Projet 1 - Analyse d'applications réseaux

Cédric Kheirallah
SINF13BA
UCLouvain-EPL
Louvain-La-Neuve, Belgium
cedric.kheirallah@student.uclouvain.be

Quentin Massinon
SINF13BA
UCLouvain-EPL
Louvain-la-Neuve, Belgium
quentin.massinon@student.uclouvain.be

***Abstract*—Ce document présente une analyse du trafic réseau de l'application de stockage en ligne OneDrive dans le cadre du cours LINFO1341-Computer networks**

***Index Terms*—LINFO1341, Computer network, OneDrive, paquet, protocole, DNS, UDP, TCP, IPv4, IPv6, WiFi, Ethernet, 4G**

I. Introduction

En tant qu'ingénieur·e réseau, il est important de pouvoir comprendre le fonctionnement des protocoles qui constituent l'Internet et également le comportement de ses applications. C'est en combinant ces deux savoirs qu'un·e ingénieur·e réseau peut à la fois améliorer les protocoles de l'Internet et concevoir des applications réseaux efficaces. Dans ce projet, nous analysons l'application réseau de stockage et partage de fichiers, OneDrive, une application très commune qui comporte beaucoup de comportements différents, chacun alliant des exigences particulières se traduisant en des utilisations et combinaisons de protocoles différentes.

A. Méthodologie

Cette analyse se base sur des captures de données obtenus via Wireshark dans plusieurs scénarios différents effectués l'un après l'autre :

1) l'ouverture de l'application
2) le chargement d'un fichier
3) l'ouverture d'un fichier
4) la modification d'un fichier
5) le téléchargement d'un fichier

Les analyses se feront d'abord à partir de la capture WiFi et sera ensuite comparé avec les captures 4G et Ethernet dans le cas où nous observerons des différences majeures (NOTE : nous n'avons au final pas eu le temps de comparer la capture WiFi à Ethernet et 4G)
Les données de capture ainsi que les enregistrements visuels des captures sont disponibles à cette adresse : https://github.com/CC4K/LINFO1341-Projet_1

B. Matériel utilisé

1) Capture WiFi: Les tests via WiFi ont été réalisés sur un navigateur Microsoft Edge version 123.0.2420.65 (Official build) (64-bit) sur un système Windows 10 Education version 22H2 (64-bit). Le routeur est une bbox3 de Proximus. Les adresses de l'ordinateur en question sont 192.168.1.59 et 2a02:a03f:6b2f:cb00:3137:2f4d:fc94:b6d5

2) Capture Ethernet: Les tests via Ethernet ont été réalisés avec le navigateur Mozilla Firefox v 124.0.1, sur un système Ubuntu 22.04.4 LTS en 64-bit. Le routeur est une b-box3 de chez Proximus dont le serveur DNS est configuré sur 195.238.2.22 et 195.238.2.21, ces adresses sont celles des serveurs DNS de Proximus.

3) Capture 4G: Pour les tests en 4G, le relais WiFi est un smartphone Huawei P30 Pro utilisant EMUI en version 12.0.0. La machine de test est la même que pour les tests Ethernet.

II. DNS

A. Domaines résolus

TABLE I
Domaines et adresses résolues

Nom de domaine	Adresse
onedrive.live.com	13.107.137.11
graph.microsoft.com	2603:1027:1:150::80
edgeservices.bing.com	2a02:a000:1:213::51f3:1d2
api.onedrive.com	13.107.42.12
login.live.com	20.190.177.21
skyapi.onedrive.live.com	40.90.136.180
storage.live.com	13.104.208.160
login.microsoftonline.com	2603:1026:3000:118::3
ams03pap001files.storage.live.com	13.107.42.12
nhwb7q.am.files.1drv.com	13.107.42.12
m365cdn.nel.measure.office.net	2a02:a000:1:213::51f3:121

Quelque soit le canal de communication, lors de la résolution DNS, les requêtes ne sont pas directement effectuées par l'ordinateur vers le serveur DNS. Elles sont faites au routeur qui effectuera alors les requêtes aux serveurs DNS et transférera la réponse du serveur à l'ordinateur.

