Nearby connections strategies: Features, usage, and empirical performance evaluation

Tomas Lagos Jenschke, Marcelo Dias de Amorim, Serge Fdida

Internet of Things 23 (2023) 100895

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Research article

Nearby connections strategies: Features, usage, and empirical performance evaluation

Tomas Lagos Jenschke *, Marcelo Dias de Amorim, Serge Fdida

Sorbonne Université, CNRS, LIP6, Paris, France

ARTICLE INFO

Dataset link: https://github.com/tlagos1/AtomD

Keywords:
Device-to-device (D2D)
Nearby connections
Throughput
Evaluation tool
Experimental evaluation

ABSTRACT

The Device-to-Device (D2D) communications involve a direct, peer-to-peer link between devices that operates independently of fixed network infrastructures. It can either complement existing network infrastructures or be used as a standalone network to provide services, such as distributing content, supporting emergency services during natural disasters, and enabling smartphone applications like AirDrop (iOS) and Nearby Share (Android). However, the behavior of D2D communication is not always predictable, and there are many challenges to implementing and deploying D2D communication systems in real-world situations. In particular, it is hard to know what throughput one can obtain, as nominal numbers provided by the documentation are seldom observed in practice. This paper focuses on Nearby Connections Application Programming Interface (API) provided by Google. This API offers distinct strategies in function of the network's topology, and choosing the right one for an application is a challenging task as they perform differently. In this paper, we contribute to the research community in two ways. Firstly, we investigate the real-time performance of throughput of two strategies (STAR and POINT-TO-POINT, as they are the only ones to use Bluetooth and Wi-Fi Direct to transfer data). Our study shows that the POINT-TO-POINT strategy generally achieves high throughput performance and is suitable for use cases involving high network traffic. In contrast, the STAR performs poorly in throughput, which can result in link instability and slow transfers in some cases. Secondly, we disclose a tool to help network designers evaluate their networks and fine-tune their protocols and algorithms according to their specificities.

1. Introduction

The concept of D2D communications, although out there for a few years, is booming in cutting-edge technologies such as 5G networks and beyond, besides local networking methods for dynamic information distribution [1]. It consists of peer-to-peer (P2P) communication without relying on fixed network infrastructure [2,3]. Moreover, its ability to be a stand-alone network has allowed it to enhance existing network infrastructures or to leverage them for stand-alone services. Examples of applications include alternative ways to deliver content to address highly demanding and efficiently distributed communications, such as emergency services and natural disasters [4–6].

In the consumer market, we have witnessed its wide adoption thanks to applications such as Apple's Airdrop [7] and, more recently, Goggle's Nearby Share [8]. Airdrop and Nearby Share are applications that rely on, respectively, Multipeer Connectivity and Nearby Connections, which are D2D APIs that provide all the necessary features and protocols for devices to connect and communicate with nearby devices. In such systems, one of the main steps is to run a NDP to detect devices within reach. Then, if

* Corresponding author.
E-mail addresses: tomas.lagos-jenschke@lip6.fr (T. Lagos Jenschke), marcelo.amorim@lip6.fr (M. Dias de Amorim), serge.fdida@lip6.fr (S. Fdida).

https://doi.org/10.1016/j.iot.2023.100895
Received 24 April 2023; Received in revised form 21 July 2023; Accepted 1 August 2023
Available online 7 August 2023

the user decides to connect to one of them, the system provides a direct wireless link so that the devices can communicate data. These links use state-of-the-art protocols such as Bluetooth and Wi-Fi Direct. Both Multipeer Connectivity and Nearby Connections are exposed to developers who wish to create their own D2D applications.

The motivation for this paper comes from the performance issues we faced in designing our own applications. More specifically, *what performance to expect in a real deployment?* A key metric for communication is throughput, which is critical in various applications. For example, in file-sharing, the main expectation is high throughput. On the other hand, low latency and high reliability are essential for a responsive and lag-free online gaming experience.

In contrast, high throughput and low latency are crucial for smooth, real-time video streaming. The overall network context should also be considered when evaluating the throughput performance of D2D communication. The impact of factors such as network congestion, device capabilities, and interference from other devices should also be considered.

Knowing in advance what performance to expect is thus of paramount importance. Both Apple and Google provide the developers with expected, nominal performance numbers, but in practice, they are hardly observed. The behavior of D2D communication is not deterministic, and much information needs to be included, resulting in many challenges and difficulties in implementing and deploying D2D communication systems. This has been evident in frequent discussions on forums such as Stack Overflow, where many developers and users have raised questions and expressed concerns about the behavior of D2D communication. In addition, the mechanisms and protocols used in D2D communication are complex. As a result, they often need to be better documented, resulting in more information about how D2D communication systems operate and how to troubleshoot issues that may arise.

We propose an experimental framework to assess performance parameters observed in real conditions. It is worth noting that several investigations focus on theoretical throughput analysis in D2D link characterization, although they provide baseline performance in ideal conditions.

In Tan et al. [9], the authors present a theoretical study that proposes a multi-hop transmission scheme combining 5G and WiFi Direct to improve edge node transmission performance, evaluated through a system-level simulation platform.

In Zreikat et al. [10], a theoretical evaluation of 5G/WiFi-6 coexistence is performed to overcome limitations in coverage and capacity, studying performance parameters by modeling the system using MOSEL-2 simulation language.

Mankar et al. [11] explores the interplay between D2D communications and real-time monitoring systems in a cellular-based IoT network, characterizing D2D communication performance through average network throughput and quantifying real-time application performance using the Age of Information (AoI) metric. Analytical derivations and simulations are used to assess the mean success probability for D2D links, capturing the spatial disparity in AoI performance through moments of conditional success probability and conditional scheduling probability for status update links.

In this paper, we report on several experiments and share with the community our insights into the performance of the Nearby Connections API using multiple models of smartphones of varying ages. We focus on Nearby Connections as the literature still lacks experiments on this framework. Although we plan to conduct similar experiments on Multipeer Connectivity in the future, we invite the reader to refer to existing outstanding work, such as Stute et al.'s analysis of Apple Wireless Direct Link (AWDL) protocol [7].

In a nutshell, the Nearby Connections API provides three strategies to establish D2D communications: CLUSTER, STAR, and POINT-TO-POINT. As their names indicate, they differ in the network topology and, consequently, in the flexibility that they offer to the application. CLUSTER allows $(n \times m)$ connectivity, STAR leads to a $(1 \times m)$ network, and POINT-TO-POINT constrains the topology to (1×1). Besides the topology, another fundamental difference is in their performance. While CLUSTER uses Bluetooth for data transfers, both STAR and POINT-TO-POINT rely on both Bluetooth and Wi-Fi. The high topological flexibility that CLUSTER provides is then highly penalized by much slower links.

The difficulty is to know exactly the performance differences between the STAR and POINT-TO-POINT strategies. Google gives some advice in their documentation [12] but reality shows different numbers. In this paper, we address exactly this point by digging into the operation of the Nearby Connections API. Our contributions are:

- We review the different strategies that Google proposes through the Nearby Connections API. We also identify the aspects that need specific attention from developers when designing their own applications on top of the API.
- We share with the community AtomD. A tool we developed to perform a number of tests of the Nearby Connections API. We show the steps we follow as well as a methodology that developers can follow to characterize their own setups.
- We report on the experiments we have run using different smartphones and discuss the performance that one may achieve in terms of throughput. As we will see, different strategies lead to very different performance levels, which forces developers to think thoroughly when deciding which strategy to adopt in their applications.

The rest of the paper is organized as follows. We review in Section 2 the characteristics and strategies of the Nearby Connections API. Section 3 introduces the problem statement, our developed application, and our approach to addressing the problem. In Sections 4 and 5, we describe the setup used for our experimental evaluation plus the methodology we employed, and in Section 6, we present the experimental results. We make a comparative analysis of the pros and cons of each strategy based on the results obtained in Section 7. We postpone the related work to Section 8 so that the reader has enough material to capture our specificities to other papers in the literature. Finally, in Section 9, we conclude the paper and list ideas for future developments.

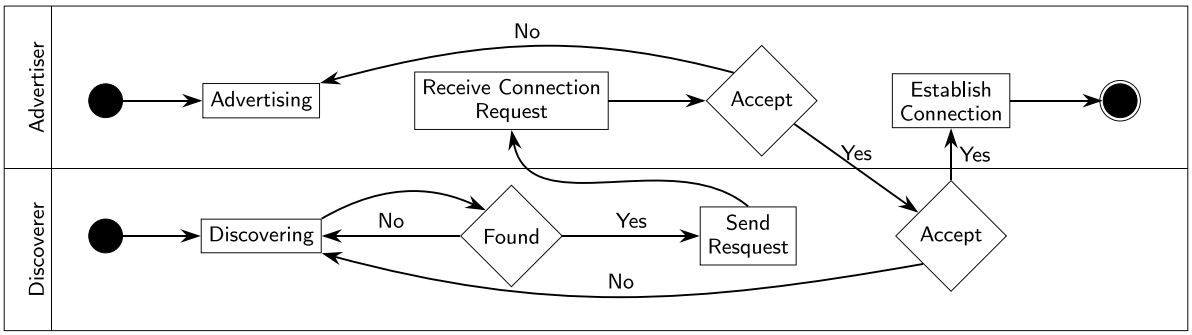


Fig. 1. NDP flowchart using the Nearby Connections API. During this process if the Discoverer finds an Advertiser, then the Discoverer has the option to send a connection request. If both devices agree to accept the connection request, then they will establish a D2D communication link.

