Réduction de l'impact du paradigme REST sur le DNS en utilisant des technologies pour l'internet des objets

Arnol Lemogue

COLLEGES BRETAGNE LOIRE
SCIENCES POUR L'INGENIEUR ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
MINES-TÉLÉCOM ATLANTIQUE BRETAGNE
PAYS-DE-LA-LOIRE - IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 648
Sciences pour l'Ingénieur et le Numérique
Spécialité : *Informatique*

Par
Arnol LEMOGUE

Réduction de l'impact du paradigme REST sur le DNS en utilisant des Technologies pour l'Internet des Objets.

Thèse présentée et soutenue à IMT Atlantique Campus Rennes, le 04 décembre 2023
Unité de recherche : IRISA
Thèse N° : 2023IMTA0373

Rapporteurs avant soutenance :

André-Luc BEYLOT — Professeur, ENSEEIHT, Institut National Polytechnique de Toulouse
Kinda KHAWAM — Maître de Conférences HDR, Université de Versailles

Composition du Jury :

Président :	Fabrice VALOIS	Professeur, INSA Lyon
Examinateurs :	André-Luc BEYLOT	Professeur, ENSEEIHT, Institut National Polytechnique de Toulouse
	Kinda KHAWAM	Maître de Conférences HDR, Université de Versailles
	Andrzej DUDA	Professeur, Grenoble INP-Ensimag
	Ivan MARTINEZ	Ph.D., RAN Software Researcher, Nokia Bell Labs
Dir. de thèse :	Laurent TOUTAIN	Professeur, IMT Atlantique
Encadrant de thèse :	Ahmed BOUABDALLAH	Maitre de Conférence, IMT Atlantique

Invités :

Fabienne NOUVEL — Professeure émérite, INSA Rennes
Benoît AMPEAU — Directeur partenariats et innovation, AFNIC

Je dédie ce travail à la mémoire de mes grands-pères Kemguimdock (1930-2010) et Kemlemogue (1945-2018), qui étaient d'éminents dignitaires de la chefferie Fossong Ellelem (Ouest-Cameroun)

Remerciements

Mon grand-père me disait toujours de son vivant, je cite : *"Mon jeune petit fils ! une seule main ne peut attacher un paquet, et tu dois toujours être reconnaissant envers ceux ou celles qui t'ont tendu la main quand tu en avais besoin"*. J'aimerais donc exprimer ma gratitude envers toutes les personnes, qui de près ou de loin, m'ont apporté leur soutien durant ces trois années.

J'aimerai exprimer mes sincères remerciements en premier lieu à Laurent Toutain, ainsi qu'à Ahmed Bouabdallah, respectivement Professeur et Maître de Conférences au Département Systèmes Réseaux, Cybersécurité et Droit du Numérique (SRCD) à IMT Atlantique, pour leur attention, leurs conseils avisés, et leur précieuse écoute qui ont grandement contribué à la réussite de cette thèse. Merci de m'avoir accepté avec mes défauts. Grâce à vous, j'ai pu tenir la promesse que j'avais faite à mon grand-père (mon modèle) avant sa mort en Mai 2010 à la veille de mon examen de brevet d'études secondaires. Je vous serai éternellement reconnaissant. J'ai été honoré de collaborer avec vous.

Je souhaite exprimer mes remerciements aux membres du jury qui ont eu la gentillesse de juger ce travail malgré leurs emplois du temps chargés en cette fin d'année 2023. Je tiens à remercier tout particulièrement M. André-Luc Beylot, professeur des universités à l'ENSEEIHT, Institut National Polytechnique de Toulouse, et Mme Kinda Khawam, Maître de Conférences et HDR à l'Université de Versailles Saint-Quentin-en-Yvelines, d'avoir accepté d'être rapporteurs de cette thèse.
Mes remerciements vont également à Mme Fabienne Nouvel, professeure émérite à l'INSA Rennes et M. Julien Montavont, Maître de Conférences à l'Université de Strasbourg, les membres de mon comité de suivi de thèse pour leurs précieux conseils. Merci également à M. Fabrice Valois, professeurs des Universités à l'INSA Lyon et M. Andrzej DUDA, professeurs des Universités à Grenoble INP-Ensimag qui ont accepté d'examiner ce travail.

Un grand merci à Ivan Martinez, anciennement Postdoc à IMT Atlantique et aujourd'hui RAN Software Researcher chez Nokia Bell Labs. C'est une chance pour moi de t'avoir rencontrer, mon cher Ivan, tu as été un grand soutien pour moi pendant les deux premières années de cette thèse. Tu fais partie des personnes que j'ai le plus sollicitées avec mes questions durant cette thèse. Merci encore pour ta disponibilité.