B. Serveurs autoritatifs + différentes entreprises

Liste des serveurs autoritatifs :

- spov-msedge.net → Microsoft SharePoint
- (westeurope.cloudapp.azure.com) → Microsoft Azure
- (northeurope.cloudapp.azure.com) → Microsoft Azure
- trafficmanager.net → Microsoft Azure
- l-msedge.net → Azure Front Door
- a-msedge.net → Azure Front Door
- 1drv.com → Microsoft OneDrive
- akadns.net → Akamai (service provider for applications including Microsoft-365 (includes OneDrive)) → https://techdocs.akamai.com/eaa/docs/apps

Ces serveurs appartiennent presque tous à Microsoft à l'exception du dernier qui appartient à Akamai, un fournisseur de services pour plusieurs application y compris Microsoft 365 et donc par extension OneDrive.

C. Types de requêtes DNS

- AAAA : utilisé pour trouver le nom du serveur correspondant à une adresse IPv6
- A : utilisé pour trouver le nom du serveur correspondant à une adresse IPv4
- CNAME : utilisé pour trouver le nom de domaine réel à partir de son alias
- HTTPS : utilisé pour faire des requêtes DNS sécurisées sur HTTPS

D. Version IP préférée

L'application demande à chaque fois l'adresse IPv4 et IPv6 pour vérifier la compatibilité du serveur. Très peu de requêtes DNS ont cependant été effectués en IPv6 (4-5).

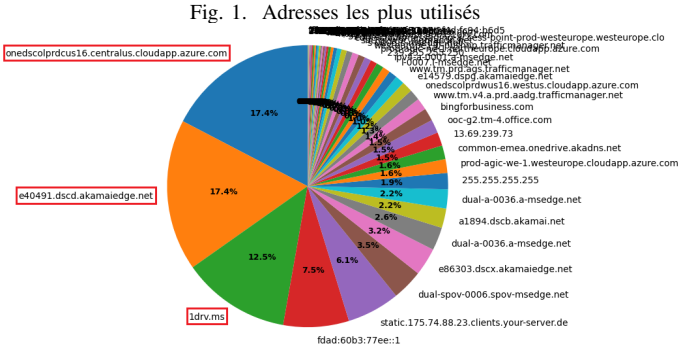
E. Conclusion

On remarque que peut importe le type de connection utilisé, il y a aura toujours une double requête DNS : une en A et une autre pour le même domaine en AAAA.

III. Couche réseau

A. Adresses les plus utilisées

Fig. 1. Adresses les plus utilisés



Les 2 adresses de loin les plus utilisés sont `onedscolprdcus16.centralus.cloudapp.azure.com` et `e40491.dscd.akamaiedge.net` En effet ils appartiennent respectivement aux domaines azure.com et akamaiedge.net que nous avions déjà déterminé comme les majeurs serveurs autoritatifs plus tôt.

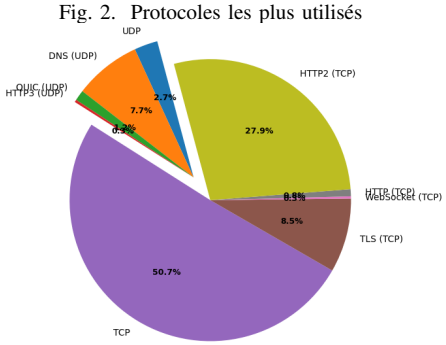
B. NAT

Aucun protocole NAT n'apparaît lors de nos captures.

IV. Couche transport

A. Protocoles utilisés

Fig. 2. Protocoles les plus utilisés



On peut constater que la très grande majorité des protocoles utilisés (88.1%) sont de type TCP. Les 11.9% restants le protocole UDP.

1) TLS sur TCP: Les protocoles TCP se font sur le port 443 dédié aux transferts HTTPS et sert également à utiliser les certificats TLS par dessus (TLS sur TCP). Les différents protocoles utilisant TLS sur TCP sont : HTTP, HTTP2, WebSocket et bien sûr TLSv1.2 et TLSv1.3 .

2) UDP: Ce protocole est utilisé en très grande majorité pour effectuer des requêtes DNS et est parfois utilisé pour transférer du QUIC*. Les conversations UDP se font principalement sur le port 53.
Les protocoles UDP autre que DNS et QUIC sont :

- Link-Local Multicast Name Resolution [1] : LLMNR est un protocole basé sur le format DNS qui autorise à la fois des hôtes IPv4 et IPv6 de réaliser des résolution pour des hôtes du même lien local.
- Multicast DNS [2] : mDNS est un protocole qui résout des noms d'hôtes pour des adresses IP à l'intérieur de réseaux de tailles restreintes qui n'incluent pas un nom de serveur local.
- NetBIOS Name Service [3] : afin de distribuer des datagrammes, une application doit enregistrer son nom NetBIOS en utilisant le NetBIOS Name Service (NBNS). Ne supporte pas l'IPv6.
- Simple Service Discovery Protocol [4] : SSDP est un protocole basé sur HTTPU qui utilise UDP comme protocole de transport. Les services sont annoncés par le système hôte grâce à du multicast vers une adresses IP spécifique (239.255.255.250 en IPv4) sur le port UDP 1900.

