall back to Bluetooth (2.4 GHz band), as explained in Section 2.2. Nonetheless, when Nearby we enforced Nearby to remain in the 5 GHz band, the distance limit is around 280 m, exactly like Wi-Fi P2P. To the best of our knowledge, there is no reference in the literature reporting D2D experiments that achieve such distances using off-the-shelf Android devices.

In Figure 8, we show the goodput results when the devices communicate using both Nearby and Wi-Fi P2P. When we observe the results for Nearby (Figure 8a), several patterns emerge. On short distances (1 m to 20 m), the throughput seems relatively constant ranging from 23 MBytes/s to 25 MBytes/s. The dispersion around the median is however quite large, with values as high as 33 MBytes/s and as low as 10 MBytes/s – but this is consistent with our observations in the RSSI analysis in Section 4.2, where a close-range communication generated more spread around the median. We also observe an increase in the goodput in the range 30 m to 50 m. We were expecting this behavior as it corresponds to the constructive interference due to ground reflection (see Section 4.2). This enables the devices to maintain a high goodput, ranging from 13 MBytes/s to 17 MBytes/s; it is notable that such a high goodput is achieved within medium-range distances.

As for the Wi-Fi P2P API (Figure 8b), we see that the large variance is also found within short distances (1 m to 20 m). The median ranges from 35 MBytes/s to 39 MBytes/s, which is significantly higher than the speed reached with Nearby. We believe that this difference is due to a software bottleneck. Another hint is the RSSI-to-goodput relationship (Section 4.2) that showed no increase of throughput, albeit modulations with higher theoretical speeds. Hence, we can deduce that Nearby's maximum goodput is, in fact, software-related more than hardware-related.

The next pattern, which we labeled as the sawtooth behavior, can be observed in the range 60 m to approximately 270 m. This is a consequence of the low RSSI over long-range communications, where any obstacle may increase or decrease the signal (e.g., leaves

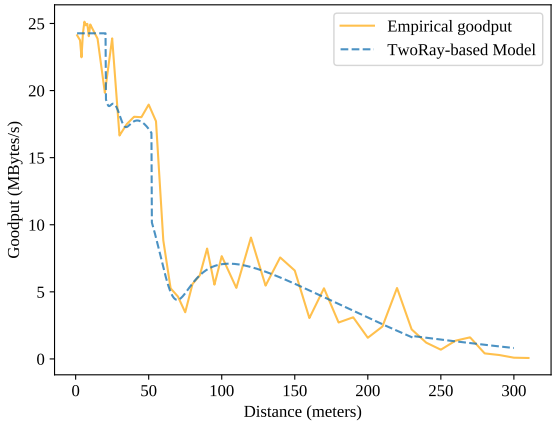


Figure 9: Comparison between the empirical data and our model for Nearby.

on the ground and nearby trees). Regardless, the goodput ranges from 1 MByte/s to 7 MBytes/s, which is still a significant value.

Over even longer distances, between 280 m and 310 m, the goodput measurements are less accurate due to Nearby's behavior. When the Wi-Fi connection drops between two devices, which can happen when the RSSI is around -85 to -90 dBm, Nearby automatically falls back to the Bluetooth connection. Nearby still allows devices to exchange data, but this time with a significantly lower bit-rate between 10 KBytes/s to 100 KBytes/s.

5.2 Model of the goodput in function of the distance

We now propose a model to estimate the goodput based on the distance between two Android devices. We use the $PL_{\text{tworay}}(d)$ model (see Appendix 7) to estimate the path loss and fix the inner parameters from the least-square estimation, where $A \approx 9.19$ (see Section 4.1). From Equation 3, we derive:

$$\widehat{\text{RSSI}}(d) = 9.19 - PL_{\text{tworay}}(d). \tag{5}$$

To obtain the goodput as a function of the distance, we combine Equations 4 and 3:

$$\text{goodput}(d) = \widehat{\text{goodput}}(\widehat{\text{RSSI}}(d)). \tag{6}$$

In Figure 9, we show how this model fits the dataset. The original data is the same as the one presented in Figure 8a, but in Figure 9 we only show the median for the sake of readability. Our model presents the expected characteristics that we observed in practice, notably the sudden surge in goodput in the range 30 m to 50 m along with the significant drop after 70 m.