Cette thèse s'inscrit dans le cadre du projet DiNS, un consortium réunissant des acteurs industriels et des instituts académiques tels que IMT Atlantique - IRISA, Afnic, Bouygues Telecom, ACKLIO et INP Grenoble. Ce consortium a été formé en réponse à l'appel à projets de l'ANR 2019. Je souhaite également exprimer ma gratitude envers tous les collègues de ce consortium et en particulier envers Benoit Ampeau, Sandoche Balakrichenan de l'Afnic et Philippe Cola de Bouygues Telecom pour m'avoir accueilli tout au long de cette thèse. Nos différentes réunions ont été une expérience très enrichissante pour moi.

Je remercie également l'Institut Mines-Telecom (IMT), mon employeur, pour avoir financé mes voyages en France et à l'étranger. Ces voyages m'ont enrichi, et aujourd'hui, je suis plus ouvert à la différence et à la diversité.

Oh ! mes collègues exceptionnels de l'association des doctorants ADER, je tiens à vous exprimer ma gratitude pour tous les moments que nous avons partagés, que ce soit à travers nos jeux ou lors de nos repas cuisinés à la MAISEL. Je pense à Gwen, Tania, Modou, Renzo, Awaleh, Pierre-Marie, Ndao, Manel et Leo (je m'excuse auprès des collègues non cités). Merci aussi pour la confiance que vous m'avez accordée en me désignant comme secrétaire général.
Un grand merci s'adresse à tous les collègues du département SRCD et à l'équipe de recherche OCIF (Objets communicants pour l'Internet du Futur) du laboratoire IRISA pour leur bienveillance. Je pense particulièrement à Sandrine FROUIN, assistante du chef de Département SRCD.

Ma gratitude s'adresse aussi à Sonfack serge, Tsafack Emmanuel et à la grande communauté Fossong-Ellelem. Merci de m'avoir accueilli en France et pour les nombreux appels téléphoniques que vous avez passés pour prendre de mes nouvelles et surtout pour me motiver.

Enfin, je souhaite évidemment remercier ma famille pour son soutien inébranlable tout au long de cette thèse, en particulier pendant les moments de doute : À ma tendre épouse MSH (comme j'aime bien t'appeler), merci d'avoir accepté que je m'éloigne de toi pendant tant d'années pour poursuivre mon rêve. Mes parents et grand-mères, je vous remercie pour vos prières et les nombreux sacrifices que vous avez consentis pour moi. Ce Doctorat est autant le vôtre que le mien.

Résumé

Le système de noms de domaines (DNS) utilisé dans l'Internet pour établir une correspondance effective entre les identifiants réseaux et leurs déclinaisons conviviales, s'appuie sur une architecture optimisée et résiliente d'une grande stabilité. Toutefois la préservation de l'efficacité des protocoles sous-jacents, n'a pas permis à ces derniers d'évoluer au même rythme que le reste de l'Internet. L'introduction du paradigme REST ouvre néanmoins la voie à l'évolution du DNS via sa ***RESTification***. Cette dernière entraîne cependant une augmentation de la latence et de la verbosité qui pourraient à terme compromettre sa généralisation.

Dans cette thèse, nous nous intéressons à la réduction de l'impact de la RESTification sur le DNS. Notre exploitation des avancées obtenues dans le domaine de l'Internet des objets, nous conduit à deux contributions significatives concernant respectivement la taille des données véhiculées par les protocoles, et l'architecture du DNS. Nous définissons ainsi un nouveau format reposant sur CBOR pour encoder efficacement les messages DNS. Nous introduisons une nouvelle représentation appelée ***efficient CBOR (e-CBOR)*** qui fournit des formats DNS/DNSSEC compacts et flexibles tout en préservant la compatibilité avec les protocoles existants. Les tests effectués sur l'implémentation d'e-CBOR montrent une réduction substantielle par rapport à la taille des messages encodés dans le format traditionnel. La seconde contribution concerne la conception et la mise en oeuvre d'une nouvelle entité le ***DNS-Broker***. Ce dernier rend possible une ***résolution DNS privée*** pour contrôler l'accès à une ressource réseau, tout en réduisant le nombre de messages échangés dans le cas où la résolution est autorisée.

Un démonstrateur permet de valider la pertinence de nos propositions en les appliquant au cas d'usage de l'itinérance entre opérateurs de réseaux LoRaWAN.