2. Nearby Connections API

The Nearby Connections API was created by the Google development team in order to bring D2D connections to both Android and IoT devices [13].[1] The API allows devices to perform NDP and establish direct, offline wireless links to exchange data.

2.1. General functioning

The first feature that Nearby Connections API provides is the *discovery* of neighboring devices. To this end, it relies on either Bluetooth Classic, which reaches distances of up to 100 m in line-of-sight areas, or Bluetooth Low Energy (BLE), which reaches distances of up to 20 m [14]. During the discovery phase, devices can play two different types of roles:

- **Advertiser.** The device is responsible for announcing itself so that neighboring devices can detect it.
- **Discoverer.** The device with this role is responsible for listening to the environment to detect potential Advertisers.

When two devices discover each other, the Discoverer can send a connection request to the Advertiser, which triggers a symmetric authentication flow where both parties independently accept or reject the connection request (see Fig. 1). At this point, the link is supposed to be bidirectional and symmetric (i.e., the devices become both potential source and destination of data).

After the establishment of the direct link, the Nearby Connections API offers three data transmission methods:

- **Chunk transmission.** The source Nearby Connections API can send data by encapsulating it in a set of bytes. A chunk can encompass up to a size of 1 047 552 B.
- **File transmission.** The Nearby Connections API manages file transmissions by subdividing them into data chunks. The size of each chunk is variable and under the discretion of the Nearby Connections API, depending on the link quality.
- **Streaming.** The Nearby Connections API can send large amounts of data on the fly, such as an audio stream via stream payloads.

Once the transmission has been successfully completed, the Nearby Connections API reports the `SUCCESS` status. Alternatively, if the user cancels the communication during transmission, it returns `CANCELLED`, or if the transmission was interrupted during transmission, it returns `FAILURE`.

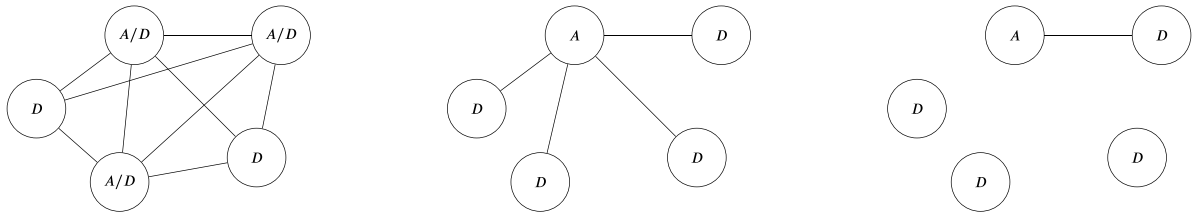
Finally, the Nearby Connections API provides the function to close the link and disconnect listeners in case the user requires them.

2.2. Strategies

The Nearby Connections API proposes three strategies (see Fig. 2) concerning the network topology and at least one wireless technology as the link medium [15–17]. These strategies must be set during the initial configuration of the API instance (before performing the NDP) and cannot be changed until a new instance is created.

CLUSTER Strategy. This is Nearby Connections API's default strategy. As the name suggests, it creates a cluster topology ($n \times m$) between devices that are within range of each other. To this end, the Nearby Connections API enables devices with the Discovery role to establish D2D communications with one or multiple devices within the same cluster. It is important to underline that, in the CLUSTER strategy, during the network lifetime a device can work as a Discoverer and an Advertiser for the sake of flexibility. This

[1] Recently, Google announced a beta to run Nearby Share between Android and Windows devices – https://www.android.com/better-together/nearby-share-app.



(a) CLUSTER b) STAR Strategy c) POINT-TO-POINT Strategy

Fig. 2. Nearby Connections API's available strategies: (a) CLUSTER, with two devices set as Discoverer-only (D) and three devices set as Advertiser/Discoverer (A/D), (b) STAR, with one Advertiser (which becomes the root) and four Discoverers (which become leaves), and (c) POINT-TO-POINT, with one Advertiser and four Discoverers (only one Discoverer could connect to the Advertiser).

strategy fits applications that enforce a more elastic architecture and do not require the transfer of large payloads. Consequently, it uses Bluetooth 5.0 as a D2D communication link since it bets on energy efficiency and flexibility over throughput.

STAR Strategy. This strategy focuses on architectures involving a star topology $(1 \times m)$, i.e., applications that distribute information from a centralized point (e.g., streaming or client–server models). Unlike the CLUSTER strategy, the STAR strategy leads to less flexibility but can benefit from higher bandwidth. It relies on both Wi-Fi Direct and Bluetooth 5.0 for communication links. To build a star topology, the Nearby Connections API allows the Advertiser to keep one or more Discoverers connected to it. In turn, Discoverers can only maintain a single connection with their respective Advertiser. It should be noted that during NDP, a device can be set up with both Discoverer and Advertiser roles. However, when a link is established using this strategy, the device will retain only the role with which it established the connection (i.e. if a device receives an advertisement from another device and decides to establish a connection with that device, it will only maintain its role as a Discoverer. Otherwise, it will keep its Advertiser role).

POINT-TO-POINT Strategy. This strategy focuses on maintaining a single connection (1×1) between two devices to maximize link bandwidth. It is intended for applications that require the exchange of large payloads, such as file transfers. The Nearby Connections API limits both the Advertiser and the Discoverer to maintain a single connection. In addition, like the STAR strategy, it supports role parallelism but keeps one of the roles after the link is established. It uses Wi-Fi Direct to create its links, and it can work with Wi-Fi Aware on compatible devices. It also uses Bluetooth 5.0 in case the Wi-Fi interface is disabled.

2.3. Underlying data transfer technologies

The Nearby Connections API relies on both Bluetooth 5.0 and Wi-Fi as data transfer technologies, depending on the strategy. Bluetooth 5.0 is integrated as the basis for each of the three strategies. This technology is notable for its energy-efficient design, which makes it ideal for battery-powered devices. It is also cost-effective and provides secure connections. Nevertheless, it only offers slow data rates of up to $2\,\mathrm{Mbit/s}$ and a limited range of up to $100\,\mathrm{m}$ in line of sight.

The other technology, Wi-Fi, is available in two ways: Wi-Fi Direct and Wi-Fi Aware. Wi-Fi Direct is used in the STAR and POINT-TO-POINT strategies and consists of a device providing the Wi-Fi network, requiring a centralized coordinator. Wi-Fi Aware is used in the POINT-TO-POINT strategy. It provides discovery and connection functions that allow the devices to be discovered and connected without any intermediaries, creating a decentralized peer-to-peer connection. The advantages of Wi-Fi over Bluetooth 5.0 include much higher data rates, which makes it ideal for high-bandwidth applications such as multimedia streaming and greater coverage ranges. The downsides are that Wi-Fi requires more power than Bluetooth 5.0 and is more prone to interference.

2.4. Security in D2D Communications

Nearby Connections API offers proximity-based services for Android and Android Things applications. This API allows users of the same application to share data when they are in close proximity, typically within Bluetooth radio range. The API provides Bluetooth classic, Bluetooth LE, and Wi-Fi links based on the user requirements. For short-range low-latency communication, Bluetooth is recommended, while Wi-Fi is recommended to be employed for medium-range high-bandwidth communication.

Proximity-based services, similar to the Nearby Connections API, have been studied extensively in the context of wireless sensor networks, along with their associated security challenges. Eavesdropping and wormhole attacks, which involve reading or manipulating the traffic between nearby devices, have been a hotlt debated topic. However, most existing research focuses on link or network layer attacks, such as routing or path finding protocols. Since Nearby Connections API operates at the application layer, it presents a different context compared to best practices in the field. However, similar security challenges apply, including the authentication of mobile users and the establishment of secure channels through key exchange protocols.

Overall, while the Nearby Connections API offers convenient proximity-based services, its closed-source nature and limited information availability raise concerns about its security. Further research and analysis are needed to assess and mitigate potential vulnerabilities and threats associated with the API's implementation.

