# Rapport LINFO1341 : Analyse d'une application de vidéoconférence : Discord

Groupe Discord1

1st Accou Pierre
*NOMA : 01291800*
pierre.accou@student.uclouvain.be

1st Botton Florentin
*NOMA : 30081900*
florentin.botton@student.uclouvain.be

## I. Introduction

Nos recherches ont pour but d'analyser le fonctionnement de l'application de visioconférence Discord d'un point de vue réseaux informatiques. Cette application a la capacité d'envoyer des messages mais également de passer des appels audio et vidéo avec partage ou non d'écran. Discord sera donc décomposée sous différentes couches qui composent les couches réseaux. Nous aborderons donc les protocoles et/ou termes suivants: DNS, IPv4, IPv6, protocoles de transport, TLS. Nos analyses ont été essentiellement effectuées en wifi.

Dans ce cadre, nous avons établi des scénarios pour comprendre et visualiser l'utilisation des protocoles. Grâce au logiciel Wireshark [1], nous avons réalisé des captures avec ou sans appel audio, avec ou sans caméra (flux vidéo), avec ou sans partage d'écran[1]. Les graphiques et une partie de l'analyse ont quant eux été réalisés à l'aide de la librairie Python Pyshark ainsi que certaines commandes telles que dig [2] ou whois [3] présentes dans un terminal. Cette analyse a été réalisée dans le cadre du cours LINFO1341 : Computer Network donné à l'UCLouvain par le professeur Olivier Bonaventure. Voici un lien vers notre dépot GitHub $https://github.com/Cailloux2123/ResO_Discord$.

## II. Domain Name System

DNS signifie Domain Name System. C'est un élément clé de l'infrastructure Internet car il permet aux utilisateurs d'accéder facilement aux sites web et autres services en utilisant des noms de domaines conviviaux, grâce à une hiérarchie de serveurs qui traduisent des noms de domaine en adresses IP, celles-ci étant des adresses numériques complexes peu mémorisables. Lorsqu'un utilisateur entre un nom de domaine dans un navigateur web, il envoie une requête DNS au serveur DNS le plus proche. Si celui-ci ne possède pas l'information demandée, la requête est transmise à un autre serveur DNS pour obtenir l'information recherchée. L'utilisateur peut ainsi accéder au site web correspondant en utilisant l'adresse IP fournie.

[1]Scénario 1 : envoi de messages, Scénario 2 : appel sans caméra, Scénario 3 : appel avec caméra, Scénario 4 : appel sans caméra avec partage d'écran, Scénario 5 : appel avec caméra et avec partage d'écran

### A. Les différents domaines contactés

Dans le cas d'un envoi simple de messages avec Discord, un seul serveur est contacté. Il s'agit de "discord.com". Nous observons 2 types de requêtes. Une "query" et une "query answer". Lorsque chaque serveur est contacté, 2 requêtes DNS sont lancées. Il s'agit d'une requête A demandant les adresses IPv4 du serveur et une requête AAAA qui demande les adresses IPv6. Parfois, nous observons que la réponse de la requête nous donne le serveur autoritatif. Pour le serveur "discord.com", son serveur autoritatif est "gabe.ns.cloudflare.com". Après une analyse plus approfondie à l'aide de dig (et plus particulièrement sa commande "dig NS le_nom_du_serveur"), nous observons également un autre serveur autoritatif "sima.ns.cloudflare.com".

Dans le cas d'un appel sans caméra, c'est un autre serveur, "rotterdam2796.discord.media", qui est contacté. À l'aide des précédentes commandes dig, nous observons que les serveurs autoritatifs sont "gabe.ns.cloudflare.com" et "dns.cloudflare.com"

Lors d'un appel avec la caméra, un autre serveur est contacté : "rotterdam7058.discord.media".

