DeToks: Decentralised Social Media using 5G NAT puncturing

Orestis Kanaris
Delft University of Technology
Delft, Netherlands
O.Kanaris@student.tudelft.nl

Johan Pouwelse (MSc Supervisor)
Delft University of Technology
Delft, Netherlands
J.A.Pouwelse@tudelft.nl

Abstract—We present a decentralised alternative to the winner-takes-everything dynamics of social media platforms. For 25 years there have been continuous attempts to decentralise file sharing, music streaming, video conferencing, and social media. None of these hundreds of projects to *re-decentralise* the Internet have reached the uptake level of YouTube and TikTok. We present DeToks, a fully decentralised alternative to Youtube and Tiktok. DeToks is not dependant on any central server or cloud. DeToks is specifically designed to be as decentralised and attack-resilient as Bitcoin and Bittorrent.

Our core contribution is effortless 5G Network Address Translators (NATs) puncturing. Direct phone-to-phone communication is not available on smartphones available today. DeToks solves this problem.

NATs and carrier-grade NATs block direct communication between smartphones. We procured numerous SIMs card on European 4G/5G mobile networks and measured the carrier-grade NATs behavior. We determined the NAT types (full cone,restricted,symmetric) and their time-out settings. By leveraging provider-aware (Vodafone,Orange,Telia, etc.) NAT puncturing strategies we create direct UDP-based phone-to-phone connectivity. By relying on provider-aware IPv4 range allocations, provider-aware port prediction heuristics, high bandwidth probing, and the birthday paradox we can successfully bypass even symmetric NATs. Our communication method achieves peer-to-peer 5G connectivity at the cost of merely some initial delay and bandwidth, without any assistance from third party servers or clouds.

Detoks validates our 5G puncturing work. We demonstrate the feasibility of fully decentralized social media platforms on consumer mobile devices.

Index Terms—NAT, CGNAT, 5G, Distributed Social Media, NAT puncture

I. Introduction

We present DeToks, meticulously designed to empower Internet users to take full control of their social media experiences. DeToks is a decentralised TikTok alternative. Our proof-of-principle social media app does not require any servers, avoids using any cloud, bypasses the need for any legal entity, and abstains from any centrality in general. Relentless improvements in mobile hardware now enable on-device alternatives for the cloud. DeToks offers a fully decentralised swipe-based media experience using BitTorrent and *5G NAT puncturing*.

5G NAT puncturing is a technique to communicate freely on The Internet. Personal devices, specifically smartphones,

communicate through home Wi-Fi and mobile networks like 4/5G. Using these networks, the devices usually end up behind a home NAT or a Carrier-Grade NAT (CGNAT). These devices divide scarce IPv4 Internet addresses amongst active users. However, existence of these NATs makes it hard for the devices to communicate with each other since they lock their discoverability by hiding the devices behind the NAT's private network, forcing the "NATed" device to initiate the connection first. This is not a particularly impossible problem if one of the two peers has a static IP address and is discoverable. It is particularly bad when both peers are behind NATs (even worse when it is the same NAT, a problem common with CGNATs [1]), then both need to initiate the connection first, but none of them is "visible" to the other.

The novel contribution of this work is bypassing the carrier-grade NAT hardware inside 4G and 5G networks. The need for this came when the depletion of IPv4 addresses and lack of cybersecurity "forced" 5G network operators to violate Internet protocols. DeToks shows that with advanced 5G NAT puncturing it is possible to create fully decentralised social media.

II. Problem Description

The challenge is to overcome the restricted communication in 5G networks. These mobile networks are exclusively designed to communicate with the cloud. Firstly, this restricts user autonomy and choice, as individuals are often confined to the services and applications approved by the smartphone manufacturers, limiting their ability to access alternative, potentially more innovative or privacy-respecting options [2]. This closed nature also fosters a dependency on a few dominant tech companies, which can lead to monopolistic practices, reduced competition, and a lack of diversity in the market [3]. Furthermore, the exclusive reliance on cloud communication raises significant privacy concerns, as vast amounts of personal data are continuously sent to and stored in centralized servers, making it susceptible to data breaches, surveillance, and misuse by third parties.

All the problems mentioned above are spawned from the design choice of smartphones not being able to communicate directly with each other, making any peer-to-peer application virtually impossible to run on a smartphone using conventional approaches.

Since 2017[1] Delft University of Technology has successfully expanded its research on decentralised 5G overlay network. See the Android devices participating in a peer-to-peer-based network overlay in figure 1. This pure-phone network is bootstrapped by a smartphone with a well-known IPv4 address.
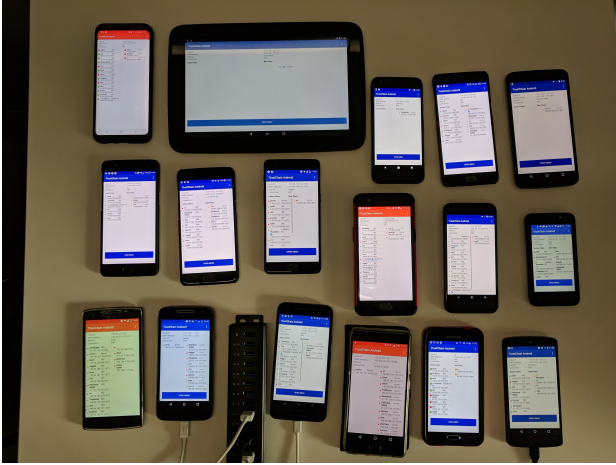

Fig. 1: IPv8 in 2017 running on multiple Android devices

Until now, brute force network flooding-based NAT puncturing is the only way to overcome device connectivity issues for devices behind NATs and firewalls. Although some improvements were proposed, i.e. birthday-paradox-based network flooding [4], to our knowledge, no one utilizes insights into the LAN hardware used, specifically the ones used by cellular providers, to maximize the connectivity success rate.

Two studies were conducted more than a decade ago [5], [6], to understand how the infrastructure of cellular providers work and how this affects the cellular network behaviour and performance; their findings are now outdated due to newer generations of cellular networks becoming the new de-facto.

Establishing a connection between devices is essential to enable peer-to-peer distributed social media. Key communication parameters, such as NAT type, timeouts, and maximum UDP packet size, are critical for maintaining the connection and selecting appropriate connectivity strategies. However, cellular providers do not publicly disclose these parameters.

Anderson's use of a Birthday attack to minimize packet exchanges showed limited success with Dutch cellular providers, as discussed in Section III-D. Consequently, an in-depth analysis of the NATs employed by these providers was conducted to improve connectivity rates by exploring a method for peer-to-peer communication via randomized packet exchanges to achieve a successful connection.

Our solution is a standalone open-source Kotlin library designed using insights from analyzing various European cellular providers. It is evaluated as a standalone library and used in a decentralised TikTok on Android mobile phones.

[1] https://github.com/Tribler/tribler/issues/3237

III. ARCHITECTURE AND IMPLEMENTATION

Figure 1 shows two smartphones running our DeToks application. We pioneered a new level of decentralisation by crafting the first 5G phone-to-phone network. No prior scientific work offers the same level of decentralisation as DeToks, to the best of our knowledge.