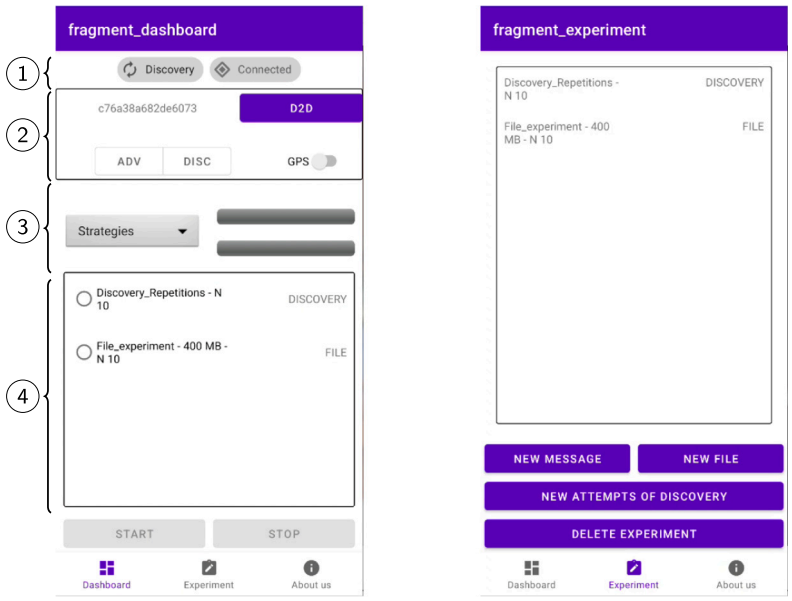


Fig. 3. On the left is the Dashboard fragment where the user can perform and visualize the experimentation procedure. This layout is subdivided into four layers. On the right is the Experiment fragment, where its layout allows the user to create, view and delete experiments allowed by the application.

3. Scrutinizing STAR and POINT-TO-POINT strategies

One of the problems that we faced when developing applications with Nearby Connections API is that the performance we observed diverged significantly from the theoretical numbers we found in the documentation. That is why we decided to conduct a number of experiments and share our insights with the research community. As we discussed in Section 2, only the STAR and POINT-TO-POINT strategies use Wi-Fi as a data transfer technology. Therefore, the question we address in the following is *what performance in terms of throughput to expect from these two strategies*?

To this end, we have developed an Android application called AtomD[2] that performs operations and procedures to exploit and stress all the functionalities provided by the Nearby Connections API. It gathers several measures that are recorded in a database for further analysis.

In terms of its interface, AtomD has two particular fragments for managing experiments, shown in Fig. 3.[3] The first one, called "Dashboard", allows the user to define the roles that a node can play in the D2D communication and allows the user to observe the status and progress of the connection. The second one, called "Experiments", allows the user to set the different actions that the application can take during the D2D transfers. We detail the two fragments in the following.

3.1. Dashboard

We structure the "Dashboard" fragment (the view on the left of Fig. 3) in four layers. In the first layer, we find the connectivity states during the realization of the D2D communication link. If a device performs NDP, it will change its `Discovery` state to reddish; otherwise, it will keep a gray color. Similarly, if a device maintains at least one D2D connection with another device, it will change its `Connected` status to reddish; otherwise, it will remain gray.

In the top left corner of the second layer is the device ID, which is used as a human readable name for other users who want to discover the device. Below the ID, there is a button that allows the user to configure the role of the device – `Disc.` to configure the device as a Discoverer and `Adv.` to configure the device as an Advertiser. Finally, on the right side of the second layer, there is a button to generate a D2D instance and a switch that allows tracking of the GPS coordinates of the device.

In the third layer, there is a drop-down list of the three strategies provided by the Nearby Connections API on the left side. In contrast, on the right side, there are 2 bars representing the progress during a D2D transfer. The first one is used to represent a single transmission, while the second one shows the progress of the set of transmissions if transfers are done in batches.

Finally, the fourth layer shows a list of predefined processes to measure D2D performance, of which only one can be executed. We explain how a user can define a process in Section 3.2.

It is important to note that the user must select at least the role and the strategy to be able to initiate a D2D instance.

[2] https://github.com/tlagos1/AtomD.

[3] In Android development, a fragment represents a modular portion of the user interface within an activity.

Algorithm 1: Chunk Experiment

```
Input: jp: Parameters to be sent, enclosed in a JSON object.
       id: The ID of the device connected through D2D.
       tr: Number of transmission repetitions.
Output: False in case the transfer was not successful. Otherwise, True.
Function SEND_CHUNK(jp, id, tr)
    P ← TO_PAYLOAD(jp)
    for r ← 1 To tr do
        if SEND_PAYLOAD(id, P) ≠ SUCCESS then
            return False
        end if
    end for
    return True
end
```

For each successful reception, we record the reception time and the received parameters. P is a variable containing a payload with the parameters to be sent.

Algorithm 2: File Experiment

```
Input: jp: File parameters, enclosed in a JSON object.
       fl: File to be send.
       id: The ID of the device connected through D2D.
       tr: Number of transmission repetitions.
Output: False in case the transfer was not successful. Otherwise, True.
Function SEND_FILE(jp, fl, id, tr)
    P ← TO_PAYLOAD(fl)
    for r ← 1 To tr do
        if SEND_CHUNK(jp, id, 1) = SUCCESS then
            if SEND_PAYLOAD(id, P) ≠ SUCCESS then
                return False
            end if
        else
            return False
        end if
    end for
    return True
end
```

During each file transfer, we log the number of bytes transferred, the time of their reception, and the corresponding file metadata chunk.

Algorithm 3: Connectivity Experiment

```
Input: rn: Readable name of the device connected through D2D.
       cr: Number of connections repetitions.
Output: In case it does not find an matching rn, then the device will keep listening to the channel.
Function CONNECTIVITY_REPETITIONS(rn, cr)
    for r ← 1 To cr do
        DISCONNECT_FROM(rn)
        START_DISCOVERY()
        if GET_DISCOVERED_DEVICE() = rn then
            ESTABLISH_CONNECTION_WITH(rn)
            STOP_DISCOVERY()
        end if
    end for
end
```

During each run of theNDP, we record when the discovery process was initiated, when the target device was discovered, when the node emits the connection request, and when the connection is established.

3.2. Experiment

To configure the behavior of the AtomD application and, consequently, of the D2D links, we have developed the "Experiment" fragment. The user can configure three different types of experiments (or processes), which we explain in the following.

- **Chunk experiment** (see Algorithm 1). It sends a set of data chunks with a maximum of 1 047 552 B per Chunk. Within the payload is set an instance ID of the experiment, the current GPS location of the device, and the counter tracking of the set of chunks that have been sent.
- **File experiment** (see Algorithm 2). It consists in performing a set of transmissions of the same file of size $n \times 10^m$ B to one or more devices. Before sending the file, the transmitter sends a Chunk with tracking data to track the file to be sent. Within this Chunk is set an ID of the experiment instance, the ID of the payload where the file is packed, the total number of repetitions

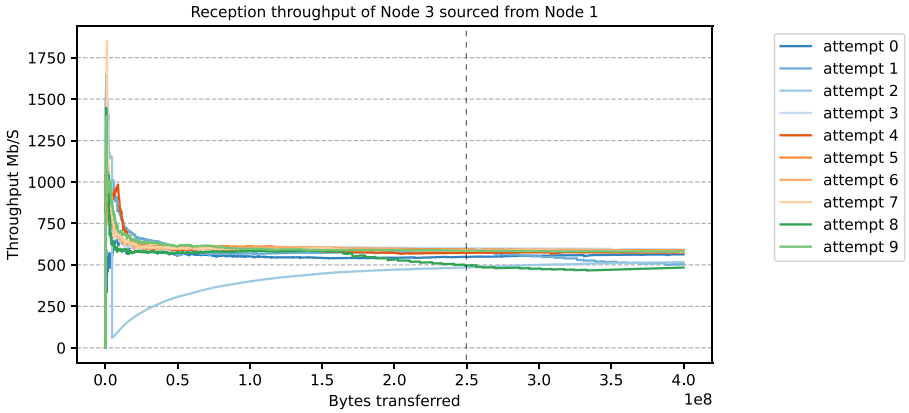


Fig. 4. Throughput of the receiver during the transfer of a file. Considering that the Nearby connection API splits the file into several chunks for transmission, the calculation of the throughput takes place from the beginning of the transfer until the total transferred byte count.

Table 1
Nodes' characteristics in our experiments.

Nodes	Processor	RAM	Android	Wi-Fi standard
1	Exynos 990	8 GB	12	802.11ax
2	Exynos 8895	4 GB	9	802.11ac
3	Snapdragon 835	8 GB	10	802.11ac
4	Kirin 970	6 GB	10	802.11ac
5	Snapdragon 662	4 GB	10	802.11ac

that the file will be sent, and the metadata of the transmitter. During the transmission process, the transmitter and receiver will record in a database the information registered within the Chunks, as well as the number of bytes expected and the bytes transferred for each sent file.

- **Connectivity experiment** (see Algorithm 3). This experiment allows the devices to perform a set of NDP. Among the configurations, the user can set the device to use either Bluetooth classic or BLE for the discovery and connectivity request procedures. In the course of this experiment, AtomD records the times of radio initiation, device discovery, request sending, and request reception in nanoseconds. In addition, it also records the GPS location of each device.