Our proposed model, which is the upper-bound of D2D communication in off-the-shelf Android smartphones, now allows the research community to estimate the bit rate based on the distance between two devices.

6 RELATED WORK

Given the ubiquitous nature of Wi-Fi in modern devices, there are several studies that evaluate the performance of the different Wi-Fi standards. Zeng *et al.* explore the relationship between overlapping Wi-Fi standards (e.g., 802.11n/ac using the same frequency) and power consumption [23]. Saha et al. extend this idea by focusing on the characterization of the relationship between Wi-Fi throughput of smartphones and battery consumption using several popular models [20]. Chowdhury *et al.* [5] have previously proposed to use the estimation of the RSSI as a mean to obtain the throughput between a base station and a mobile client, based on the distance between the two. Contrary to our empirical approach, they took the physical speed of the Wi-Fi 4 as a reference model. Certain domains commonly infer the distance between two devices from the received signal strength. This is notably done in wireless sensor networks [13, 22], vehicular networks [4, 21], and in mobility detection of smartphones [17].

Qayyum *et al.* [18] take an empirical measurement of the throughput according to distance using a mobile Android application. Interestingly, while they admittedly use dated hardware and software compared to ours, they report significantly shorter range of $\approx$ 10 m and undeniably slower throughput as they used Bluetooth.

Several authors have implemented D2D-based frameworks using Android devices Android. For instance, Keller et al. propose a cooperative streaming system named Microcast [14]. Other examples of full-fledged D2D data sharing solutions exist, but they all assume the devices communicate within a short range [1, 15]. Another issue is the assumption that devices can maintain several simultaneous D2D connections, while in reality this is not guaranteed and switching from one Wi-Fi Direct connection to the next may create a latency of several seconds [8]. On top of this, while these frameworks may be functional, they are not found in stock Android and are therefore not always available to developers nor have guaranteed support.

While all these works are substantial to their domains, they often ignore the importance of distance as a parameter for the throughput estimation in a D2D context or were outdated in regards to hardware and software. Hopefully, our work is an appreciated contribution to this goal.

7 CONCLUSION AND FUTURE WORK

In this paper, we explored the current state of D2D communications in stock Android devices. To this end, we first designed and implemented Ocat, an Android application that eases the measurement process of D2D communications through both Google Nearby and Wi-Fi P2P. As a result, we were able to establish a model for the upper-bound goodput of D2D communications as a function of the distance between the communicating devices.

We showed that the Nearby API can maintain a goodput of 25 MBytes/s in the 1 m to 20 m range. Within the same range, Wi-Fi P2P can reach up to 39 MBytes/s, showing that Nearby possibly lacks software optimization to achieve similar speeds. In our findings, we note that the ground reflection has a significant impact on goodput performance, which can be positive within medium distances between 30 m to 50 m, enabling a steady goodput rate between 13 MBytes/s to 17 MBytes/s. To our surprise, devices were

able to communicate at fairly long distances (280 m to 310 m), which was undocumented so far.

In our future work, to better assess D2D data sharing capabilities, we intend to exchange data between several devices at the same time and at different distances. We also want to model the asymmetrical relationship between emitter and transmitter devices, since our tests showed that devices may perform differently when they transmit or receive signals. Ultimately, this work enables a new sort of characterization for D2D communications by establishing how much data can be exchanged between devices through a spatiotemporal contact. We believe that our work will significantly help researchers and application designers understand the possibilities and limits of D2D communications using Android devices.