Par contre, lors d'un appel sans caméra avec un partage d'écran, nous contactons 2 serveurs. D'une part, un serveur "rotterdam5742.discord.media" au moment où un appel est donné, et, d'autre part, un autre serveur "rotterdam2421.discord.media" est contacté au lancement du partage d'écran. Nous en déduisons que le partage d'écran est assimilé à un appel dans l'appel déja en cours et pour ne pas surcharger le serveur gérant déjà l'appel, le partage d'écran est géré par un autre serveur. Il est utile de préciser que les serveurs cités précédemment ne sont pas uniquement ceux-ci. Lors d'un autre appel, cela peut être un autre serveur qui est contacté.

Tous ces serveurs sont gérés par Cloudflare, a Rotterdam, aux Pays-Bas. Il semblerait que Discord n'ait aucun serveur qui appartient à l'entreprise.

### B. Les différentes types de requêtes

Dans la majorité des situations, sauf lorsque la requête DNS ne reçoit pas de réponse, c'est IPv4 qui est utilisé par le réseau. Dans le cas où il n'y a pas de réponse à une query, IPv6 est utilisé. Dès qu'une réponse est reçue en IPv6, nous rebasculons sur IPv4.

En effet, lors d'un lancement d'appel, chez un de nous deux, une requête DNS A (en IPv4) n'a jamais obtenu de réponse à sa query. Dans ce cas, une nouvelle requête DNS AAAA (en IPv6) a été effectuée avec un record additionnel OPT. Lorsque la requête DNS AAAA OPT a reçu sa réponse, une nouvelle requête DNS a été lancée, cette fois plus traditionnellement, en utilisant IPv4.

Le record DNS OPT a été introduit dans la RFC 2671 [4] pour permettre aux serveurs DNS de transmettre des informations supplémentaires dans les requêtes et les réponses telles que des options de négociation de protocole, des métriques de performances, des identificateurs de transactions ou encore des clés de sécurité,... OPT est souvent utilisé dans des situations où la sécurité et la performance sont critiques, telles que la résolution DNS sécurisée (DNSSEC), la prévention du détournement de DNS (DNS hijacking) et l'amélioration des performances avec le protocole EDNS (Extension Mechanisms for DNS). [5]

## III. Couche réseau

Cette section abordera plus spécifiquement la couche réseau de l'application Discord. Pour rappel, une adresse réseau permet une identification d'un élément du réseau. Grâce à elle, nous allons pouvoir contacter quelqu'un ou un élément d'un réseau en particulier. Les 2 types d'adresses possibles sont les adresses IPv4, encodées sur 32 bits et les adresses IPv6,encodées sur 128 bits.

### A. Les différentes adresses IPv4

Pour Discord, grâce à l'outil "conversation" de Wireshark, nous avons constaté que le type d'adresses utilisées était majoritairement des adresses au format IPv4 :

- 192.168.x.x : Ce sont nos adresses privées.
- 162.159.x.x : Ce sont les adresses des serveurs Cloudflare utilisés par Discord.
- 66.22.x.x: RIPE (qui est un forum ouvert qui nous renverra vers une certaine section sur internet) renvoie ensuite vers "discord-nlrtm1-1" (qui semble être un serveur contactable via internet).
- 224.0.x.x: D'après whois, cela appartient à MCAST-NET qui correspond à un élément de partage pour les noms de domaine avec DNS.
- 239.255.255.250 : C'est l'adresse qui est utilisée par le protocole SSDP afin de diffuser et découvrir le service de réseau et d'informations.

Ces différentes informations ont été obtenues via la requête whois présente dans le terminal.

### B. NAT

Dans les réseaux informatiques, certains routeurs ont la capacité de réaliser du NAT. Dans ce cas, le routeur fait correspondre des adresses IP privées à d'autres adresses IP publiques [17]. Parmi les solutions qui traversent le NAT [18], nous retrouvons les suivantes : SOCKS, TURN, HOLE PUNCHING, STUN, ICE, IGDP, NAT-PMP, PCP, ALG. Cependant, suite à l'analyse de nos traces, nous ne retrouvons aucun de ces éléments. Nous pensons donc que c'est un concept qui n'est peut-être pas utilisé par Discord.

## IV. Couche Transport

Cette section aborde les différents protocoles de transport qu'utilise Discord afin d'échanger des données.