4. Experimental setup

We conducted a set of experiments to evaluate the performance of the STAR and POINT-TO-POINT strategies using five Android smartphones. The nodes were labeled as Node 1 to Node 5, and their characteristics are summarized in Table 1.

To ensure a strong and stable D2D wireless connection, we distribute these nodes no more than 20 cm away from each other. The distribution of the nodes consisted of placing one of them in the center while the other four smartphones were placed around, forming a square-shaped network topology. The Nearby Connections API version we used was 18.3.0.

To generate data, we configured our application to run a "File experiment" (see Section 3.2), in which the application generates and sends a 4×10^8 B file. It is important to note that the Nearby Connections API splits large files into chunks of no more than 1 047 552 B. However, the exact sizes of the chunks depend on an estimation of the available bandwidth at the time of the transmission. As a consequence, we must run the experiment for enough time to avoid any bias due to a temporary condition affecting the bandwidth. This behavior can be seen in the example shown in Fig. 4, where Node 3 received a file from Node 1.

Therefore, we saturated the sample set with a large number of chunks to calculate the throughput accurately and performed ten runs of the same transmission to satisfy the reliability interval of the data. We can see that as chunks are received, the chunk's saturation shows the actual throughput of the link. However, due to the random nature of wireless networks, each saturation point of each transmission may be at different zones. More specifically, while most of the saturation points of each transmission attempt converge to around 0.25×10^8 B, the saturation point of the third attempt reaches values of around 2.5×10^8 B.

5. Experimental methodology

To evaluate the performance of D2D communication using the Nearby Connections API, we performed a series of single-file transmissions using the STAR and POINT-TO-POINT strategies. As explained previously, we have chosen these two strategies because they use Wi-Fi as a communication medium (see Section 2). Furthermore, taking into account that each strategy uses a different topology, we elaborated an execution procedure in order to ensure fair testing.

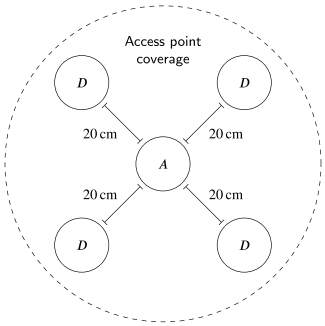


Fig. 5. Distribution of the nodes during the experimentation process. In the center of the distribution is located the node with the role of Advertiser (A), while on the sides are the Discoverers (D).

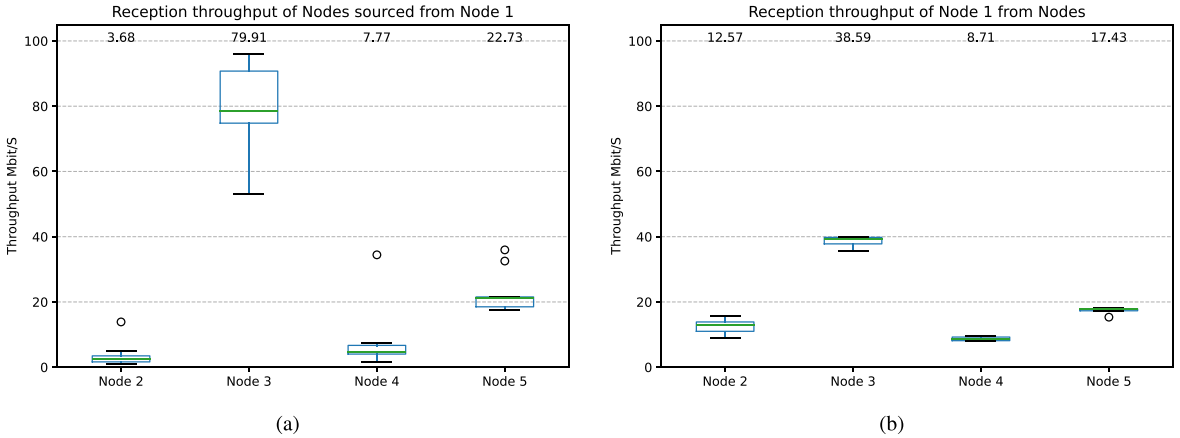


Fig. 6. Reception throughput of nodes in a STAR strategy with Node 1 acting as the access point. (a) displays the reception throughput of all nodes when Node 1 is transmitting, while (b) shows Node 1's reception throughput when other nodes are transmitting to it.

For the STAR strategy, we placed five nodes no more than 20 cm from each other, with the node configured with the Advertiser role in the center and the other nodes set as Discoverers around, as shown in Fig. 5. Such a topology facilitates the Advertiser device to act as an access point. Note that the Advertiser maintains a connection with all the Discoverers. The Advertiser then chooses one of the Discoverers to perform the predefined "File Experiment", starting with the node in the top right corner and proceeding clockwise to the other nodes. It is worth underlying that the "File Experiment" process is bidirectional, with the Advertiser initially acting as the transmitter (data: Advertiser → Discoverer) and then as a receiver (data: Advertiser ← Discoverer). Once the device picked as chosen Advertiser has communicated with all the Discoverers, all nodes' databases are exported and reset. We choose then another device as the Advertiser and run the same process until all combinations have been tested.

Similarly, for the POINT-TO-POINT strategy, we maintained the same node distribution as in the STAR strategy, with the Advertiser in the center. The Advertiser then establishes a single connection to a Discoverer, one at a time, following the order explained in the STAR strategy, and then performs the execution of the experiment. Contrarily to the STAR strategy, once the Advertiser has completed all possible combinations, the databases of all nodes are exported and reset. Then, a new Advertiser is established, and the process is restarted.

Through these tests, we evaluated the throughput during the transmission of a file to another node, providing insights into the performance that D2D communication can maintain under different strategies.

6. Throughput analysis

We start by analyzing the throughput of both the STAR and POINT-TO-POINT strategies. To this end, we perform ten transmissions per source and per destination, shown as boxplots. In Figs. 6–10, we show the observations for the STAR strategy, while in Figs. 11 to 15, we show the number for POINT-TO-POINT. Each boxplot represents data from the first ($Q1$) to the third ($Q3$) quartile, as well as the median. We also show the mean as a number at the top of the plot. Whiskers show the data range and extend no more than $1.5 \times (Q3 - Q1)$ from the box's edges, ending at the furthest data point within that range. Outliers appear as circles.

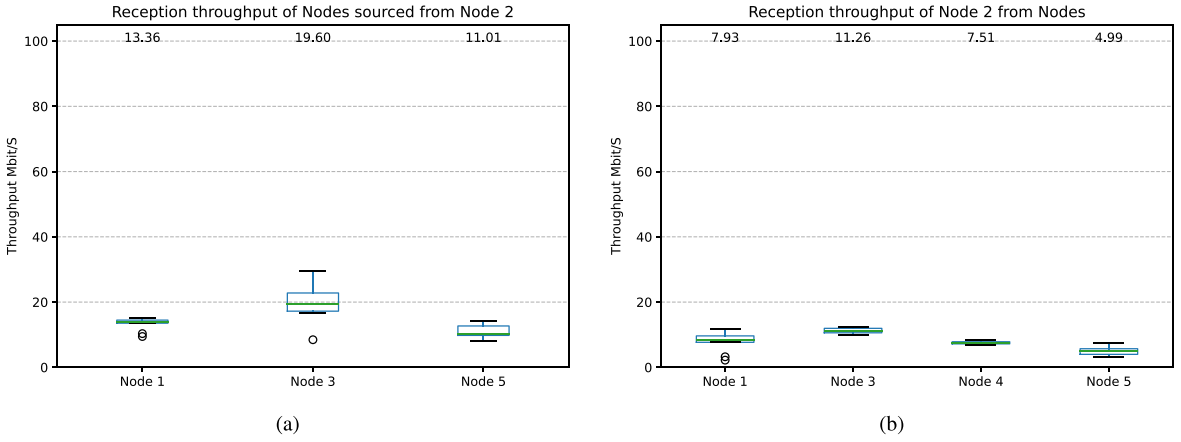


Fig. 7. Reception throughput of nodes in a STAR strategy with Node 2 acting as the access point. (a) displays the reception throughput of all nodes when Node 2 is transmitting, while (b) shows Node 2's reception throughput when other nodes are transmitting to it.