REFERENCES

[1] Xuan Bao, Yin Lin, Uichin Lee, Ivica Rimac, and Romit Roy Choudhury. 2013. DataSpotting: Exploiting naturally clustered mobile devices to offload cellular traffic. In *2013 Proceedings IEEE INFOCOM*. IEEE, 420–424. https://doi.org/10.1109/INFCOM.2013.6566807

[2] K. Benkic, M. Malajner, P. Planinsic, and Z. Cucej. 2008. Using RSSI value for distance estimation in wireless sensor networks based on ZigBee. *Proceedings of the 15th International Conference on Systems, Signals and Image Processing* (2008), 303–306. https://doi.org/10.1109/IWSSIP.2008.4604427

[3] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano. 2013. Device-to-device communications with Wi-Fi Direct: overview and experimentation. *IEEE Wireless Communications* 20, 3 (jun 2013), 96–104. https://doi.org/10.1109/MWC.2013.6549288

[4] Chien-Ming Chou, Chen-Yuan Li, Wei-Min Chien, and Kun-chan Lan. 2009. A Feasibility Study on Vehicle-to-Infrastructure Communication: WiFi vs. WiMAX. In *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*. IEEE, 397–398. https://doi.org/10.1109/MDM.2009.127

[5] Helal Chowdhury, Janne Lehtomäki, Juha-Pekka Mäkelä, and Sastri Kota. 2010. Data Downloading on the Sparse Coverage-Based Wireless Networks. *Journal of Electrical and Computer Engineering* 2010 (2010), 1–7. https://doi.org/10.1155/2010/843272

[6] Cisco. 2014. *802.11ac: The Fifth Generation of Wi-Fi.* Technical Report March. 1–4 pages.

[7] Marco Conti and Silvia Giordano. 2014. Mobile ad hoc networking: milestones, challenges, and new research directions. *IEEE Communications Magazine* 52, 1 (jan 2014), 85–96. https://doi.org/10.1109/MCOM.2014.6710069

[8] Colin Funai, Cristiano Tapparello, and Wendi Heinzelman. 2017. Enabling multi-hop ad hoc networks through WiFi Direct multi-group networking. In *2017 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 491–497. https://doi.org/10.1109/ICCNC.2017.7876178 arXiv:1601.00028

[9] Google. [n. d.]. ADB. https://developer.android.com/studio/command-line/adb

[10] Google. [n. d.]. Nearby - Connections API Overview. https://developers.google.com/nearby/connections/overview

[11] Google. [n. d.]. Wi-Fi peer-to-peer overview. https://developer.android.com/guide/topics/connectivity/wifip2p

[12] Google. [n. d.]. WifiManager. https://developer.android.com/reference/android/net/wifi/WifiManager

[13] Lei Guan, Xing Zhang, Zongchao Liu, Yu Huang, Ruoqian Lan, and Wenbo Wang. 2013. Spatial modeling and analysis of traffic distribution based on real data from current mobile cellular networks. *Proc. of ICCP'13 - 2013 IEEE International Conference on Computational Problem-Solving* (2013), 135–138. https://doi.org/10.1109/ICCPS.2013.6893524

[14] Lorenzo Keller, Anh Le, Blerim Cici, Hulya Seferoglu, Christina Fragouli, and Athina Markopoulou. 2012. MicroCast: Cooperative Video Streaming on Smartphones. In *Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys '12*, Vol. 68. ACM Press, New York, New York, USA, 57. https://doi.org/10.1145/2307636.2307643

[15] Kyunghan Lee, Injong Rhee, Joohyun Lee, Song Chong, and Yung Yi. 2010. Mobile data offloading. In *Proceedings of the 6th International COnference on - Co-NEXT '10*, Vol. 21. ACM Press, New York, New York, USA, 1. https://doi.org/10.1145/1921168.1921203

[16] Matthew Newville and Till Stensitzki. 2017. Non-Linear Least-Squares Minimization and Curve-Fitting for Python Matthew Newville, Till Stensitzki, and others. (2017).

[17] Pavan Kumar Pedapolu, Pradeep Kumar, Vaidya Harish, Satvik Venturi, Sushil K Bharti, Vinay Kumar, and Sudhir Kumar. 2017. Mobile Phone User's Speed Estimation using WiFi Signal-to-Noise Ratio. In *Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing - Mobihoc '17*. ACM Press, New York, New York, USA, 1–2. https://doi.org/10.1145/3084041.3084072

[18] Shiraz Qayyum, Mehrab Shahriar, Mohan Kumar, and Sajal K. Das. 2013. PCV: Predicting contact volume for reliable and efficient data transfers in opportunistic networks. *Proceedings - Conference on Local Computer Networks, LCN* (2013), 801–809. https://doi.org/10.1109/LCN.2013.6761335

[19] Theodore Rappaport. 2001. Wireless Communications: Principles and Practice. (2001).