The software used to gather the cellular-specific parameters mentioned in this chapter is available on GitHub [7]. Two phones running the decentralized DeToks app using CytaMobile-Vodafone's 4G are shown in figure 2.
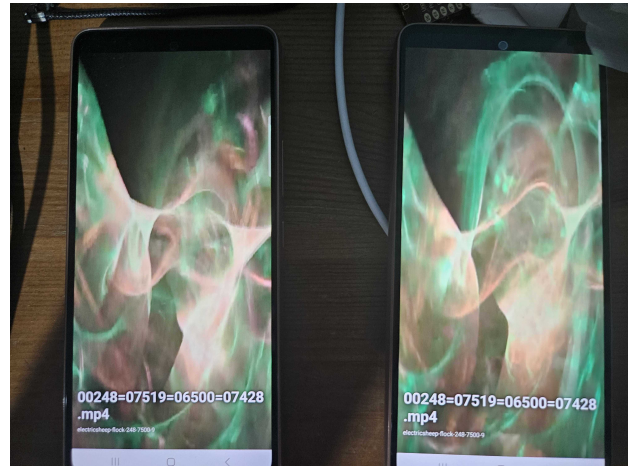

Fig. 2: Two Android phones running decentralized TikTok on cellular data after Nat puncturing

A. NAT Types

The STUN protocol (RFC3489 [8]) classifies NATs into four types: Full-cone NAT, Restricted-cone NAT, Port-restricted cone NAT, and Symmetric NAT. RFC4787 [1] refines this categorization into "easy" NATs using Endpoint-Independent Mapping (EIM) and "hard" NATs using Endpoint-Dependent Mapping (EDM). EIM ensures consistent external address and port mapping for requests from the same internal port.

According to V. Paulsamy et al. [9], the specifications for these NAT types are:

- **Full-Cone NAT:** An EIM NAT that maps all requests from a single internal IP to a corresponding public IP, allowing any internet host to communicate wit the LAN host by targeting this public IP.
- **Restricted-Cone NAT:** Similar to Full-cone NAT, but restricts communication from the internet to a LAN machine unless initiated by the LAN machine.
- **Port-restricted cone NAT:** A more restrictive EIM NAT than Restricted-cone, limiting external host communication by both IP and port.
- **Symmetric NAT:** An EDM NAT that maps requests from the same internal IP:Port to different public IP:Port pairs depending on the destination, complicating peer-to-peer connections.

Symmetric NAT, functioning like a strict firewall, restricts incoming packets to those from an IP pair that has already

sent an outgoing packet. Administrators may prefer Symmetric NAT for its ability to support up to 65535 users per IP while seemingly enhancing security by limiting exposure to selected hosts, a response to the Internet's lack of protection against unsolicited data, including spam and malware [10]. While it does not affect browsing, Symmetric NAT significantly hinders peer-to-peer protocols like BitTorrent. J.J.D. Mol et al. [11] noted that peers behind firewalls struggle with fair sharing, suggesting NAT puncturing or static IPs for better performance.

The method used to determine the NAT type of each cellular provider is based on RFC3489 [8] where the client (in this case, the mobile phone) sends a Binding Request —over UDP— to a STUN server to determine the bindings allocated by the NATs. The STUN server will respond with a message containing the IP address and port from which the request came. The client will then send more Binding Requests to ports and STUN servers. With the responses to these requests, the client can then determine the NAT type that they are behind by analysing how the responses of the STUN servers changed.

The type of NAT used by the cellular providers tested are shown in section IV-B. The information can be used to tailor the connectivity technique on the peer's carrier.

B. Determining NAT Timeouts

To get a clear idea of how the NAT mappings over UDP work, one can imagine the first outgoing packet as both a regular packet and a connection initiation message. When this first packet is sent, the NAT that the packet was sent from starts a timer as soon as the packet leaves. That timer waits for a response from the receiving client, meaning that the packet was received/accepted, and regular communication will follow. This timer will be referred to as `connection initiation timeout` throughout this section. Knowing this parameter is very useful for the case of a fully collaborative distributed network since the connection initiation timeout is the time that the peers have to collaborate and connect the new joiner based on the NAT mapping that the new joiner advertised.

The second type of timeout is called `session timeout`, meaning how long will the mapping remain active while there are no outgoing or incoming packet flows? Knowing how long the session can remain active while idle is used to determine how often "connection maintenance" packets need to be sent to keep the connection alive. Once a connection is established, it is preferred to be maintained since maintaining a connection is much "cheaper" than re-establishing one.

The two are considered separately because the connection initiation timeout is usually different, often smaller than the session timeout.

Starting with determining the connection initiation timeout, initially a lower and an upper bound on the time the mapping will remain active while waiting for a response is estimated. This is achieved by sending a packet to the server; the server waits a fixed amount of time before responding. After receiving each packet, the server's wait time increases by a fixed amount. If no response is received, indicating that the NAT mapping has expired, the time the server waited to send the response represents the upper bound of the timeout. Therefore, the wait time of the last successfully received packet is the lower bound.

When the bounds are established, a binary search is performed on those bounds to find the precise —down to the second— timeout of the NAT.

The process for determining the session timeout is very similar to the one for connection initiation timeout; Initially, a lower and upper bound on the idleness time of a connection is estimated. To achieve this, the client sends a packet to the server, and the server responds with the port number from which the client sent it. Then, the client waits a fixed amount of time until it sends the next packet. The wait time is incremented by a fixed amount for each subsequent transmission. The client compares the port in the server's response, i.e., the port from which the client's NAT sent the message. If the port in the latest response is not the same as in the previous one, then the mapping timed out, creating a new one. The time the client waited to send the last request becomes the upper bound, and the time they waited for the penultimate request becomes the lower bound of the session timeout.

When the bounds are determined, a binary search is run within those bounds to determine the expiration time down to the second precisely.

The results of multiple runs using different cellular providers can be seen in section IV-C.

C. Maximum Transmission Unit

The maximum transmission unit (MTU) denotes the maximum size of a single data unit that can be sent in a network layer transaction. MTU is related to the maximum frame size at the data link layer (such as an Ethernet frame).

A larger MTU is linked with reduced overhead, allowing more data to be transmitted in each packet. Conversely, smaller MTU values can help decrease network delay by facilitating quicker processing and transmission of smaller packets. Determining the appropriate MTU often hinges on the capabilities of the underlying network. It may require manual or automatic adjustment to ensure that outgoing packets don't exceed these capabilities.

A jumbo frame is an Ethernet frame with a payload greater than the standard maximum transmission unit (MTU) of 1,500 bytes.

Determining the MTU of a cellular provider requires running multiple ICMP pings [2] with binary-search-like varying payload sizes until the exact payload size is determined, where one more byte will cause the packet to be slit into two. Section IV-D presents the MTU of the different providers tested and whether they support Jumbo frames.

Algorithm 1 Simple Birthday Attack

Require: On packet received, send an ACK
Require: On packet received, $ackRcvd \leftarrow True$
Require: On packet received, store senders port
    $ackRcvd \leftarrow False$
    $msgsSent \leftarrow 0$
    $packet \leftarrow createPacket(UUID)$
    while $msgsSent < 243587$ and no $ackRcvd$ **do**
        $port \leftarrow getRandomPort()$
        $sendPacket(port, packet)$
    end while
    if $ackRcvd$ **then**
        return port
    else
        Birthday Attack was unsuccessful
    end if

D. Simple Birthday Attack

The rule for communicating in a NATed network is that the person behind the NAT must initiate communication first. The assumption is that the Internet works mainly in a Client-Server fashion where the Server is discoverable (has a *Public Static IP address*). This assumption breaks in the case of peer-to-peer communication between two clients behind a NAT since none are discoverable, and no one can initiate the communication.