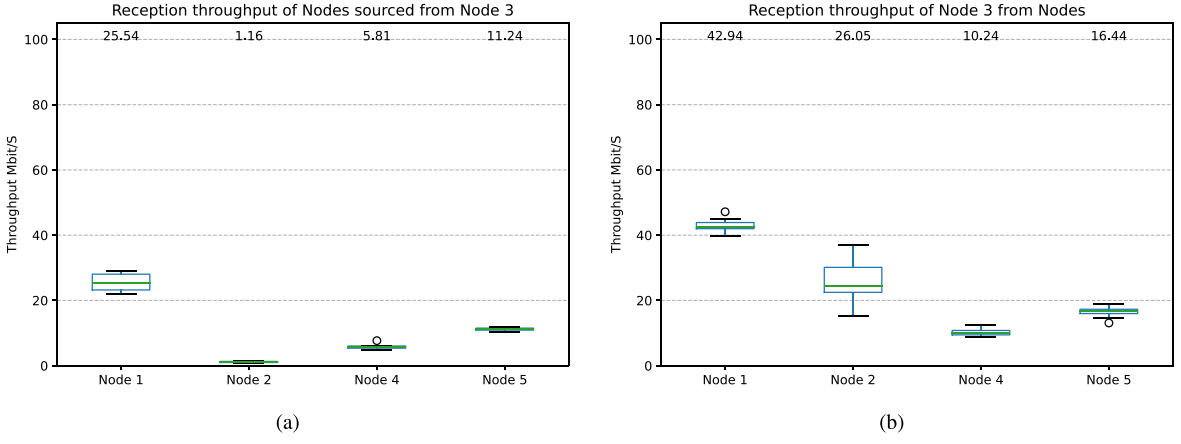


Fig. 8. Reception throughput of nodes in a STAR strategy with Node 3 acting as the access point. (a) displays the reception throughput of all nodes when Node 3 is transmitting, while (b) shows Node 3's reception throughput when other nodes are transmitting to it.

These plots represent the throughput obtained during the reception of the file. More specifically, the plot on the left shows the throughput obtained by the Advertiser's neighbors, i.e., the throughput reception of the nodes connected to the access point node. In contrast, the plot on the right shows the Advertiser's reception throughput when each neighbor was transmitting to the Advertiser, i.e., the throughput reception of the access point from its connected devices. In each plot, the x-axis corresponds to the node the Advertiser interacted with (see Section 4), and the y-axis corresponds to the throughput in Mbit/s obtained during the transfers.

6.1. STAR strategy

Based on the results obtained using Node 1 as the Advertiser (Fig. 6), we found that it lacked consistent throughput performance across all nodes. In fact, in both Figs. 6(a) and 6(b), it can be observed that the link between Node 1 and Node 3 has a better throughput than the other nodes, while the link performance of the rest of the nodes remains below the 20 Mbit/s. On top of that, if we compare the results of both Figs. 6(a) and 6(b), we can observe that Node 1 has a better throughput performance when acting as a receiver (Fig. 6(b)).

In the case of Node 2 as the Advertiser (Fig. 7), we can observe that its throughput performance is significantly lower than in the case of Node 1. Moreover, as seen in Fig. 7(a), we could only collect data with three out of the four nodes because there were constant link interruptions between Node 2 and Node 4 during the file transmission. Another feature is that Node 3 maintains a better link performance than the rest of the other nodes. Finally, Node 2 performed better as a transmitter (Fig. 7(a)) than as a receiver (Fig. 7(b)). However, given its instability in maintaining a link while transmitting, we must determine if this performance is realistic.

Taking Node 3 as the Advertiser (Fig. 8), we can observe that the link formed with Node 1 performs better than the other nodes. Similarly, Node 3 performs better as a receiver (Fig. 8(b)), as was observed when Node 1 acts as an Advertiser (Fig. 6).

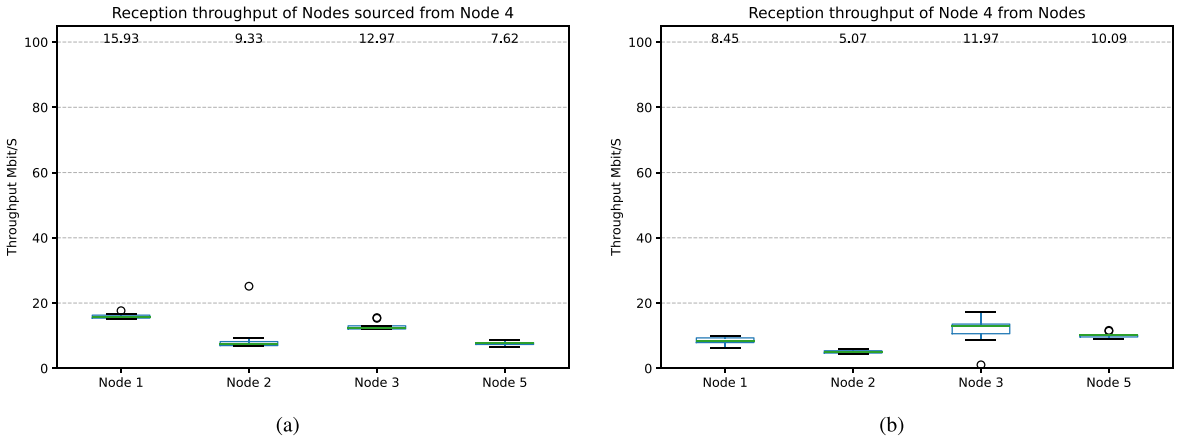


Fig. 9. Reception throughput of nodes in a STAR strategy, with Node 4 acting as the access point. (a) displays the reception throughput of all nodes when Node 4 is transmitting, while (b) shows Node 4's reception throughput when other nodes are transmitting to it.

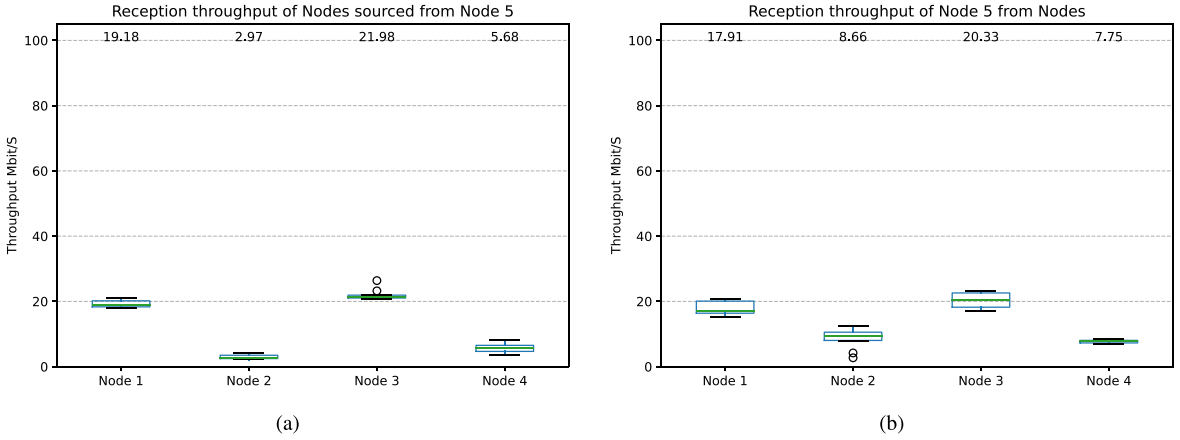


Fig. 10. Reception throughput of nodes in a STAR strategy with Node 5 acting as the access point. (a) displays the reception throughput of all nodes when Node 5 is transmitting, while (b) shows Node 5's reception throughput when other nodes are transmitting to it.

As for Node 4 as the Advertiser (Fig. 9), it maintains a low performance while maintaining relative throughput uniformity. Furthermore, compared to Node 2 as Advertiser Fig. 7), Node 4 performs slightly better.

Finally, when Node 5 is taken as the Advertiser (Fig. 10), it can be observed that its performance is better using Node 1 and Node 3. As for its links using Node 2 and Node 4, respectively, acting as receivers, the respective links perform better (Fig. 10(b)).

Summary. When applying the STAR strategy, Node 1 and Node 3 stood out, while the throughput performance was too low with Node 2 and Node 4. The best performance that we could observe varies between 30 Mbit/s and 40 Mbit/s, except for the 79.91 Mbit/s shown in Fig. 6(a). In many cases, however, the value can be below 10 Mbit/s, which is relative regard to the performance that we could expect from Wi-Fi links. The question we will address in the following is whether we can obtain higher throughputs when adopting the POINT-TO-POINT strategy and, if so, how much.

6.2. *POINT-TO-POINT strategy*

From the results obtained with Node 1 as the Advertiser (Fig. 11), we can see that its throughput is higher when it acts as a transmitter (Fig. 11(a)). Furthermore, for each link generated, the throughput performance of each node doubled when Node 1 acted as a transmitter. Also, Node 1 has a better throughput with its link with Node 3 than the rest.

In the case where Node 2 acts as the Advertiser (Fig. 12), we can observe a sharp drop in throughput, keeping its values below 70 Mbit/s. Regarding performance, we can observe that Node 1 and Node 3 stand out, doubling the throughput of Nodes 4 and 5.