B. QUIC

Ce protocole est utilisé presque exclusivement à l'ouverture de l'application sur le port 443 (port dédié aux transfert HTTPS via TCP et QUIC/UDP). QUIC est utilisé sous deux formes différentes : du QUIC simple qui est une alternative au protocole TCP et du QUIC utilisé pour faire du HTTP3.
Lors du handshake QUIC, le client envoie les extensions suivantes [5] :

1) Client Hello:

- `server_name (len=25) name=substrate.offic`
- `supported_groups (len=8)`
- `signature_algorithms (len=20)`
- `application_layer_protocol_negotiation (l`
- `supported_versions (len=3) TLS 1.3`
- `psk_key_exchange_modes (len=2)`
- `key_share (len=38) x25519`
- `quic_transport_parameters (len=104)`
- `application_settings (len=5)`
- `compress_certificate (len=3)`
- `encrypted_client_hello (len=282)`

Et le serveur renvoie sa réponse dont les extensions sont :

2) Server Hello:

- `supported_versions (len=2) TLS 1.3`
- `key_share (len=2) secp384r1`

et dans à la fin du handshake on peut observer que certains des paramètres négociés ont été mis en place : par exemple `Signature Algorithm: rsa_pss_rsae_sha256 (0x0804)` (la plupart sont cryptés donc illisibles).

C. Connections multiples sur un même domaine

TABLE II
CONNECTIONS SUR UN MÊME DOMAINE

Temps [sec]	Domaine
0.751459	login.live.com
1.380260	login.live.com
2.738962	skyapi.onedrive.live.com
2.809227	skyapi.onedrive.live.com
36.688431	res-1.cdn.office.net
37.152817	res-1.cdn.office.net

Ces connections multiples s'expliquent lorsque l'on fait un `whois` sur ces domaines :

- login.live.com : associé à Microsoft Auth qui offre du single sign-on (SSO) aux différents services et applications de Microsoft.
- skyapi.onedrive.live.com : associé à Microsoft OneDrive
- res-1.cdn.office.net : associé à Microsoft Office qui gère les sous-applications de Office (dont OneDrive)

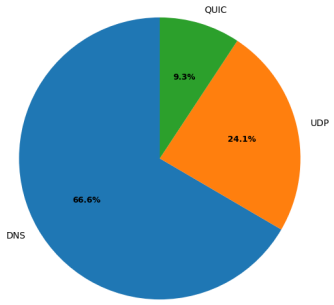
La seule et unique différence entre les 2 paquets dans les 3 cas est le `encrypted_client_hello` qui est différent (mais malheureusement crypté donc on ne peut pas savoir quel est la vraie différence)

V. CHIFFREMENT ET SÉCURITÉ

A. UDP

Ce protocole n'est pas sécurisé quand utilisé seul car il utilise des datagrammes et n'est pas crypté.

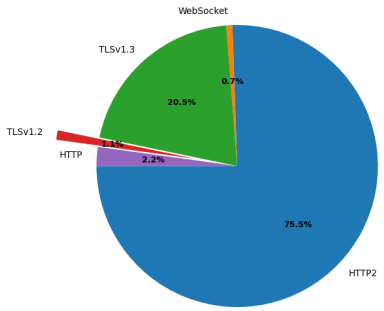
Fig. 3. Différents protocoles utilisant UDP



Une grande majorité (66.6%) du trafic UDP est utilisé afin d'effectuer des requêtes DNS. La totalité du trafic QUIC utilise du DNS (dont 10 paquets HTTP3) et est quand à lui scurisé car le protocole QUIC a sa propre encryption.
Le reste du trafic UDP est un bruit de fond provenant du système d'exploitation.

B. Versions de TLS utilisés

Fig. 4. Différentes versions de TLS



Les premiers "Client Hello" entre le client et le serveur se font en TLSv1.2 pour des raisons de compatibilité et passe bien vite en TLSv1.3 pour le reste des transmissions.
On peut observer une majorité de HTTP2 utilisant du TLSv1.3 et sert à transmettre toutes les informations pour charger et afficher la page de l'application à l'utilisateur, transférer les JS, JSON, txt et HTML... (cfr paquets 862, 2270 et 3326 dans capture_wifi.pcqpng pour clairement lire les textes affichés à ces moments là).
Très peu de protocoles HTTP sont présent et sont exclusivement utilisé pour transmettre les options au navigateur tels que les cookies, les icônes, les options de l'utilisateur...
Enfin le protocole WebSocket sert à mettre fin aux connexions (transmet [FIN]).