Mots clés : DNS, DoH, REST, IoT, CBOR, CoAP, Itinérance, LoRaWAN

Abstract

The Domain Name System (DNS) used on the Internet to provide an effective correspondence between network identifiers and their user-friendly variants, is based on a highly stable, optimised and resilient architecture. However, maintaining the efficiency of the underlying protocols has not allowed them to keep pace with the rest of the Internet. The introduction of the REST paradigm has nevertheless paved the way for the evolution of DNS via its ***RESTification***. However, the latter leads to an increase in latency and verbosity, which could ultimately compromise its widespread use.

In this thesis we focus on reducing the impact of RESTification on DNS. Leveraging advances in the Internet of Things, we make two significant contributions concerning the size of data carried by protocols, and DNS architecture. We define a new format based on CBOR for efficient encoding of DNS messages. We present a new representation, called ***efficient CBOR (e-CBOR)***, which provides compact and flexible DNS/DNSSEC formats while maintaining compatibility with existing protocols. Tests performed on the e-CBOR implementation show a significant reduction in message size compared to messages encoded in the traditional format. The second contribution concerns the design and implementation of a new entity called ***DNS-Broker***. It allows ***private DNS resolutions*** to control access to network resources, while reducing the number of messages exchanged when these resolutions are authorised.

A demonstrator allows us to validate the relevance of our proposals by applying them to the use case of roaming between LoRaWAN network operators.

Key words : DNS, DoH, REST, IoT, CBOR, CoAP, Roaming, LoRaWAN

Table des Matières

Table des Figures

Liste des Abbréviations

5G – Fifth Generation Mobile Radio.

ABP – Activation By Personalization.
AC – Autorité de Certification.
AppEUI – Application Extended Unique Identifier.
ARPANET – Advanced Research Projects Agency Network.
AS – Application Server.
ASCII – American Standard Code for Information Interchange.

CA – Certificate Authority.
CBOR – Concise Binary Object Representation.
CoAP – Constrained Application Protocol.
CoRE – Constrained RESTFul Environment.

DASH – Dynamic Adaptive Streaming over HTTP.
DevEUI – Device Extended Unique Identifier.
DNS – Domain Name System.
DoC – DNS over CoAP.
DoDTLS – DNS over DTLS.
DoH – DNS over HTTPS.
DoQ – DNS over QUIC.
DoT – DNS over TLS.
DTLS – Datagram Transport Layer Security.

e-CBOR – Efficient Concise Binary Object Representation.
EDHOC – Ephemeral Diffie-Hellman Over COSE.

fNS – Forwarding Network Server.

gTLD – generic top level domain.

HLS – HTTP Live Streaming.
hNS – Home Network Server.
HTTP – HyperText Transfer Protocol.

IdO – Internet des Objets.
IETF – Internet Engineering Task Force.
IMAP – Internet Message Access Protocol.
IoT – Internet of Things.

JoinEUI – Join Request Extended Unique Identifier.
JS – Join Server.
JSON – JavaScript Object Notation.

LoRa – Long Range.
LoRaWAN – Long Range Wide Aera Network.
LTE-M – Long Term Evolution Cat M1.

MQTT – Message Queuing Telemetry Transport.
MX – Mail Exchange.

NB-IoT – Message Queuing Telemetry Transport.
NO – Opérateur Réseau ou Network Operator.

OSCORE – Object Security for Constrained RESTful Environments.
OTAA – Over The Air Activation.

PKI – Public Key Infrastructure.
POP – Post Office Protocol.

REST – Representational State Transfer.
RFC – Request For Comments.
ROA – Resource-Oriented Architecture.
RR – Resource Record.
RTT – Round-Trip Time.

SMTP – Simple Mail Transfer Protocol.
sNS – Serving Network Server.

TLS – Transport Layer Security.
TTN – The Things Network.

UDP – User Datagram Protocol.
URI – Universal Resource Identifiers.

VN – Réseau visité ou Visited Network.

XML – eXtensible Markup Language.
XMPP – Extensible Messaging and Presence Protocol.

Part I

Contexte et État de l'Art

Chapter 1

Introduction Générale

In protocol design, perfection has been reached not when there is nothing left to add, but when there is nothing left to take away.