When Node 3 is the Advertiser (Fig. 13), we observe that the link it maintains with Node 2 has a better throughput performance than the rest of the other links. In fact, if we compare its results with those obtained using Node 1 as Advertiser (Fig. 11) and Node 2 as Advertiser (Fig. 12), we should expect a better performance with Node 1. Another observable characteristic in Fig. 13 is that it

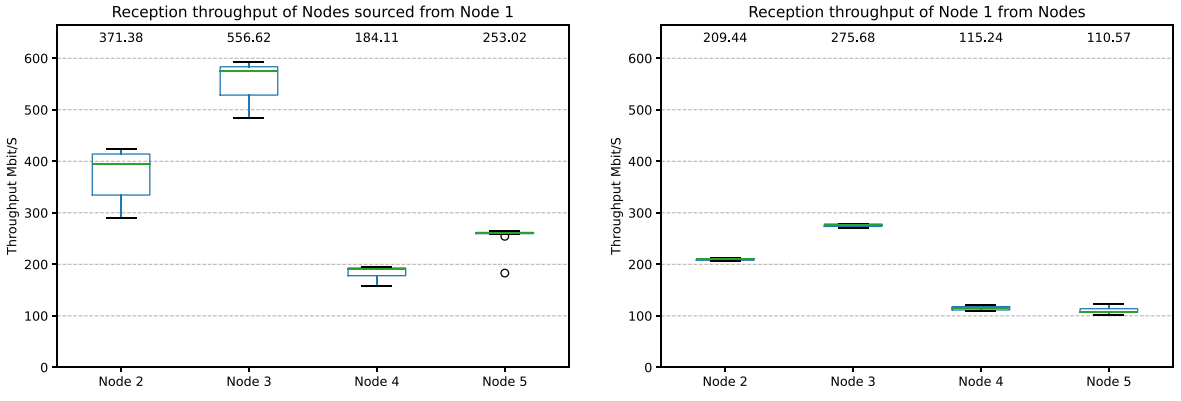


Fig. 11. Reception throughput of nodes in a POINT-TO-POINT strategy with Node 1 acting as the access point. (a) displays the reception throughput of all nodes when Node 1 is transmitting, while (b) shows Node 1's reception throughput when other nodes are transmitting to it.

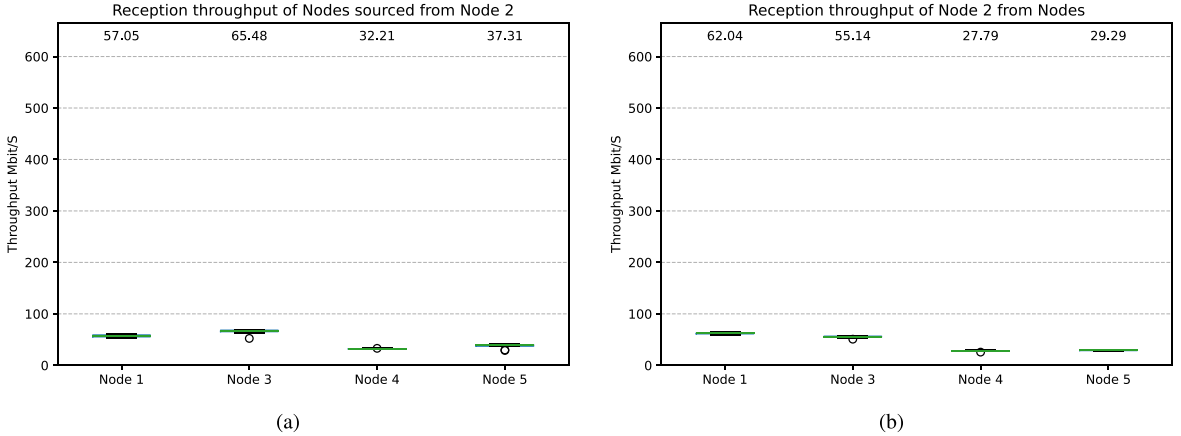


Fig. 12. Reception throughput of nodes in a POINT-TO-POINT strategy with Node 2 acting as the access point. (a) displays the reception throughput of all nodes when Node 2 is transmitting, while (b) shows Node 2's reception throughput when other nodes are transmitting to it.

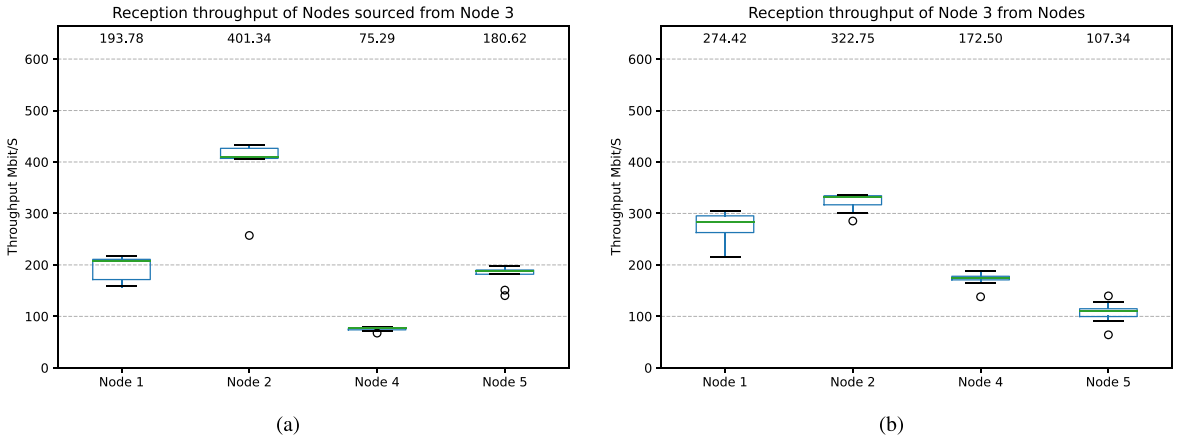


Fig. 13. Reception throughput of nodes in a POINT-TO-POINT strategy with Node 3 acting as the access point. (a) displays the reception throughput of all nodes when Node 3 is transmitting, while (b) shows Node 3's reception throughput when other nodes are transmitting to it.

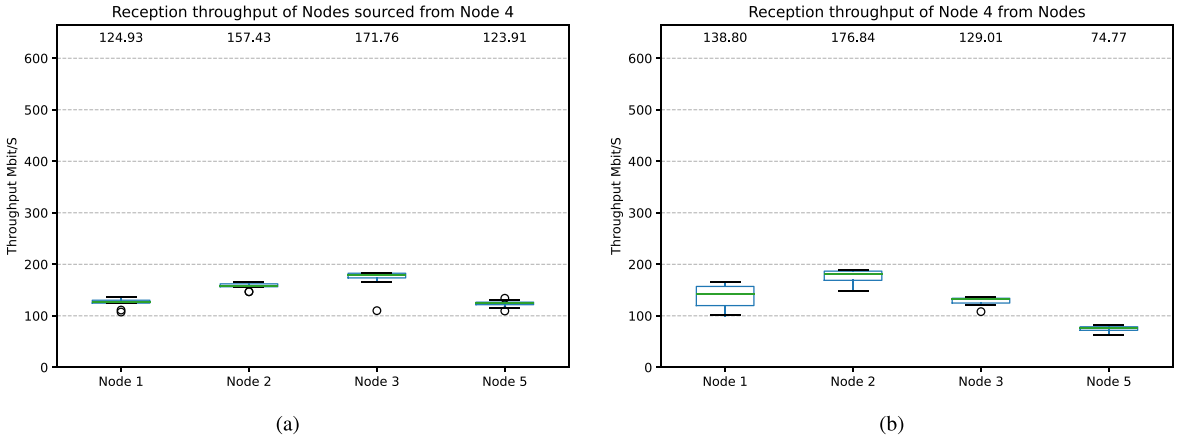


Fig. 14. Reception throughput of nodes in a POINT-TO-POINT strategy with Node 4 acting as the access point. (a) displays the reception throughput of all nodes when Node 4 is transmitting, while (b) shows Node 4's reception throughput when other nodes are transmitting to it.