C. Algorithmes de chiffrement

Deux différents algorithmes de chiffrements ont été observés dans les paquets "Server Hello" en TLSv1.3 et les "HANDSHAKE" en QUIC :

1) TLS_AES_256_GCM_SHA384: Les domaines ayant établis une connection utilisant ce chiffrement sont :

- graph.microsoft.com
- res-1.cdn.office.net
- res.cdn.office.net

L'algorithme de chiffrement TLS_AES_256_GCM_SHA384 utilise une encryption "Advanced Encryption Standard with 256bit key in Galois/Counter mode (AES 256 GCM)" et un hashage "Secure Hash Algorithm 384 (SHA384)". Autrement dit c'est un algorithme hautement sécurisé [6] qui supporte en plus le PFS (Perfect Forward Secrecy) mais au prix de rare soucis de compatibilité.

2) TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384: Les domaines ayant établis une connection utilisant ce chiffrement sont :

- onedrive.live.com
- api.onedrive.com
- login.live.com
- 4.bing.com
- ams03pap001files.storage.live.com
- eu-mobile.events.data.microsoft.com
- nhwb7q.am.files.1drv.com
- self.events.data.microsoft.com

L'algorithme de chiffrement TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 utilise une encryption "Advanced Encryption Standard with 256bit key in Galois/Counter mode (AES 256 GCM)" et un hashage "Secure Hash Algorithm 384 (SHA384)" (exactement similaire au TLS_AES_256_GCM_SHA384). Cette algorithme est considéré bien sécurisé et bien qu'il ne supporte pas le PFS (Perfect Forward Secrecy), il est compatible avec la grande majorité des systèmes d'exploitations et navigateurs existants.

D. Durée de vie

TABLE III
DURÉE DE VIE DES CERTIFICATS DNS

Domaine	TTL [sec]
onedrive.live.com	10-53
res-1.cdn.office.net	59-247
skyapi.onedrive.com	69-179
storage.live.com	40-109
login.live.com	100-282
m365cdn.nel.measure.office.net	206-290
api.onedrive.com	203-867
edgeservices.bing.com	6442
graph.microsoft	4640-10561

bing.com (edgeservices.bing.com) et Microsoft Auth (graph.microsoft) renvoie des réponses DNS avec de très longues durées de vie car ces serveurs occupent une place importante sur Internet et sont très utilisés. Par conséquent ils renvoient toujours les mêmes réponses DNS et n'ont pas besoin d'une durée de vie courte contrairement aux autres domaines qui doivent parfois mettre à jours leurs réponses et fixent une courte durée de vie (<1000 secondes).

VI. APPLICATION

A. Upload Vs Modification

Commençons par observer ce qui se passe (en résumé) lors d'un upload de fichier et lors d'une modification de fichier préexistant.

1) Upload d'un nouveau fichier:

- Connection à microsoft.com (13.69.239.73 / self.events.data.microsoft.com): Client Hello, Server Hello, handshake
- Client fait un [OPTIONS] et envoie `createUploadSession` à OneDrive (13.107.42.12) et envoie un [PING]
- Server réponds 200 OK et [PONG]
- Client fait un [POST] et réenvoie `createUploadSession` à OneDrive : (même lien)
- Client et Server s'échange les données du fichier : JavaScript, Données Textuelles (voir paquet 1650 pour texte du fichier de test)
- Client fait un [OPTIONS] et envoie un token pour terminer l'upload
- Server réponds 200 OK
- Client fait un [GET] et réenvoie le token : (même lien)