— **RFC 1925, Nr.12**
(The Twelve Networking Truths)

Sommaire

Depuis longtemps, l'être humain a toujours cherché à établir une communication avec ses semblables en utilisant divers moyens. Avant l'avènement des technologies de télécommunication, la seule façon de faire parvenir un message à distance était d'envoyer une lettre au destinataire. Cependant, l'expéditeur devait écrire l'adresse sur l'enveloppe pour que la lettre parvienne à destination, quel que soit l'itinéraire emprunté. Peu après, en raison de l'importance croissante du courrier en circulation, les boîtes postales ont été introduites, ce qui a permis à l'expéditeur de simplifier l'écriture de l'adresse en indiquant simplement le numéro de boîte postale du destinataire. Ce numéro était ensuite converti en adresse complète par les services postaux pour acheminer le courrier jusqu'à sa destination.

De manière similaire au courrier traditionnel, les machines ont également besoin d'adresses indiquées préalablement par l'utilisateur pour communiquer entre elles. Par exemple, pour consulter un site web, l'utilisateur doit obligatoirement fournir à son navigateur l'adresse du serveur où se trouve la page désirée. Cette adresse est généralement exprimée sous la forme d'une adresse IP, composée de quatre nombres séparés par des points (ex : 192.168.0.1). Cependant, ce type d'adressage n'est pas du tout intuitif pour les humains, d'autant plus avec l'introduction d'IPv6, qui rend l'adressage encore plus complexe en utilisant des séquences encore plus longues, de nombres. Cela a ainsi suscité la nécessité de le rendre plus pratique et facile à mémoriser. La solution trouvée fut de transformer cette série de numéros en une série de caractères ou de noms de domaine, qui seraient bien plus compréhensibles pour les êtres humains. Ainsi est apparu un service de traduction de noms de domaine en adresses IP, ce que nous appelons aujourd'hui le Domain Name System (DNS).

1.1 Contexte et formulation du problème

Ces dernières années, on a constaté un intérêt croissant pour que le protocole DNS [102, 103] garantisse un certain niveau de sécurité, étant donné que le trafic DNS est nativement non chiffré [14]. À cette fin, diverses architectures ont été développées à l'Internet Engineering Task Force (IETF) pour ajouter une couche de sécurité supplémentaire afin de garantir au moins l'intégrité et la confidentialité. La figure 1.1(A) compare les différents protocoles nécessaires pour mettre en œuvre cette couche de sécurité entre un client et un serveur. Trois protocoles principaux sont apparus : DNS over TLS (DoT) [60], DNS over HTTPS (DoH) [57] et DNS over QUIC (DoQ) [31].
Comme indiqué dans [80], l'utilisation de l'une ou l'autre de ces techniques est beaucoup plus verbeuse qu'une requête et une réponse DNS classiques sur UDP. En outre, le transport pour DoT et pour DoH est basé sur TCP, qui nécessite plusieurs RTT[1] pour établir une

[1]Le Round-Trip Time (RTT) dans le DNS représente le temps nécessaire pour qu'un message ou une requête DNS soit envoyé depuis un client vers un serveur DNS distant, traité par ce serveur, puis que la réponse correspondante soit renvoyée du serveur au client. Le RTT est souvent utilisé comme mesure de la latence ou de la réactivité d'un système distribué.

connexion et une clé, ce qui consomme également de la bande passante et introduit un temps de latence. Cela peut être un problème non seulement pour des raisons de frugalité et d'économies d'énergie, mais aussi dans certains cas, tels que l'Internet of Things (IoT), où il n'y a pas de moyen évident de le faire dans des appareils limités.
L'utilisation des protocoles ou techniques de l'IETF destinés à l'IoT, comme l'illustre la figure 1.1(B) pour réduire l'impact de cette verbosité, est une stratégie viable pour progresser dans ce domaine. La première option consiste à utiliser : DNS over DTLS (DoDTLS)) [61], qui utilise User Datagram Protocol (UDP) et réduit ainsi la surcharge causée par la poignée de main TCP/CoAP [137] avec DTLS pour garantir la sécurité. Cependant, il s'agit toujours de TLS nécessitant de multiples messages d'échange pour définir la clé commune de chiffrement. Une option plus appropriée consiste à suivre la méthode proposée dans [82], où les auteurs suggèrent d'utiliser CoAP [137] au lieu de HTTP, avec OSCORE [132] au lieu de DTLS. Cela nous permet de conserver le paradigme REST pour améliorer de manière significative la latence en réduisant le nombre de RTT et la taille des paquets tout en atteignant le même niveau de sécurité que DoH. Techniquement, cela nécessite la capacité de traiter les messages DNS comme des données à transporter par un protocole de haut niveau, comme illustré dans [143, 25], où un format JSON est utilisé pour représenter les messages DNS.