A solution to this is, as explained in [4], [12], both peers should send packets to random ports until a "match" is achieved. A match is when peer A sends a packet from port X to port Y, and peer B sends a packet from port Y to port X in a timeframe smaller than their NAT's timeout. One can understand that the probability of this match is $\frac{1}{65535^2}$, which is almost impossible to achieve given restrictions imposed by the providers and NATs having limited memory, let alone it will take a lot of time.

This can be improved using a Birthday Attack, which is an attack built on the Birthday Paradox [13], a counterintuitive probability theory concept that states that in a group of just 23 people, there's a better than 50% chance that two people share the same birthday. This might seem surprising, as intuition might lead one to think that with 365 days in a year, it would require many more people to have such a high probability of a shared birthday. The paradox arises because we're not just looking for a specific birthday match but any pair of people with matching birthdays. The probability of two people not sharing a birthday decreases as more people are added to the group, and the likelihood of at least one pair sharing a birthday increases significantly.

The birthday paradox can be used to reduce the number of combinations of $sender\_port, receiver\_port$ while maintaining a satisfactory match probability. From the Birthday Paradox calculator [14], one can get a 50% success rate of a match after sending 77162 packets, and for a 99.9% success

[2]https://www.cloudflare.com/learning/ddos/glossary/internet-control-message-protocol-icmp/

rate, 243587 packets are needed. Due to the nature of NATs (timeouts and a limited number of mapping maintained), these probabilities are unlikely to occur. Still, it gives a good idea of how the problem becomes more accessible.

Using the numbers above, an Android application [7] was developed to attempt to connect two mobile peers using 4/5G (which is by default using a NAT) using algorithm 1.

The evaluation results of 10 runs per provider in the Netherlands and Cyprus are shown in table III, where the first row of each cell (on the left side of the arrow) shows the success rate of penetrations using a randomized birthday attack. The evaluation of the simple birthday attack did not show encouraging results. The first conclusion that can be derived is that whether the attack will lead to a connection is very dependent on the cellular provider pair.

These aside, a small part of the trials resulted in at least one successful connection, although many provider combinations never succeeded. Even once they manage to eventually succeed at connecting, it is not satisfactory since one successful attempt out of ten makes this technique costly in terms of cellular data used and time inefficient since coordinating two users to start attacking at the same time is already challenging and error-prone on its own, doing it multiple times to achieve a single connection will deem the application not very useful.

E. Improving the Birthday Attack

Given the complexity and the cost of cellular data and time, the success rate of the simple birthday attack, as shown in table III, is unsatisfactory; after multiple hypotheses on how to improve the connectivity based on time of connection, area, etc., the most prominent idea that the research team came up with is to understand the inner workings of each NAT, i.e., discover how the mapping works, whether there are any consistent patterns followed, etc. and then use that to predict what the following ports that each NAT will map the requests to. So instead of trying to connect to random ports of the peer, make a prediction —partially based on the NAT's inner workings, partially random, to enable exploration and exploitation— of what port the peer's NAT will map to and attempt that. The peer will do the same, thus *potentially* increasing the probability of connecting. This will still be attempted based on the Birthday Paradox 99.9% likelihood of success, i.e. maximum 243587 attempts.

To understand the inner workings of the NATs, it was first assumed that no provider uses the same NAT (in terms of configuration) since it was observed from different experiments, i.e. NAT timeouts, that even though there are some standards on how a NAT should be configured such as RFC 2663 and 4787 [1], [15] which are not necessarily followed; thus an Android mobile client and a Kotlin server were developed [16] to gather data on each provider that a SIM card could be easily acquired from by visiting the country, buying SIM cards and collecting data about their NAT's mapping strategy on their local network. The mobile client sends packets containing a UUID[3] to the server from random mobile ports to random

[3]https://docs.oracle.com/javase/8/docs/api/java/util/UUID.html

server ports. The UUID, a timestamp, and the source and destination ports are saved in a CSV file for each packet sent. The server, which lies behind an unrestricted network having a static IP address, does the same; once a packet is received, it stores the UUID (which is in the body of the packet), the port that the mobile sent it from (NAT mapping), and the port that the server received it from together with a timestamp on when the packet was received. The two CSVs are then inner-joined on the UUID column, resulting in two crucial columns: the port the mobile believes it sent the packet from and the port the packet came from, i.e. the exact NAT mapping.

To figure out the algorithm behind each mapping, i.e. what drives the decision-making on which port maps to which and when, a manual Exploratory Data Analysis (EDA) [17] is performed [18] to uncover the hidden inner workings of each NAT. The questions that the EDA aims to answer are:

1) Is the first port mapping completely random?
2) Is the mapping following a pattern?
3) Does the pattern, if it exists, depend on the port choices, sender or receiver? Is it time-based?
4) For how long is the pattern being followed, and if it changes at some point, why?

To answer these, different tests are performed while trying to make sense by visualizing the data or analysing time or population-based windows. The results of this EDA are explained in section IV-A.

Using the findings of the EDA, a new connectivity library was developed. The implementation is very similar to algorithm 1 used for the simple birthday attack, with two significant differences. First, instead of choosing a random port to send to, it selects the port based on the peer's port-choosing algorithm shown in table I. This means that to have a higher probability of connecting to some peer, one needs to know the provider from which the peer is connected. The second difference is that the phone opens five hundred random ports and listens to them simultaneously, potentially increasing the probability of choosing an open port.

IV. EVALUATION

A. Inner workings of NATs

This chapter explores the internal mechanisms of cellular data NATs used by various European carriers, focusing on the five leading Dutch providers: KPN, Vodafone, LycaMobile, Lebara, and Odido (formerly T-Mobile Netherlands). Through reverse engineering, it reveals the distinct mapping algorithms, timeouts, and MTUs each provider employs. Understanding these behaviours enables the establishment of successful peer-to-peer connections on 5G networks without intermediary assistance. The methodologies for analyzing NAT mappings are documented and available on GitHub [18] and summarized in Table I.

The chapter begins with an analysis of the mapping algorithms of the primary Dutch providers, followed by a discussion of timeout settings and MTUs for all tested carriers. It then addresses issues encountered due to roaming, which led to reconsidering the project's original goal of measuring connection times across different providers. Finally, the chapter evaluates a new connectivity library, examining whether leveraging insights into NAT behaviours enhances connection success rates for Dutch and Cypriot carriers compared to the simple birthday attack strategy detailed in Section III-D.

1) Lebara Netherlands: Analysis of Lebara's NAT behaviour showed that sender ports often followed a sequential pattern, starting with a random port and incrementing to adjacent ones until a new random port was selected. These initial random ports were frequently reused within the same session, making them significant, though inconsistent across runs.

As shown in figure 3a, which shows the port mappings of an $\approx$ 2 hour run, port mappings often began around the 3625 region, cycled through sequential increments, moved to other random blocks, and returned to the original region at consistent intervals. During low traffic periods, initial ports were typically multiples of 256, followed by 255 consecutive ports, as illustrated in figure 3b. This is consistent with prior findings [5], with the added insight that Lebara's NAT behaviour is both usage-based and time-based. Users stay in a port block until it times out or all ports are used.