### A. Les protocoles de transport

Les protocoles que nous retrouvons dans les différents scénarios sont :

- UDP : Est un protocole qui fournit un service de transport sans connexion non fiable en plus du service sans connexion de couche réseau non fiable.
- TCP : Est un protocole qui fournit un service de transport fiable par flux d'octets, orienté connexion, en plus du service réseau sans connexion peu fiable.
- QUIC : Est un protocole de transport qui a l'ambition de remplacer TCP car il réutilise certaines de ses fonctionnalités comme la ré-émission de paquets. De plus, il ajoute quelques fonctionnalités supplémentaires telles que la ré-émission de paquets perdus non bloquants, la gestion de la couche transport par l'application, un chiffrement TLS intégré (TLSv1.3 dans notre cas).

### B. Quic et ses extensions

Le protocole QUIC est présent dans la plupart de nos traces. Nous avons découvert que lorsque nous nous envoyons des messages, il s'agissait du protocole utilisé. Grâce à Wireshark, nous avons déterminé que ce protocole utilise un cryptage différent. En effet, QUIC utilise TLSv1.3 dans ses paquets alors que, dans les autres protocoles, nous retrouvons du TLSv1.2. Nous avons listé les extensions suivantes dans les paquets de négociations d'ouverture de la connexion QUIC : length, server_name, supported_groups, application_layer_protocol_negotiation, signature_algorithms, key_share, psk_key_exhange_modes, early_data, supported_versions, quic_transport_parameters, compress_certificate, application_settings, pre_shared_key. A noter également que nos captures ont été réalisées sur Windows, MacOS et Linux.

### C. D'autres protocoles

En plus des 3 principaux protocoles, nous avons observé le protocole RTCP [6] utilisé pour la collecte de statistiques de performance en temps réel, telles que la qualité de la transmission audio et vidéo, les pertes de paquets et les retards, ainsi que pour la synchronisation des participants à une visioconférence.

## V. Sécurité et Chiffrement

Le chiffrement est un processus qui permet de sécuriser les données envoyées sur un réseau en les rendant illisibles pour toute personne non autorisée. Il s'agit d'un moyen de protection contre les attaques, les interceptions de données ou les écoutes illégales. Le chiffrement utilise des algorithmes mathématiques pour transformer les données en un code

brouillé qui ne peut être déchiffré que par ceux qui possèdent la clé de déchiffrement. Ainsi, seules les personnes ayant accès à la clé peuvent accéder aux données sensibles échangées sur le réseau. [7]

### A. Transport Layer Security

Par défaut, DNS n'est pas sécurisé. Cependant, il existe des versions de DNS comme DNSSEC qui utilisent la cryptographie pour garantir l'authenticité et l'intégrité des données DNS en ajoutant des signatures numériques aux enregistrements DNS. Dans nos analyses, nous n'avons pas observé de tels protocoles. [8]

TLS est un protocole de sécurité qui permet de chiffrer les communications sur internet et de garantir l'authenticité et l'intégrité des données échangées. Il est utilisé pour sécuriser les connexions entre les clients et les serveurs web, ainsi que pour d'autres types de communication sur internet.

Discord utilise la version 1.2 et 1.3 de TLS.

La principale différence entre TLSv1.2 (RFC 5246 [9]) et TLSv1.3 réside dans leurs mécanismes de sécurité. TLSv1.3 (défini par la RFC8446 [10]) a été conçu pour être plus rapide et plus sûr que TLSv1.2 en utilisant des algorithmes de chiffrement plus forts et en supprimant certains mécanismes de sécurité obsolètes. Il utilise également une approche plus simple pour les négociations de clés de chiffrement, ce qui réduit les risques d'attaques liées à cette étape. Une des clé de chiffrement que nous rencontrons est Diffie-Hellman. [11]