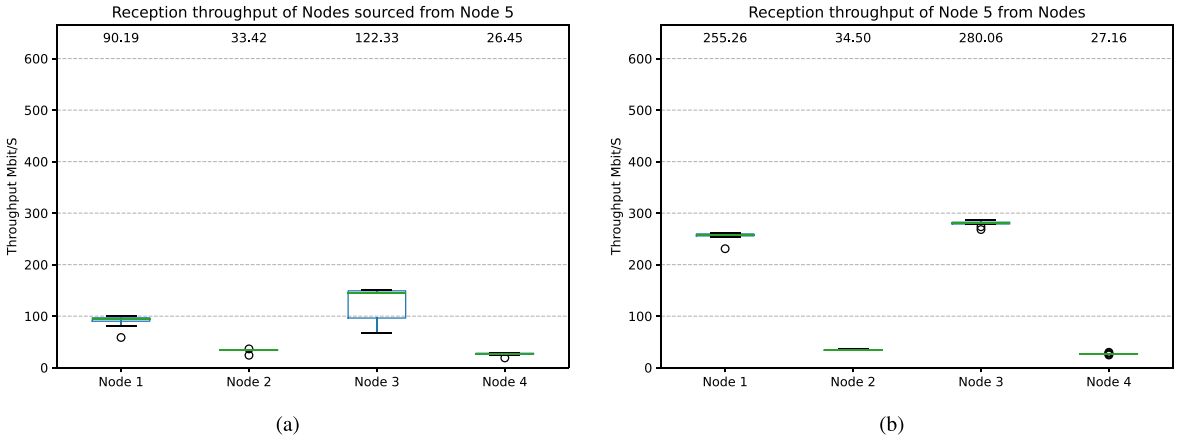


Fig. 15. Reception throughput of nodes in a POINT-TO-POINT strategy with Node 5 acting as the access point. (a) displays the reception throughput of all nodes when Node 5 is transmitting, while (b) shows Node 5's reception throughput when other nodes are transmitting to it.

did not maintain a throughput pattern when Node 3 acted as the transmitter or receiver. This can be observed with Nodes 4 and 5; Node 4 stands out in Fig. 13(a), while Node 5 stands out in Fig. 13(b). On the other hand, the throughput values of Nodes 1 and 2 are similar when Node 3 acts as a receiver.

Concerning Node 4 as the Advertiser (Fig. 14), this node maintains a low throughput while maintaining a relative uniformity in performance. Furthermore, compared to the performance using Node 2 as the Advertiser, Node 4 performs slightly better than Node 2 (Fig. 12).

Finally, when Node 5 is taken as Advertiser (Fig. 15), it is observed that its performance is better using Node 1 and Node 3. As for their links using Node 2 and Node 4 as transmitters, respectively, the respective links perform better (Fig. 15(b)).

Summary. In the POINT-TO-POINT strategy, we observed that two nodes (Nodes 1 and 3) experienced much better performance than the others when they acted as Advertisers, reaching values of up to 600 Mbit/s. Clearly, as we could see in all experiments, the POINT-TO-POINT strategy definitely leads to much better performance than STAR. This brings food-for-thoughts that we present in Section 7.

7. Discussion

After reviewing the results obtained with the STAR (see Section 6.1) and POINT-TO-POINT (see Section 6.2) strategies, it is evident that the POINT-TO-POINT strategy outperforms the STAR strategy by far. Specifically, the maximum throughput achieved with STAR was 79.91 Mbit/s, while the POINT-TO-POINT strategy achieved maximum throughputs close to 600 Mbit/s. This is because the Nearby Connections API probably uses a best-effort mechanism, depending on the topology of each strategy.[4]

[4] Unfortunately, Google does not disclose all the implementation details of the Nearby Connections API.

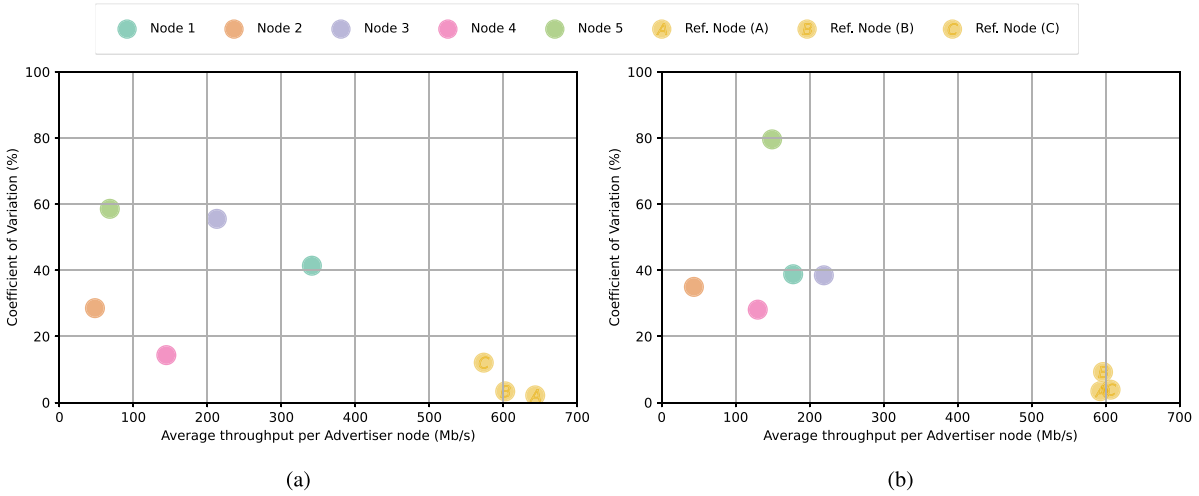


Fig. 16. Coefficient of variation versus average throughput of the throughput set corresponding to each node acting as Access Point using the POINT-TO-POINT strategy. Panel (a) shows the results for the sets where the access point transmits to connected nodes, while panel (b) shows the results for the sets where the connected nodes transmit to the access point.

Taking a closer look at the individual performance of Advertisers, Nodes 1 and 3 showed consistent superiority in both strategies. These nodes performed best as access points. This highlights the importance of access point selection in facilitating effective direct communication between nodes. In contrast, Node 2 demonstrated the worst throughput performance in most cases, generating unstable links during its time as an access point. Furthermore, it was observed that the throughput fluctuation between links from the same access point in both strategies was more pronounced at Node 2 and Node 3, probably due to the links' low throughput. This underlines the need to investigate the factors contributing to these performance differences. Finally, Node 5 maintained better-performing links when connected to Nodes 1 and 3 in both strategies, while it demonstrated lower throughput when connected to Nodes 2 and 4. This highlights the importance of considering the individual characteristics of each node when determining the optimal communication strategy.

7.1. Correlation between strategies

The results of this study provide insights into the performance of the POINT-TO-POINT and STAR strategies. On the one hand, we find that the POINT-TO-POINT strategy exhibited high throughput performance in most cases, making it suitable for use cases involving high network traffic, such as file transfer or actions requiring high traffic flow. On the other hand, the STAR strategy exhibited much lower throughput performance, leading to link instability and slow transfers in some instances, making it recommended for communication use cases with low network traffic density.

To further illustrate the difference in transfer time between the two strategies, the study considered a binary file of size 4×10^8 B. Based on the results, the estimated transfer time for the STAR strategy was 74.52 s, while it was around 8 s for the POINT-TO-POINT strategy. This is a significant difference that needs to be taken into account when setting up a network on top of the Nearby Connections API.

Another critical factor to consider in the D2D links generated by the Nearby Connections API is their asymmetric nature. As observed in Figs. 11 to 15, the throughput performance of the links is highly dependent on the node set as the access point, as well as on the node to which it is connected. This dependency is further highlighted in Fig. 16, where the coefficient of variation of each set belonging to the POINT-TO-POINT strategy is compared with the average throughput corresponding to these sets. The study also includes three reference nodes with the same parameters, which maintain a low coefficient of variation due to their similar hardware characteristics, resulting in better throughput performance. In contrast, the other nodes had much higher coefficients of variation, which leads to lower throughput, more representative of the real case scenario.

Overall, these findings have important implications for device-to-device communication, as they suggest that the POINT-TO-POINT strategy provides higher performance and throughput, making it the preferred choice for use cases with high network traffic density. The study also highlights the importance of considering the hardware characteristics of the nodes when selecting the access point to ensure optimal throughput performance.

8. Related work

Several studies have focused on improving the accessibility and performance of wireless networks based on Wi-Fi Direct and related communication protocols.

Table 2
Comparative table of related work based on throughput performance evaluation, D2D characterization, D2D extension for throughput performance, and throughput performance-based D2D experimental work.

Work	Area of work			
	Performance evaluation	Characterization	Extension	Experimental work
[9]	✓	✗	✓	✗
[18]	✓	✓	✗	✓
[19]	✓	✓	✓	✗
[20]	✓	✓	✗	✗
[21]	✓	✗	✓	✓
This work	✓	✓	✗	✓

Adam et al. the authors analyzed the accessibility and performance of Wi-Fi and cellular networks in Rochester, NY, compared to multi-hop ad hoc networks based on Wi-Fi Direct [18]. They showed that, although Wi-Fi access points were widely available, more than 20% of locations had inaccessible access points due to security restrictions. Moreover, the LTE cellular network led to higher download speeds when compared to Wi-Fi and multi-hop D2D networks, while Wi-Fi and cellular upload speeds were comparable. The authors concluded that extending access to the Internet through multi-hop D2D connections is feasible but comes at the expense of a 62% reduction in upload and a 64% reduction in download speeds.

Funai et al. proposed and analyzed different practical solutions for supporting communications between multiple Wi-Fi Direct groups using Android devices [19]. Firstly, they listed several limitations of the current implementation of the Android Wi-Fi Direct framework and presented possible solutions to interconnect different groups to create multi-hop ad hoc networks. The authors found that their proposed approaches were feasible with different overheads regarding energy consumption and delays at the gateway node. Their experimental results demonstrated the superiority of techniques that exploit the device's ability to maintain simultaneous physical connections to multiple groups, enabling multi-hop ad hoc networks at low overhead.