It appears that ports are organized in blocks of 256 and assigned to users based on availability. Assignment strategies may depend on active users or available ports, as indicated by different behaviours observed during low and high traffic periods. Clarifying these allocation strategies requires further study, as proposed in section VI-A.

After concluding the analysis, it was noticed that the Lebara SIM card used was deprovisioned without notice, and subsequent attempts to reconnect or reactivate it were unsuccessful, indicating a potential carrier response to the testing activities.

2) KPN: KPN, like Lebara, groups ports into blocks of 256 consecutive numbers. The main difference is that KPN, with more infrastructure as a primary network operator, allows more extensive consumption of these port blocks compared to Lebara, as shown in Figure 4. This difference likely results from fewer users per IP address on KPN.

During testing, the phone completely consumed a block of 256 ports 32.3% of the time. Additionally, 36.9% of the time, the phone was assigned to a group starting with a port number divisible by 256. This pattern suggests that targeting ports divisible by 256 potentially increases the likelihood of successful connections, as all mappings remain active concurrently if attempted without interruption.

3) LycaMobile Netherlands: LycaMobile, using the KPN network, adopts a random address mapping strategy. Analysis of approximately 288,000 mappings revealed port usage within the range of 2048 to 65535, resulting in around 51,000 unique mappings. Port reuse lacked a consistent pattern, with intervals varying from 85 to 5980 seconds, suggesting a First Come, First Serve allocation approach.
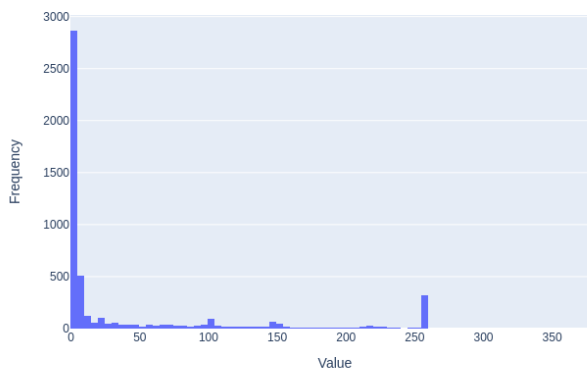
The observed port reuse after 85 seconds, shorter than the typical timeout, implies that ports may timeout earlier due to high demand or excess usage. LycaMobile appears to
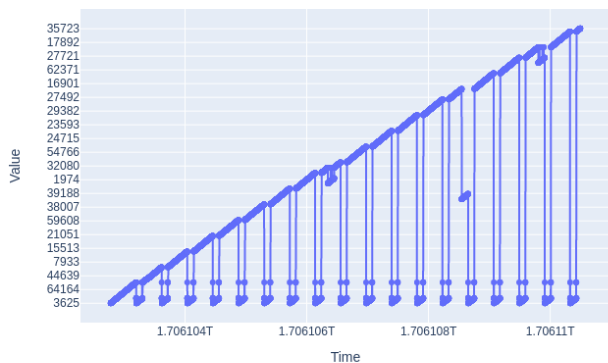
(a) 128 minutes Lebara of Lebara port mappings

(b) Frequency of consecutive port numbers used by Lebara

Fig. 3: Lebara analysis

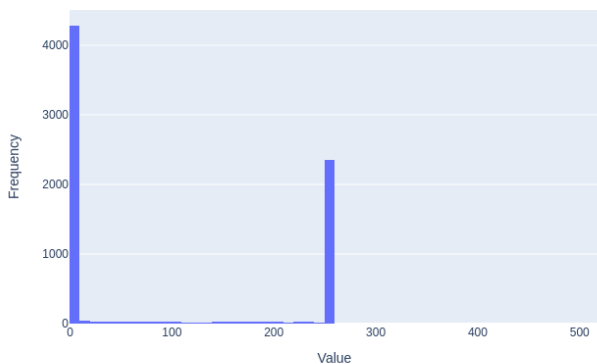

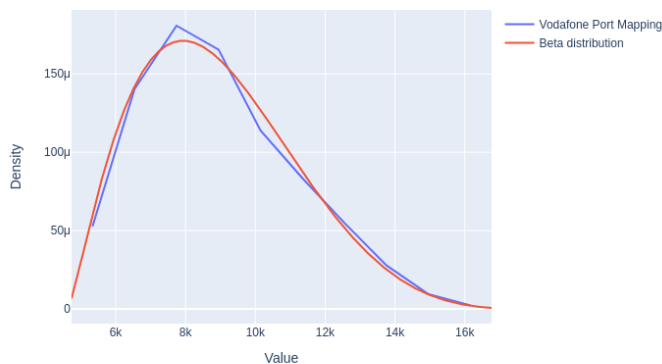Fig. 4: Frequency of consecutive port numbers used by KPN

Fig. 5: Beta Distribution fitting on Vodafone's mappings

use a shared port pool for the $[2048, 65535]$ range, with no evidence of systematic sorting, as consecutive port usage was infrequent.

4) Vodafone Netherlands: Over 900,000 mappings were analyzed for Vodafone, revealing that it does not follow the KPN and Lebara model of dividing ports into 256 blocks, nor does it use LycaMobile's random port assignment strategy. Initial observations suggested a normal distribution in the frequency of port mappings; however, a Shapiro-Wilk test [19] disproved this, indicating that a beta distribution provides a better fit, as shown in Figure 5.

The beta distribution, a continuous probability distribution defined on $[0, 1]$, is characterized by shape parameters α and β, influencing the distribution towards higher or lower values, respectively [20]. The location and scale parameters were also relevant, shifting and spreading the distribution along

the x-axis. The empirical beta distribution for Vodafone is:
$$Y \sim \mathrm{B}(\alpha, \beta, loc, scale) = \mathrm{B}(2.242, 5.008, 4630, 13937)$$

5) Odido: Odido's NAT randomly selects ports, with analysis showing that only 9,920 unique ports were used out of 493,982 mappings scattered across the port space rather than using consecutive ports like Vodafone. Figure 6 illustrates the distribution of Odido's mappings.

To optimize NAT penetration, a proposed algorithm merges consecutive bins with frequencies above 30 to avoid over-specifying, accommodating time-dependent results. An exploitation-exploration strategy is then applied, with 40% of port selections based on established frequency ranges and 60% from random ports within $[2048, 65535]$[4]. Selecting bins based

---

[4]Details of the bins and algorithm implementation are available in section IV-F
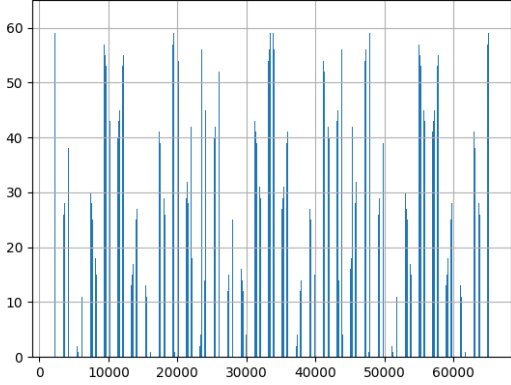
Fig. 6: Thousand bins histogram of the output of Odido's mapping strategy

on frequency showed inferior performance due to increased delays and potential overfitting, leading to random bin selection.