*1) L'utilisation de TLSv1.2 dans Discord:* De manière générale, c'est TLS 1.2 qui est utilisé. On observe un handshake à chaque établissement de la connexion TLS entre un client et un serveur Discord. Un message "client Hello" est envoyé au serveur dans lequel le client propose une liste de suites cryptographiques [12] que le client supporte. En échange, le serveur renvoie un message "serveur Hello" dans lequel le serveur précise quelle suite il préfère. Dans notre situation, voici 5 suites que le client propose :

1) TLS_AES_128_GCM_SHA256
2) TLS_AES_256_GCM_SHA384
3) TLS_CHACHA20_POLY1305_SHA256
4) TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
5) TLS_RSA_WITH_AES_128_GCM_SHA256

Le serveur choisi, dans l'ensemble des cas observés, la suite cryptographique suivante : TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256

Il arrive aussi, à certains moments, que le serveur décide de changer de suite. Dans ce cas, un message "Change Cipher Spec Protocol" est envoyé. Cependant, Wireshark ne mentionne pas quelle nouvelle suite cryptographique ou spécification sera utilisée.

*2) L'utilisation de TLSv1.3 dans Discord:* Nous observons du TLS 1.3 afin de sécuriser le protocole QUIC.

### B. Validité des certificats

Les certificats TLS sont délivrés par des autorités de certification (CA) de confiance, telles que Symantec, Comodo, Let's Encrypt, etc [13]. Les navigateurs Web, les clients de messageries et d'autres applications TLS font confiance aux CA pour vérifier l'identité des serveurs Web et délivrer les certificats correspondants. La validité d'un certificat TCP dépend de sa période de validité [13] , qui est déterminée par l'autorité de certification qui l'a délivré. La plupart des certificats sont valables pendant un an, mais certains peuvent être valables pendant plusieurs années. Les navigateurs et les autres applications TLS vérifient que le certificat du serveur est valide et qu'il n'a pas été révoqué avant de permettre une connexion chiffrée.

Dans le cas de Discord, le certificat est valide pour la période allant du 14 octobre 2022 au 14 octobre 2023, soit un an. En effet, depuis le 1er septembre 2020, la limite des certificats TLS a été fixée à 397 jours maximum. Cette décision a été annoncée au forum d'Autorité de certification/Navigateur à Bratislava en mars 2020. Celle-ci a pour but d'augmenter la sécurité sur le web et d'assurer avec davantage de certitude l'identité des éditeurs de sites internet [19]. En revanche, les certificats établis avant cette date et ayant une durée supérieure à 13 mois restent valables jusqu'à leur date d'échéance. Nous avons rencontré ce cas de figure pour le serveur "rotterdam5742.discord.media" pour lequel la période de validité s'étend du 27 janvier 2020 au 31 décembre 2024.

Les certificats sont émis par la société Digicert. Ils utilisent un algorithme ecdsa-with-SHA256.

### C. Chiffrement de UDP

Nous observons beaucoup de trafic UDP dans nos analyses. UDP (User Datagram Protocol) est un protocole de communication permettant d'envoyer des datagrammes de données de manière rapide et efficace, sans garantir la livraison des paquets ou leur ordre de réception à l'inverse de TCP. Il est utilisé dans des applications où une légère perte de données ou un faible retard ne pose pas de problème, comme le streaming de vidéo. Dans le cas de Discord, UDP est utilisé car il n'y a pas de grande importance si un datagramme est perdu. En effet, il n'est pas très grave de perdre quelques frames de la vidéo. En revanche, son utilisation est très utile car il ne nécessite pas de three way handshake comme TCP pour établir une connexion. UDP est donc beaucoup plus rapide pour transmettre des données. Par ailleurs, nous observons plutôt des paquets de données QUIC qui est une évolution de UDP. [14]

## VI. Application

A l'aide de graphiques réalisés avec les librairies Python Pyshark [15] et Matplotlib [16], nous allons vous démontrer quelques constatations.