Tan et al. proposed a multi-hop transmission scheme that combines 5G and Wi-Fi Direct to improve the transmission performance of edge nodes [9]. Furthermore, they built a 5G+ Wi-Fi D2D system-level simulation platform to test the scheme's performance. The simulation results showed that the proposed scheme can effectively improve the throughput and reduce the packet loss rate of edge nodes. Moreover, the study also shows that the number of routing hops and new radio signal thresholds can affect the transmission performance of the node.

The work from Bertier et al. reported on an empirical characterization of Wi-Fi P2P and Google Nearby, two D2D technologies in Android [20]. To do so, the authors developed a custom Android application, called Ocat, which measures the link's goodput between Android devices. Then, using the wireless signal strength as a link measurement, they combined it with the two-ray ground-reflection model to infer the goodput and obtain a good fit for characterizing D2D links between Android devices. Their findings provided a reality check regarding the direct data-exchange capabilities of Android devices and helped assess the system performance of D2D applications.

Another interesting work is the one by Li et al. who implemented and evaluated the performance of a Wi-Fi Direct (WFD)-based local communication system [21]. They proposed an application-layer forwarding solution for inter-group communication and a self-adaptive handover mechanism taking user mobility and node failures into account. They utilized a fuzzy-logic-based normalized quantitative decision algorithm (FNQD) with the weights derived from the fuzzy analytic hierarchy process (FAHP) to deal with the uncertainty in the handover decision procedure. The authors evaluated the performance of the system through both simulation and experiment analysis. Their results showed that they could achieve a maximum throughput of 31.7 Mb/s for intra-group communication and a maximum goodput of 4.76 Mb/s for inter-group communication. Moreover, mobile devices could perform various types of handover according to their roles and status, which could improve the robustness of the local communication system.

All of the previous works, as well as ours, intend to help the research community dare experimental work related to device-to-device communications. We all address specific issues that either prevent efficient implementation or are too complicated to sort out. In our case, we contribute with a better understanding of what to expect in terms of throughput over the Nearby Connections API. We hope that other researchers will share the lessons they learned when implementing their solutions (see Table 2).

9. Conclusion and perspectives

In this study, we have extensively evaluated throughput performance on D2D links using the Nearby Connections API over POINT-TO-POINT and STAR strategies. The results show that the POINT-TO-POINT strategy presents higher performance in most cases, making it suitable for use cases involving high network traffic density. This is especially useful for actions that require high traffic flow, such as file transfer. On the other hand, the STAR strategy shows much lower throughput performance, leading to link instability and slow transfers in some cases. It is therefore recommended for communication use cases with low network traffic density. In addition, the study highlighted the significant difference in transfer time between the two strategies.

Another critical factor affecting the D2D links generated by the Nearby Connections API is their asymmetric nature. Our results demonstrate that the performance of the links is highly dependent on the node established as the access point and the node to which it is connected. This dependency is further highlighted in Fig. 16, where the coefficient of variation of each set belonging to the POINT-TO-POINT strategy was compared with the average throughput corresponding to these sets.

The study's results also indicate the importance of considering the nodes' hardware characteristics when selecting the access point to ensure optimal throughput performance. The study added three reference nodes with similar parameters, which maintained a low coefficient of variation due to their similar hardware characteristics, resulting in better throughput performance. In contrast, other nodes had much higher coefficients of variation, resulting in worse throughput performance, representing the real case scenario. Overall, these results have important implications for communication between nodes. They suggest that the POINT-TO-POINT strategy provides higher throughput performance, making it the preferred choice for use cases with high network traffic density. On the other hand, the STAR strategy can be used in low-traffic density scenarios, but its performance is limited. Therefore, it is crucial to consider the asymmetric nature of D2D links and the hardware characteristics of the nodes when selecting the access point to ensure optimal throughput performance.

A network architect or application developer can now refer to the numbers we provide to decide which strategy fits better their solutions, or even run their own experiments using our methodology and open-source tool. As future work, we will address other metrics such as connection establishment delay and battery consumption. In addition, future research could explore ways to improve link stability in Nearby Connections API, such as by implementing adaptive algorithms or optimizing network protocols to cope with the variability of the environment and maintain consistent throughput.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the code in a git repository at https://github.com/tlagos1/AtomD.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 957246.

References

[1] Rafay Iqbal Ansari, Chrysostomos Chrysostomou, Syed Ali Hassan, Mohsen Guizani, Shahid Mumtaz, Jonathan Rodriguez, Joel J.P.C. Rodrigues, 5G D2D networks: Techniques, challenges, and future prospects, IEEE Syst. 12 (2018) 3970–3984.
[2] Mittal Pedhadiya, Rakesh Jha, Hetal Bhatt, Device to device communication: A survey, Netw. Comput. Appl. 129 (2019) 71–89.
[3] Mohsen Nader Tehrani, Murat Uysal, Halim Yanikomeroglu, Device-to-device communication in 5G cellular networks: challenges, solutions, and future directions, IEEE Commun. Mag. 52 (2014) 86–92.
[4] Mengjun Yin, Wenjing Li, Lei Feng, Peng Yu, Quesong Qiu, Emergency communications based on throughput-aware D2D multicasting in 5G public safety networks, Sensors (Basel) 20 (2020) 1–18.
[5] Mohamed Rihan, Mahmoud Selim, Chen Xu, Lei Huang, D2D communication underlaying UAV on multiple bands in disaster area: Stochastic geometry analysis, IEEE Access 7 (2019) 156646–156658.
[6] Enver Ever, Eser Gemikonakli, Huan Nguyen, Al-Turjman. Fadi, Adnan Yazici, Performance evaluation of hybrid disaster recovery framework with D2D communications, Comput. Commun. 152 (2020) 81–92.
[7] Milan Stute, David Kreitschmann, Matthias Hollick, One billion apples' secret sauce: Recipe for the apple wireless direct link Ad Hoc protocol, in: Proceedings of the International Conference on Mobile Computing and Networking (MobiCom), New Delhi, India, 2018.
[8] Daniel Marcos Schwaycer, Instantly share files with people around you with nearby share, 2020, https://blog.google/products/android/nearby-share-windows/.
[9] Siying Tan, Chao Shen, Rongtao Xu, Rui Shi, Xuemei Wang, Dengtao Zhang, Bo Ai, Performance evaluation of 5G NR traffic offloading onto WiFi direct, in: Proceedings of the IEEE International Conference on Computer and Communications (ICCC), Chengdu, China, 2022.
[10] Aymen Zreikat, Performance evaluation of 5G/WiFi-6 coexistence, Circuits Systems Signal Process. 14 (2020) 903–913.
[11] Praful D. Mankar, Zheng Chen, Mohamed A. Abd-Elmagid, Nikolaos Pappas, Harpreet S. Dhillon, Throughput and age of information in a cellular-based IoT network, IEEE Trans. Wireless Commun. 20 (2021) 8248–8263.
[12] Google, Strategies, 2022, https://developers.google.com/nearby/connections/strategies.
[13] Google, Nearby Connections – API Overview, 2021, https://developers.google.com/nearby/connections/.
[14] Tomás Jenschke Lagos, Marcelo Dias de Amorim, Serge Fdida, Quantifying direct link establishment delay between android devices, in: Proceedings of the IEEE Conference on Local Computer Networks (LCN), Edmonton, AB, Canada, 2022.
[15] StackOverflow, Can we get high bandwidth in cluster strategy as we get in star strategy of google nearby connections ?, 2018, https://stackoverflow.com/a/51229552.
[16] StackOverflow, Google nearby connections 2.0 capabilities, 2018, https://stackoverflow.com/a/51997757.
[17] StackOverflow, Android nearby connection send payload partially wifi aware and bluetooth, 2022, https://stackoverflow.com/a/71960719.
[18] Nadir Adam, Cristiano Tapparello, Wendi Heinzelman, Infrastructure vs. Multi-Hop D2D networks: Availability and performance analysis, in: Proceedings of the International Conference on Computing, Networking and Communications (ICNC), Honolulu, Hawaii, USA, 2019.
[19] Colin Funai, Cristiano Tapparello, Wendi Heinzelman, Enabling multi-hop ad hoc networks through WiFi direct multi-group networking, in: Proceedings of the International Conference on Computing, Networking and Communications (ICNC), Silicon Valley, USA, 2017.
[20] Clément Bertier, Marcelo Dias de Amorim, Farid Benbadis, Vania Conan, Modeling realistic bit rates of D2D communications between android devices, in: Proceedings of the International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), Miami Beach, USA, 2019.
[21] Fuliang Li, Xingwei Wang, Zijian Wang, Jiannong Cao, Xuefeng Liu, Yuanguo Bi, Weichao Li, Yi Wang, A local communication system over Wi-Fi direct: Implementation and performance evaluation, IEEE Internet Things 7 (2020) 5140–5158.