The Nat mapping algorithms of all carriers tested are shown in table I. The results of this study align with the findings of Z. Wang et al. [5] while also showing that the NAT algorithms evolved in many cases since the publication of that study.

B. Nat Types

Knowing the NAT type of the carrier one is using, and the one of the peer they want to connect to allows one to adapt their connectivity strategy to increase the chance of connecting. Different strategies should be adopted based on the types, i.e. a Symmetric NAT requires a Birthday Attack to connect. In contrast, one can easily connect with a peer behind a Full-Cone NAT using a STUN server or some peer acting as a middleman relaying information to the rest of the network. The types of the NATs of various carriers are presented in the sixth column of table I.

C. Timeout of NATs

Analyzing the NAT timeouts showed two crucial things. First, roaming highly influences the timeout, as seen from the results of the Norwegian cellular carriers and Belgium's LycaMobile. Second, the timeouts are multiples of a minute, a behaviour that is to be expected. Still, it also shows that designers did not optimize the timeouts to the full extent since it is highly unlikely that the optimal timeout is nicely rounded to the minute.

Table II can be interpreted as LB being a lower bound and UB being an upper bound. Bounds are used since, due to network delays, it is hard to know which of the two is the actual timeout. On the left side is the timeout for how long the mapping will remain active when the recipient does not receive an initial response. On the right side is the communication timeout, i.e., communication is established (initial response is received) but is currently idle. The timeouts of each cellular provider don't differ significantly except for the two extremes, which are Lyca of Belgium being unable to establish a connection while roaming and CytaMobile of Cyprus having an enormous session timeout of half an hour.

D. Maximum Transmission Unit

Understanding a carrier network's Maximum Transmission Unit (MTU) provides several benefits. It allows for optimizing network performance by identifying the largest packet size that can be transmitted without fragmentation, thereby minimizing overhead and latency. Additionally, knowing the MTU facilitates efficient bandwidth utilization, as smaller packets can increase overhead and reduce throughput.

Jumbo frames improve network efficiency by accommodating larger packet sizes than the standard MTU, reducing transmission overhead. However, compatibility with all devices and networks must be ensured to fully utilise jumbo frames.

The MTU values for various carriers and their support for jumbo frames are presented in the last two columns of Table I. Overall, there is no consistent MTU standard across cellular providers, especially regarding jumbo frame support, with variations near the 1500-byte threshold for those not supporting jumbo frames.

E. Roaming

Roaming is significantly hindering the success of a connection attempt between two peers. No definite reason is derived on why roaming hinders connectivity since many parameters of the NATs change when roaming. Some observations of behavioural and parameter changes are:

1) **Telia and MyCall Norway:** Telia and MyCall of Norway both had a 120-second timeout for connection initiation —measured in the Oslo airport. When measured in Delft, Netherlands (both tunnelling through LycaMobile), the timeout fell to 5 seconds and 19 seconds, respectively, showing that roaming changes the timeout configuration.

2) **LycaMobile Belgium:** LycaMobile Belgium's connection initiation timeout could not be measured in Delft, Netherlands, as no response reached the phone after multiple attempts. This suggests that the timeout is extremely short, making it highly susceptible to even minimal network delays. Such a brief timeout would likely result in a poor user experience, indicating this may not be standard behaviour. The findings suggest that they also alter their timeout settings when roaming.

3) **Vodaphone Netherlands:** A birthday attack conducted between two phones on the Vodafone NL network resulted in a 40% success rate (success defined as at least one packet received out of 243587 sent). In contrast, when re-measured from Cyprus (with both phones roaming via CytaMobile-Vodafone), using identical conditions and during low traffic hours, there were no successful connections over approximately 4 hours of attempting. This significant decrease in success rate indicates that roaming introduces underlying differences despite the appearance of similar network conditions.

Country	Name	Algorithm	Infrastructure Owner	ID Required	NAT Type	MTU	Allows Jumbo Frames
NL	KPN	Let B_i represent a block of 256 port numbers $[B_i = \{256 \times i, 256 \times i + 1, \ldots, 256 \times i + 255\}]\forall i \in [0, 255]$ The user is assigned to a block B_i, Users consume ports with numbers incrementing linearly When B_i has no available ports, user is assigned to B_j, etc.	✓	✓	Symmetric	1445	×
NL	Lebara	Let B_i represent a block of 256 port numbers $[B_i = \{256 \times i, 256 \times i + 1, \ldots, 256 \times i + 255\}]\forall i \in [0, 255]$ The user is assigned to a block B_i, Users consume ports with numbers incrementing linearly When B_i has no available ports, user is assigned to B_j, etc.	KPN	×	Restrict	65507	✓
NL	Lyca	Random Sampling from the block $[2048, 65535]$	KPN	×	Full Cone	1473	×
NL	Vodafone	Beta Distribution: $Y \sim \mathrm{B}(\alpha, \beta, loc, scale) = \mathrm{B}(2.242, 5.008, 4630, 13937)$	✓	×	Restrict	1437	×
NL	Odido	Semi-Random Sampling from the block $[2048, 65535]$ Sampling strategy is analysed in \ref{sec:odido}	✓	×	Symmetric	3972	✓
FR	Orange	Random Sampling from the block $[1, 65500]$	✓	✓			
FR	SFR	Random Sampling from the block $[1025, 65535]$	✓	✓			
BG	Orange	Let B_i represent a block of 256 port numbers $[B_i = \{256 \times i, 256 \times i + 1, \ldots, 256 \times i + 255\}]\forall i \in [0, 255]$ The user is assigned to a block B_i, Users consume ports in the block randomly User will stay assigned to B_i for an extended period and reuse the same ports	✓	✓	Symmetric	1472	×
BG	LycaMobile	Random Sampling from $[2048, 65535]$	TeleNet	✓	Restrict	42987	✓
NO	Telia	Let B_i represent a block of 256 port numbers $[B_i = \{256 \times i, 256 \times i + 1, \ldots, 256 \times i + 255\}]\forall i \in [0, 255]$ The user is assigned to a block B_i, Users consume ports in the block randomly User will stay assigned to B_i for an extended period and reuse the same ports	✓	✓	Restrict	65507	✓
NO	MyCall	Let B_i represent a block of 256 port numbers $[B_i = \{256 \times i, 256 \times i + 1, \ldots, 256 \times i + 255\}]\forall i \in [0, 255]$ The user is assigned to a block B_i, Users consume ports in the block randomly User will stay assigned to B_i for an extended period and reuse the same ports	Telia	✓	Full Cone	65507	✓
CY	Epic	Let B_i represent a block of 256 port numbers $[B_i = \{256 \times i, 256 \times i + 1, \ldots, 256 \times i + 255\}]\forall i \in [0, 255]$ The user is assigned to a block B_i, Users consume ports with numbers incrementing linearly When B_i has no available ports, user is assigned to B_j, etc.	✓	×	Symmetric	4433	✓
CY	Cyta	Random Sampling from the block $[5000, 65535]$	✓	×	Restrict	1472	×
CY	Primetel	Random Sampling from the block $[1024, 15500]$	Cyta	×	Symmetric	4433	✓
CY	Cablenet	Let B_i represent a block of 256 port numbers $[B_i = \{256 \times i, 256 \times i + 1, \ldots, 256 \times i + 255\}]\forall i \in [0, 255]$ The user is assigned to a block B_i, Users consume ports in the block randomly User will stay assigned to B_i for an extended period and reuse the same ports	✓	×	Restrict	65507	✓
IT	TIM	Let B_i represent a block of 256 port numbers $[B_i = \{256 \times i, 256 \times i + 1, \ldots, 256 \times i + 255\}]\forall i \in [0, 255]$ The user is assigned to a block B_i, Users consume ports in the block randomly User will stay assigned to B_i for an extended period and reuse the same ports	✓	✓	Full Cone	65507	✓
IT	WindTre	Let B_i represent a block of 256 port numbers $[B_i = \{256 \times i, 256 \times i + 1, \ldots, 256 \times i + 255\}]\forall i \in [0, 255]$ The user is assigned to a block B_i, Users consume ports with numbers incrementing linearly When B_i has no available ports, user is assigned to B_j, etc.	✓	✓	Symmetric	8873	✓

TABLE I: The algorithm each carrier's NAT uses, their configurations, and the ease of obtaining a SIM card.