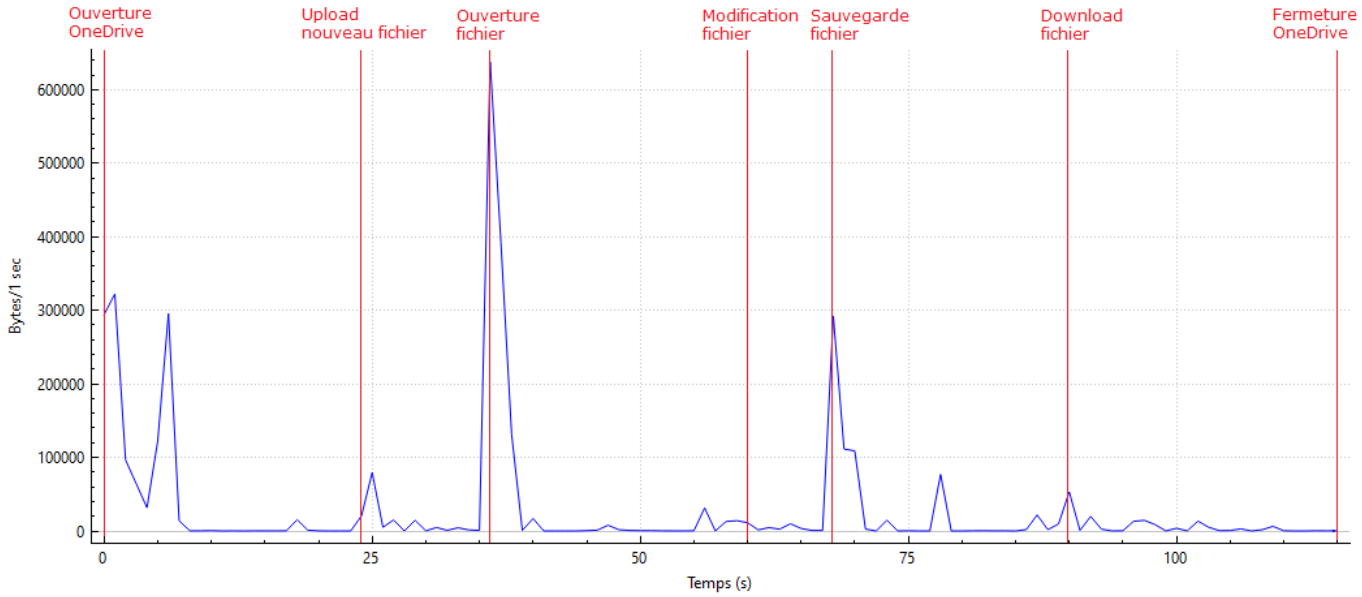
2) Modification d'un fichier existant:

- Connection à Microsoft Azure hosting service (51.11.192.48 / eu-mobile.events.data.microsoft.com) : Client Hello, Server Hello, handshake
- Client fait un [OPTIONS] et envoie `createUploadSession` à OneDrive (13.107.42.12) et envoie un [PING]
- Server réponds [PONG] et 200 OK
- Client fait un [POST] et réenvoie `createUploadSession` à OneDrive : (même lien)
- Client fait un [OPTIONS] et envoie ses préférences au Server
- Server réponds 200 OK
- Client fait un [PUT] et réenvoie ses préférences à OneDrive : (même lien)
- Client envoie le texte mis à jour du fichier modifié directement après (voir paquet 2839)
- Client fait un [OPTIONS] et envoie `CheckForChanges` au Server
- Server réponds 200 OK
- Client fait un [GET] et réenvoie `CheckForChanges` au Server : (même lien)

La seule différence majeures de comportement entre les deux scénarios est que lors de l'upload, le Client fait un [POST] pour envoyer le fichier alors que lors de la modifications, un [PUT] envoie les données au Server. Le reste du comportement est similaire : connection au même serveur de OneDrive, même demande de `createUploadSession`, PING et PONG entre Client et Server fait à des moments similaires, etc

B. Volume de données échangées

Fig. 5. Volume de données échangées en Bytes/sec



Le graphique de Wireshark parle de lui-même : les moments importants de nos scénarios sont clairement observables dans le débit de données envoyés. On notera que l'ouverture d'un fichier dans OneDrive est le premier consommateur de données suivi de l'ouverture de l'application et de la modification d'un fichier (lorsque l'on clique sur le bouton de sauvegarde plus particulièrement). Le transfert de fichier via upload ou download n'est pas aussi consommateur que l'on pourrais le penser au premier abord.

C. Utilisateurs multiples

Nous n'avons malheureusement pas eu l'occasion de tester l'application avec plusieurs utilisateurs simultanés.

References

[1] Wikipedia, Link-Local_Multicast_Name_Resolution
[2] Wikipedia, Multicast_DNS
[3] Wikipedia, NetBIOS
[4] Wikipedia, Simple_Service_Discovery_Protocol
[5] IETF, rfc8446, Page 26
[6] Ciphersuite, FAQ