### A. Liste des protocoles dans l'application

Tout d'abord, lors de l'échange de messages uniquement[2], le protocole de transport UDP n'est pas utilisé mais bien QUIC. Discord utilise donc UDP afin d'envoyer les flux audio et vidéo, car il privilégie l'envoi rapide de flux plutôt que la fiabilité de l'envoi tel que TCP. Par ailleurs, dans le scénario

[2]Scénario 1 : envoi de messages

3 [3] nous n'observons pas de protocole QUIC. Nous avons effectué différents tests afin de déterminer quand ce protocole apparaissait. Nous avons conclu que lorsqu'un message était envoyé avant un appel, nous retrouvions le protocole QUIC en plus de TCP et UDP. Par contre, si aucun message n'est envoyé alors nous ne retrouvons pas ce protocole. Entre les scénarios 2 [4] et 3, on constate un volume plus important de trafic UDP car nous ajoutons le flux vidéo. Entre les scénarios[5] 4 et 5[6], c'est le même constat. Voici des graphiques de diagramme circulaire représentant les différents protocoles présents dans la capture lors des scénarios 3 et 5.

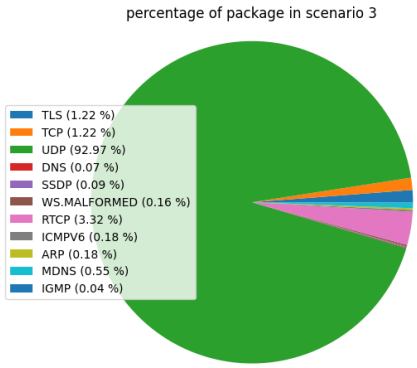


Fig. 1. Graphique circulaire des protocoles réseaux pendant un appel avec caméra

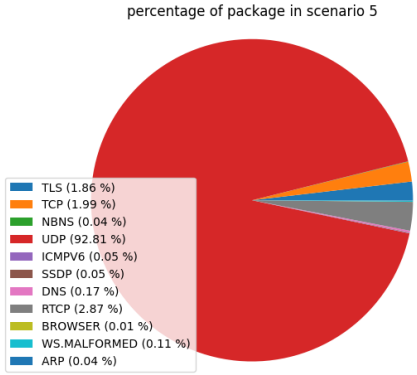


Fig. 2. Graphique circulaire des protocoles réseaux pendant un appel avec caméra et partage d'écran

### B. Débit observé

Concernant le débit par temps, nous avons décidé de compter le nombre de paquets envoyés par période de 15 secondes. Sur les graphiques suivants, nous nous concentrons sur les scénarios 1 et 5.

Le nombre de paquets envoyés par période de 15 secondes pour un envoi de quelques messages est faible (environ 143

[3]Scénario 3 : appel avec caméra
[4]Scénario 2 : appel sans caméra
[5]Scénario 4 : appel sans caméra avec partage d'écran
[6]Scénario 5 : appel avec caméra et partage d'écran

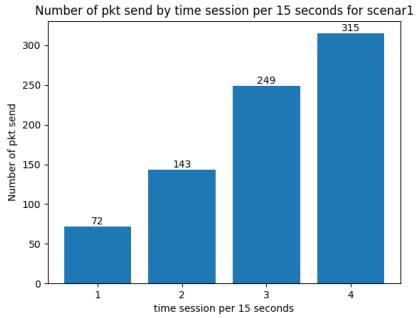


Fig. 3. Débit (via les protocoles réseaux) pendant un échange de message

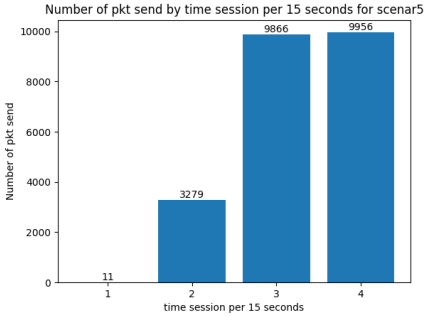


Fig. 4. Débit (via les protocoles réseaux) pendant un appel avec caméra et partage d'écran

paquets au bout de 30 secondes, alors qu'il est presque 22 fois supérieur pour le scénario 5. Ceci s'explique par l'ajout des flux audio et vidéo.