Provider	Connection Initiation Timeout					Session Timeout				
	LB(s)	UB(s)	Server Port	Location	Tunnel	LB(s)	UB(s)	Server Port	Location	Tunnel
Lebara NL	120	121	2000	Echo TuD	-	240	241	2000	Echo TuD	-
Lyca NL	120	121	2000	Echo TuD	-	120	121	2000	Echo TuD	-
Odido NL	120	121	2000	Echo TuD	-	119	120	2000	Echo TuD	-
Vodaphone NL	302	303	2000	Echo TuD	-	299	300	2000	Echo TuD	-
KPN NL	120	121	2000	Echo TuD	-	239	240	2000	Echo TuD	-
Orange BG	58	59	2000	Echo TuD	Odido	60	61	2000	Echo TuD	Odido
Orange BG	59	60	2000	Echo TuD	Lyca NL	57	58	2000	Echo TuD	Lyca NL
Lyca BG	F	F	2000	Echo TuD	Lyca NL	F	F	2000	Echo TuD	Lyca NL
Telia NO	120	121	2000	Oslo Airport	-	-	-	-	-	-
Telia NO	5	6	2000	Echo TuD	Lyca NL	300	301	2000	Echo TuD	Lyca NL
MyCall NO	120	121	2001	Oslo Airport	-	-	-	-	-	-
MyCall NO	19	20	2001	Echo TuD	Lyca NL	299	300	2001	Echo TuD	Lyca NL
CytaMobile CY	59	60	2000	Aglantzia, CY	-	1800	1801	2000	Aglantzia, CY	-
Epic CY	239	240	2001	Aglantzia, CY	-	240	241	2001	Aglantzia, CY	-
PrimeTel CY	200	201	2000	Aglantzia, CY	-	240	241	2000	Aglantzia, CY	-
CableNet CY	20	21	2001	Aglantzia, CY	-	30	31	2001	Aglantzia, CY	-
TIM IT	299	300	2000	BellaMonte, IT	-	420	421	2000	BellaMonte, IT	-
WindTre IT	30	31	2001	BellaMonte, IT	-	30	31	2001	BellaMonte, IT	-

TABLE II: Timeouts of various carriers in seconds. On the left are timeouts for waiting for communication to be established, and on the right is a timeout of the idle communication channel.

Notably, the timeout and NAT types remained consistent during roaming tests.

The initial goal of creating a matrix to show connection times across European cellular providers was aborted due to a near-zero success rate for birthday attacks when roaming. Despite using validated software and various roaming configurations, no successful connections were made when at least one phone was roaming.

Experiments conducted over a week showed that roaming features significantly hinder connectivity. Although it remains unclear if cross-provider connections are impossible, they are unachievable within a reasonable timeframe using the current approach. Thus, the creation of the connection time matrix is postponed.

1) Roaming makes birthday attack-based connectivity almost impossible: This section compares the connection time required between two roaming peers versus two peers on the local Telia network in Norway. For the roaming scenario, we assume a best-case situation where the roaming user is the sole user of the roaming tower, with both peers in Cyprus using a Samsung Galaxy A53 5G (SM-A536B/DS).

Three key variables are considered:

- $p \approx 2.98$ **ms**: processing time
- $l \approx 79.20$ **ms**: average latency from Limassol to Oslo [21]
- $P = 64511$: available ports

In this scenario, each phone's NAT (NAT A and NAT B) creates mappings, X_A and X_B, with a 5-second connection initiation timeout. The algorithm aims to cause a collision while these mappings are active by sending packets to random ports. The probability of a collision is approximately 0.00000000024. The window of puncturing opportunity is the time-to-live of a port that has yet to receive a response to its connection requests; in this case, the time-to-live is 5 seconds minus latency. Each phone can send a packet every p milliseconds. Hence, it can attempt about 1678 ports per

window, resulting in a collision probability of 0.0000004032 per window. In the worst case, about 2480159 windows are needed. Since a new window is created every l milliseconds, it equates to approximately 54.6 hours.

For the local Telia Norway network, the parameters change to:

- $p \approx 2.98$ **ms**: processing time
- $l \approx 23$ **ms**: average local latency [22]
- $P = 64511$: available ports

With a time-to-live of around 120 seconds, each phone can attempt about 40269 ports per window, resulting in a collision probability of 0.00000967616. Thus, approximately 103347 windows are needed, equating to about 39.6 minutes.

These calculations demonstrate that establishing a connection while roaming takes at least 82 times longer than on the local network.

F. Cellular Provider Aware NAT Puncturing

NAT mapping data from various cellular providers were implemented in Kotlin in a connectivity library [23]. This library can connect to multiple peers simultaneously by sending `CONNECTION-INIT` packets from various open ports and awaiting responses. Connectivity attempts are made asynchronously, allowing the main app to remain responsive during connection establishment attempts. The peer's ID, IP address, and port are returned asynchronously upon successful connection. The library also sends `CONNECTION-MAINTENANCE` packets periodically to keep connections active based on the peer's timeout settings. A demo app[5] was developed to evaluate the library's success rate against a simple random birthday attack.

Evaluation results, shown in Table III, indicate that provider-aware NAT puncturing led to successful connections in four additional combinations compared to the random method in

---

[5]https://github.com/OrestisKan/bday-demo-app

the Netherlands: Odido-Odido, Odido-Lebara, Lebara-Lebara, and VodaFone-KPN. For Cyprus ..... Cyta-Cyta combination demonstrated the highest connectivity rate (9 out of 10 attempts were successful). Notably, Odido-Vodafone showed no successful connections, and VodaFone-VodaFone had one less success than the random attack. Despite these mixed results, hypothesis testing was conducted for each provider combination to assess the statistical significance of the observed improvements.

Hypothesis testing was conducted using Fisher's exact test due to the limited sample size of connectivity attempts [24]. The null hypothesis (H_0) holds that the proportion of successful NAT punctures using cellular-aware NAT puncturing is not greater than with the random birthday attack: $H_0 = P_{random} \leq P_{CellularProviderAware}$. The alternative hypothesis (H_1) suggests that cellular-aware NAT puncturing has a higher success rate: $H_1 = P_{random} < P_{CellularProviderAware}$. A significance level of 0.05 was used. If the p-value from Fisher's test is below this threshold, H_0 is rejected. Table III presents the p-value results, with green-highlighted cells indicating where H_1 is accepted, demonstrating that cellular-aware NAT puncturing performed statistically better than the random birthday attack method.