[20] Swetank Kumar Saha, Pratik Deshpande, Pranav P. Inamdar, Ramanujan K. Sheshadri, and Dimitrios Koutsonikolas. 2015. Power-throughput tradeoffs of 802.11n/ac in smartphones. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, Vol. 26. IEEE, 100–108. https://doi.org/10.1109/INFOCOM.2015.7218372

[21] Christoph Sommer, Stefan Joerer, and Falko Dressler. 2012. On the applicability of Two-Ray path loss models for vehicular network simulation. In *2012 IEEE Vehicular Networking Conference (VNC)*. IEEE, 64–69. https://doi.org/10.1109/VNC.2012.6407446

[22] Jiuqiang Xu, Wei Liu, Fenggao Lang, Yuanyuan Zhang, and Chenglong Wang. 2010. Distance Measurement Model Based on RSSI in WSN. *Wireless Sensor Network* 02, 08 (2010), 606–611. https://doi.org/10.4236/wsn.2010.28072

[23] Yunze Zeng, Parth H. Pathak, and Prasant Mohapatra. 2014. A first look at 802.11ac in action: Energy efficiency and interference characterization. In *2014 IFIP Networking Conference*, Vol. 20. IEEE, 1–9. https://doi.org/10.1109/IFIPNetworking.2014.6857103

[24] Xiangming Zhu, Yong Li, Depeng Jin, and Jianhua Lu. 2017. Contact-Aware Optimal Resource Allocation for Mobile Data Offloading in Opportunistic Vehicular Networks. *IEEE Transactions on Vehicular Technology* 66, 8 (aug 2017), 7384–7399. https://doi.org/10.1109/TVT.2017.2668396

TWO RAY GROUND REFLECTION MODEL

The idea behind this model is that the receiver obtains two copies of the same signal, the original one and a copy reflected from the ground. The principle is to calculate the phase difference between the two copies of the signal to verify whether the interference is constructive or destructive, based on the heights of the transmitter and the receiver. In our case, these heights are equal, which guarantees that the line of sight distance between the two devices is $d_{los} = d$. The distance of the reflected signal is $d_{reflect} = \sqrt{d^2 + (h_r + h_t)^2}$, where h_r and h_t are the heights of the transmitter and receiver devices, respectively (again, equal in our setup). The phase difference as $\varphi = 2\pi \frac{d_{los} - d_{reflect}}{\lambda}$.

To obtain the final model, we also need the reflection coefficient, which gives the capacity of the ground to reflect an electromagnetic wave. It is given by $\Gamma_\perp = \frac{\sin\theta - \sqrt{\epsilon - \cos^2\theta}}{\sin\theta + \sqrt{\epsilon - \cos^2\theta}}$, where θ is the angle of incidence of the reflected signal based on the heights of the transmitter and receiver, and ϵ is a fixed parameter based on the material.

The two-ray ground reflection model is then written as [21]:

$$PL_{\text{tworay}}(d) = 20\log_{10}\left[4\pi\frac{d}{\lambda}|1 + \Gamma_\perp e^{i\varphi}|^{-1}\right]. \quad (7)$$

Using Euler's formula, we replace the complex exponential in the equation as:

$$PL_{\text{tworay}}(d) = 20\log_{10}\left[4\pi\frac{d}{\lambda}|1 + \Gamma_\perp\cos\varphi + \Gamma_\perp i\sin\varphi|^{-1}\right]. \quad (8)$$

As the modulus of a complex number yields a real number, we use this final equation:

$$PL_{\text{tworay}}(d) = 20\log_{10}\left[4\pi\frac{d}{\lambda}\sqrt{(1 + \Gamma_\perp\cos\varphi)^2 + \Gamma_\perp^2\sin^2\varphi}^{-1}\right]. \quad (9)$$