### C. Différence entre les types de connexion (wifi-wifi) + (ethernet+wifi)

Nous avons réalisé des analyses quand un de nous était en ethernet et l'autre en connexion wifi mais également quand nous étions tous les deux sur le même réseau wifi. Nous n'avons pas observé de différence flagrante mis à part que la personne en connexion ethernet avait le record supplémentaire OPT dans ses requêtes DNS comme expliqué dans la section II : DNS de ce rapport.

## VII. Conclusion

Notre analyse montre que Discord utilise de nombreux protocoles tels que DNS, UDP, TCP, QUIC, TLS ou encore IPv4. Discord ne possède aucun serveur propre. Il utilise les infrastructures de Cloudflare. IPv4 est favorisé pour les communications entre les clients et serveurs. Certains protocoles de transports sont privilégiés en fonction des fonctionnalités employées. Ainsi, UDP et TCP sont utilisés pour le flux vidéo et audio, QUIC pour l'envoi et la réception de messages. La majorité des communications sont sécurisées et chiffrées via TLSv1.2. TLSv1.3 est employé pour QUIC. Plus les scénarios utilisent de fonctionnalités, plus le nombre de paquets échangés augmente, majoritaiement de l'UDP.

## REFERENCES


[1] logiciel wireshark : https://www.wireshark.org/
[2] Man page de la commande dig : https://linux.die.net/man/1/dig
[3] Man page de la commande whois : https://linux.die.net/man/1/whois
[4] Référence réseaux RFC 2671 : https://www.ietf.org/rfc/rfc2671.txt
[5] Record Additionnel OPT dans DNS : https://dns.lookup.dog/record-types/OPT
[6] Information concernant RTCP : https://www.napsis.fr/telephonie-ip/rtp-rtcp/
[7] Différente information sur l'encryption : https://www.kaspersky.fr/resource-center/definitions/encryption, https://fr.wikipedia.org/wiki/Chiffrement, https://www.cloudflare.com/fr-fr/learning/ssl/what-is-encryption/
[8] Chiffrement de DNS : https://www.akamai.com/fr/blog/trends/dnssec-how-it-works-key-considerations, https://www.icann.org/resources/pages/dnssec-what-is-it-why-important-2019-03-20-fr
[9] Référence Réseaux RFC 5266 : https://datatracker.ietf.org/doc/html/rfc5246.html
[10] Référence Réseaux RFC 8446 : https://datatracker.ietf.org/doc/html/rfc8446.html
[11] Différentes informations concernant le protocole TLS : https://www.websecurity.digicert.com/fr/ca/security-topics/what-is-ssl-tls-https, https://fr.wikipedia.org/wiki/Transport_Layer_Security, https://beta.computer-networking.info/syllabus/default/protocols/tls.html, https://www.cloudflare.com/fr-fr/learning/ssl/why-use-tls-1.3/, https://blog.cloudflare.com/rfc-8446-aka-tls-1-3/
[12] Les suites cryptographiques : https://fr.wikipedia.org/wiki/Suite_cryptographique
[13] Information sur la validité des certificats TLS : https://docs.digicert.com/fr/certcentral/manage-certificates/end-of-2-year-dv–ov–and-ev-public-ssl-tls-certificates.html
[14] information sur le chiffrement de UDP : https://www.cloudflare.com/fr-fr/learning/ddos/glossary/user-datagram-protocol-udp/, https://fr.wikipedia.org/wiki/User_Datagram_Protocol, https://beta.computer-networking.info/syllabus/default/protocols/udp.html
[15] Man page librairie Python Pyshark : https://github.com/KimiNewt/pyshark/
[16] Man page librairie Python Matplotlib : https://matplotlib.org/
[17] Information concernant NAT : https://fr.wikipedia.org/wiki/Network_address_translation
[18] Information pour traverser NAT : https://en.m.wikipedia.org/wiki/NAT_traversal
[19] Certificats SSL/TLS: durée de validité maximale fixée à 13 mois : https://www.globalsign.com/fr/blog/certificats-ssl-tls-duree-validite-maximale-un-an