The results of cellular provider-aware NAT puncturing, although a few times showed that it improved the connectivity rate statistically significantly, did not make it worse. Since the inverse, Fisher's exact test (H_1 being that random birthday attack has a higher proportion of successful results) would not be accepted in any provider combination of the ones tested.

V. Acknowledgements

The 5G puncturing knowledge presented here shows the feasibility of a generic provider-aware NAT solution in the future. Our ideas are inspired by the IPv8 networking technology pioneered by Dr. Stokkink[6]. The IPv8 networking stack uses UDP puncturing of LAN devices as the exclusive mechanism for network discovery. This 2018 work inspired us to attempt 5G puncturing. The basic open source skeleton of DeToks was provided by user *InvictusTMC* on Github.

VI. Discussion and Future Work

This exploratory study on the inner behaviour of NATs showed various exciting properties. First of all, no NAT implementation is the same. Birthday attacks are inherently unpredictable; even with complete knowledge of a carrier's NAT mapping function, randomness and outside influences affect the attack's success. For example, attempting a birthday attack during peak network usage hours will probably lead to a lower success rate than during peak hours. This is due to the carriers experiencing congestion on their network and employing some fairness protocols to allow all users to be connected, thus limiting a user that requires high network usage (one that performs a birthday attack, which constantly opens up sockets in a "robotic" way).

Another limiting factor is roaming. As explained in section IV-E, roaming significantly hindered this research since the connectivity through birthday attacks while roaming is almost impossible for reasons that have not yet been fully identified.

The main takeaway from this research is that connectivity through birthday attacks in a fully distributed setting is possible in principle. Still, it is hard to quantify the success rate and its reliability accurately. So many factors may jeopardize it, such as NAT type of carrier, combination of carriers, time of the day, congestion of the network and roaming.

A. Future Work

This study marks a significant step towards understanding and enhancing connectivity in distributed artificial intelligence (AI) deployments over 5G networks. However, several areas for further research and development can build on the findings presented.

1) Optimization of NAT Puncturing Techniques: Although the provider-aware NAT puncturing strategy showed promising results, there is room for optimization. Future work could explore more advanced algorithms that adaptively adjust puncturing strategies based on real-time network conditions or leverage machine learning to predict the most effective puncturing patterns based on historical data. Additionally, integrating these techniques into existing networking protocols could streamline implementation and improve scalability.

2) Addressing Roaming and Mobility Challenges: The research highlighted significant challenges in maintaining connectivity during roaming, which remains an open problem. Future research could focus on developing robust mechanisms to overcome roaming's limitations.

3) Development of Decentralized AI Applications: The proof-of-concept decentralized social media application demonstrated the feasibility of the proposed architecture. Future work could expand this by developing and testing more complex decentralized AI applications across different domains, such as search engines or any area where there is a monopoly or oligopoly by some companies.

4) Security and Privacy Considerations: As decentralized AI systems gain traction, security and privacy concerns become increasingly important. Future research should explore the security implications of decentralized networks, focusing on potential vulnerabilities introduced by NAT puncturing and strategies to mitigate these risks.

5) Evaluation of Long-Term Performance and Scalability:: The study provided an initial assessment of the system's performance, but further long-term evaluations are necessary. Future research could investigate the system's scalability, especially under heavy network traffic or during peak usage times, and assess its robustness in real-world deployment scenarios. This could involve stress-testing the system in large-scale simulations or live environments to understand its limitations and potential failure points.

By addressing these areas, future work can continue to enhance the viability and resilience of distributed AI systems

$H_0 = P_{random} \leq P_{CellularProviderAware}$ $H_1 = P_{random} < P_{CellularProviderAware}$ $\alpha = 5\%$					
NETHERLANDS	**Odido**	**Lebara**	**LycaMobile**	**VodaFone**	**KPN**
Odido	Penetrations: $\frac{0}{10} \Rightarrow \frac{4}{10}$ $p = 0.0433$ H_1 is accepted	-	-	-	-
Lebara	Penetrations: $\frac{0}{10} \Rightarrow \frac{6}{10}$ $p = 0.0054$ H_1 is accepted	Penetrations: $\frac{1}{10} \Rightarrow \frac{1}{10}$ $p = 0.7632$ H_1 is rejected	-	-	-
LycaMobile	Penetrations: $\frac{1}{10} \Rightarrow \frac{1}{10}$ $p = 0.7632$ H_1 is rejected	Penetrations: $\frac{0}{10} \Rightarrow \frac{0}{10}$ $p = 1.0$ H_1 is rejected	Penetrations: $\frac{0}{10} \Rightarrow \frac{0}{10}$ $p = 1.0$ H_1 is rejected	-	-
VodaFone	Penetrations: $\frac{2}{10} \Rightarrow \frac{0}{10}$ $p = 1.0$ H_1 is rejected	Penetrations: $\frac{1}{10} \Rightarrow \frac{6}{10}$ $p = 0.0286$ H_1 is accepted	Penetrations: $\frac{1}{10} \Rightarrow \frac{2}{10}$ $p = 0.5$ H_1 is rejected	Penetrations: $\frac{4}{10} \Rightarrow \frac{3}{10}$ $p = 0.8251$ H_1 is rejected	-
KPN	Penetrations: $\frac{0}{10} \Rightarrow \frac{0}{10}$ $p = 1.0$ H_1 is rejected	Penetrations: $\frac{0}{10} \Rightarrow \frac{0}{10}$ $p = 1.0$ H_1 is rejected	Penetrations: $\frac{0}{10} \Rightarrow \frac{0}{10}$ $p = 1.0$ H_1 is rejected	Penetrations: $\frac{0}{10} \Rightarrow \frac{4}{10}$ p=0.0433 H_1 is accepted	Penetrations: $\frac{0}{10} \Rightarrow \frac{0}{10}$ $p = 1.0$ H_1 is rejected
CYPRUS	**Cyta**	**Epic**	**PrimeTel**	**CableNet**	**-**
Cyta	Penetrations: $\frac{1}{10} \Rightarrow \frac{9}{10}$ $p = 0.0005$ H_1 is accepted	-	-	-	-
Epic	Penetrations: $\frac{0}{10} \Rightarrow \frac{2}{10}$ $p = 0.2619$ H_1 is rejected	Penetrations: $\frac{0}{10} \Rightarrow \frac{0}{10}$ $p = 1.0$ H_1 is rejected	-	-	-
PrimeTel				-	-
CableNet	Penetrations: $\frac{0}{10} \Rightarrow \frac{3}{10}$ $p = 0.1053$ H_1 is rejected		Penetrations: $\frac{0}{10} \Rightarrow \frac{0}{10}$ $p = 1.0$ H_1 is rejected		-

TABLE III: Results of the hypothesis testing on Random Birthday Attack versus Cellular Provider Aware NAT Puncturing

on 5G networks, paving the way for broader adoption and innovation in this field.

VII. CONCLUSION

This thesis investigated the complex NAT configurations employed by various European cellular providers to support decentralized social media applications over 5G networks. Through reverse engineering, it was revealed that each provider utilizes distinct strategies for port management, ranging from random allocation to the use of structured port blocks. This configuration diversity poses challenges for true peer-to-peer connectivity, which requires navigating these varied NAT behaviours without relying on centralized servers.

The introduction of a provider-aware NAT puncturing strategy marked a significant advancement over traditional random methods. Empirical results demonstrated improved connection success rates, and hypothesis testing via Fisher's exact test confirmed statistically significant enhancements in specific cases, as highlighted in the results table. These outcomes affirm that tailoring NAT penetration techniques to the particular configurations of different carriers can substantially enhance connectivity performance in distributed 5G applications.

These findings highlighted the critical role of understanding and adapting to NAT behaviour in achieving robust and reliable decentralized communication. The feasibility of true decentralized mobile applications is demonstrated by integrating the decentralized connectivity library with DeToks, the Tribler lab's decentralized TikTok.

APPENDIX

A. Orange Belgium

Analysis of 264479 mappings from North Brussels revealed that Orange Belgium uses only 767 unique ports across three blocks: $[5632, 5887]$, $[9984, 10239]$, and $[25088, 25343]$, with no linear selection or frequent pairs. Ports are grouped into 256-port blocks, assigned based on availability, and used non-linearly. An exploration-exploitation strategy, similar to Dutch Lebara and KPN, is recommended for NAT penetration.

B. LycaMobile Belgium

Analysis of 316151 mappings from North Brussels shows that LycaMobile Belgium, like its Dutch counterpart, exhibits no linear port increment, specific regional preferences, or consistent port reuse, utilizing almost the entire port space. Thus, LycaMobile's NAT likely maps each port to a randomly available one.

C. Orange France

Orange France uses a wide range of ports, from 1 to 65500, with 65407 unique ports mapped to and minimal reuse, suggesting random NAT mapping. Unlike other providers, Orange France mapped to reserved ports (1-1023).

D. SFR France

SFR France shows minimal port reuse and no linear increments, covering ports from 1025 to 65535, with 64509 unique ports used. The dataset of 257,913 entries indicates random NAT mapping.

E. Telia & MyCall Norway

Telia used 256 unique ports, while MyCall used 812, with Telia selecting ports from $[14592, 14847]$ and MyCall from $[7048, 7419]$ and $[42188, 42623]$. Ports are grouped into 256-port blocks and used non-linearly. An exploration-exploitation strategy similar to that used for Dutch Lebara, KPN, and Belgian Orange is recommended.

F. Cyta Cyprus

Analysis of 824220 mappings from Aglantzia, Nicosia, showed no linear port increment, regional preference, or consistent port reuse. This indicates that Cyta's NAT randomly maps each port to an available one across the entire port space.

G. Epic Cyprus

Analysis of 681,274 mappings from Aglantzia, Nicosia, indicates that Epic divides ports into 256-port blocks, with users assigned based on availability. Ports are allocated sequentially until the block is exhausted, then the user is assigned to a new block. Epic's NAT penetration strategy is similar to Dutch Lebara, KPN, Belgium's Orange, and Norway's Telia and MyCall.

H. TIM Italy

todo Buckets not linearly inc Big amount of users per bucket and high reusability

I. WindTre Italy

Buckets linearly inc Low amount of users per bucket, low reusability

J. PrimeTel Cyprus

655849 records

[1024,15462] 12090 unique biggest reused mapping was 2 but happened a thousand times -¿ probably luck no particular trend

Not sure if its random but a lot of numbers get mapped close to the actual port but also everything under 16k

K. Cablenet Cyprus

Number of mappings analyzed 608419 405 ports mapped to 4 regions used through out $\approx 90 hours$ same as mycall and telia!

REFERENCES

[1] C. F. Jennings and F. Audet, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP," RFC 4787, Jan. 2007. [Online]. Available: https://www.rfc-editor.org/info/rfc4787

[2] "Lane v. facebook, inc." p. 811, 2012.

[3] "Apple inc. v. pepper," p. 1514, 2019.

[4] D. Anderson, "How nat traversal works - nat notes for nerds," Apr 2022. [Online]. Available: https://blog.apnic.net/2022/04/26/how-nat-traversal-works-nat-notes-for-nerds/

[5] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang, "An untold story of middleboxes in cellular networks," *Proceedings of the ACM SIGCOMM 2011 conference*, Aug 2011.

[6] A. Nikravesh, D. R. Choffnes, E. Katz-Bassett, Z. M. Mao, and M. Welsh, "Mobile network performance from user devices: A longitudinal, multidimensional analysis," in *Passive and Active Measurement: 15th International Conference, PAM 2014, Los Angeles, CA, USA, March 10-11, 2014, Proceedings 15*. Springer, 2014, pp. 12–22.

[7] O. Kanaris, "NAT measurements gathering with Naive Birthday Attack for connecting smartphones," Dec. 2023.

[8] J. Rosenberg, C. Huitema, R. Mahy, and J. Weinberger, "STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)," RFC 3489, Mar. 2003. [Online]. Available: https://www.rfc-editor.org/info/rfc3489

[9] V. Paulsamy and S. Chatterjee, "Network convergence and the nat/firewall problems," 2003.

[10] M. Zolotykh, "Comprehensive classification of internet background noise," 2020.

[11] J. Mol, J. Pouwelse, D. Epema, and H. Sips, "Free-riding, fairness, and firewalls in p2p file-sharing," 2008.

[12] O. Kanaris and J. Pouwelse, "Mass adoption of nats: Survey and experiments on carrier-grade nats," 2023.

[13] K. Suzuki, D. Tonien, K. Kurosawa, and K. Toyota, "Birthday paradox for multi-collisions," in *Information Security and Cryptology – ICISC 2006*, M. S. Rhee and B. Lee, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 29–40.

[14] Fast-Reflexes, "Fast-reflexes/birthdayproblem-python: Implementation of a solver of the generalized birthday problem in python." [Online]. Available: https://github.com/fast-reflexes/BirthdayProblem-Python

[15] M. Holdrege and P. Srisuresh, "IP Network Address Translator (NAT) Terminology and Considerations," RFC 2663, Aug. 1999. [Online]. Available: https://www.rfc-editor.org/info/rfc2663

[16] O. Kanaris, "NAT Mapping data Gathering and analysing tool," Feb. 2023.

[17] Oct 2021. [Online]. Available: https://www.ibm.com/topics/exploratory-data-analysis

[18] O. Kanaris, "Cellular Network NAT Reverse Engineering and Exploration," Apr. 2024. [Online]. Available: https://https://github.com/OrestisKan/telecom-analysis

[19] [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.shapiro.html

[20] J. B. McDonald and Y. J. Xu, "A generalization of the beta distribution with applications," *Journal of Econometrics*, vol. 66, no. 1, pp. 133–152, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0304407694016124

[21] Jun 2024. [Online]. Available: https://wondernetwork.com/pings/Limassol

[22] S. Ltd., "Telia norge as speed test," Jun 2024. [Online]. Available: https://www.broadbandspeedchecker.co.uk/isp-directory/Norway/telia-norge-as.html

[23] O. Kanaris, "Birthday-Attack-based smartphone connectivity kotlin library," May 2023.

[24] A. Edwards, "Chapter 67 - r.a. fischer, statistical methods for research workers, first edition (1925)," in *Landmark Writings in Western Mathematics 1640-1940*, I. Grattan-Guinness, R. Cooke, L. Corry, P. Crépel, and N. Guicciardini, Eds. Amsterdam: Elsevier Science, 2005, pp. 856–870. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444508713501480