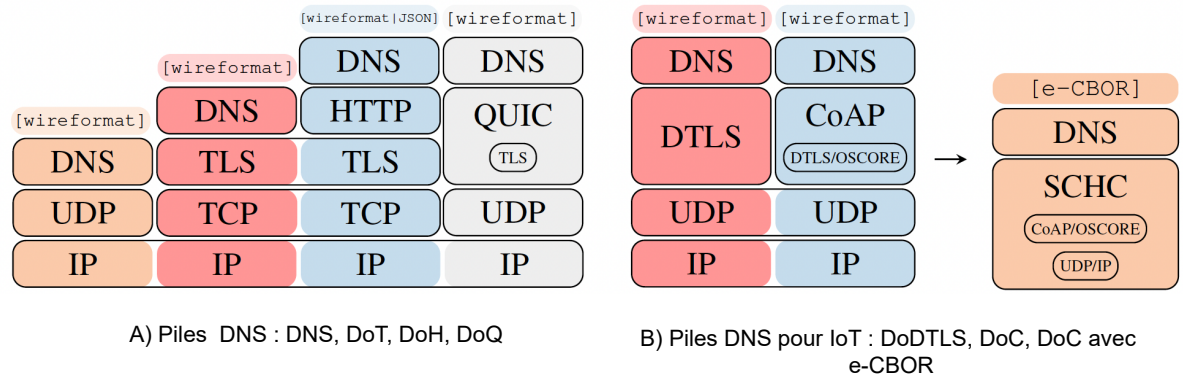


Figure 1.1: Pile protocolaire DNS

1.1.1 Questions de recherche

La problématique de recherche qui motive cette thèse porte sur l'utilisation des techniques développées dans l'IoT (Internet des objets), qui ont permis de réduire l'empreinte numérique, conduisant à de nouveaux protocoles tels que SCHC [100, 99] et CoAP/OSCORE/EDHOC [137, 132] pour réduire l'impact de la "RESTification" sur le DNS. Pour cela, les questions de recherche auxquels nous nous proposons d'apporter des réponses, sont les suivantes :

1. QR1 : *Format d'encodage des données DNS*
   Pour réduire l'impact de la RESTification sur le DNS, nous nous intéressons au format d'encodage des données et la question à laquelle nous allons répondre est, comment peut-on rendre le format de message DNS plus compact et flexible en utilisant Concise Binary Object Representation (CBOR) ? Nous savons que l'implémentation actuelle de DoH qui a ouvert la voie à la RESTification du DNS, utilise le formats JSON ou "wireformat"[139] pour transporter les messages DNS sur HTTPS. Ce qui augmente la taille des messages. Notre travail est de proposer un nouveau format plus compact et flexible que ces formats traditionnels.
2. QR2 : *Résolution DNS privé*
   Comment peut-on limiter l'accès à l'espace de noms de domaine grâce à une résolution DNS privée ? Nous montrons comment on peut optimiser l'architecture d'itinérance entre opérateurs de réseaux LoRaWAN [88] développée par LoRa Alliance grâce à cette résolution DNS privée. Ce deuxième objectif de la thèse est de montrer qu'un usage de la RESTification du DNS est prometteur.

1.2 Contributions et approches

Dans cette thèse, nos travaux sont présentés en trois parties, les deux dernières parties traitant chacune d'un problème important pour le DNS et apportant des réponses aux défis soulevés. La première partie définit les concepts nécessaires à la compréhension de cette thèse ainsi que les travaux existants. Quant à la deuxième partie, elle présente nos contributions sur le format d'encodage des données DNS et la méthode de résolution DNS privée; la troisième partie porte sur une solution de gestion de l'itinérance entre opérateurs LoRaWAN. Les domaines de recherche traités dans le cadre de notre travail sont résumés dans la figure 1.2.

1.2.1 e-CBOR: Nouveau format de message DNS plus flexible et compact basé sur CBOR

Nous avons, dans un premier temps, proposé un nouveau format reposant sur le format binaire de sérialisation CBOR pour encoder les messages DNS. Pour cela, nous introduisons une nouvelle représentation appelée efficient CBOR (e-CBOR) qui traduit les messages DNS en table CBOR afin de le rendre plus compact et flexible tout en gardant la promesse de sa RESTification possible dans le domaine de l'IoT.

1.2.2 DNS-Broker: Application à l'optimisation de l'architecture LoRaWAN pour l'itinérance entre les opérateurs IoT

La deuxième contribution porte sur la conception et l'implémentation d'une nouvelle entité appelée *DNS-Broker* reposant sur DoH. Elle est utilisée pour faire une résolution DNS privée afin de limiter l'accès à une ressource via des certificats et réduire le nombre de messages. Elle permet de stocker des données directement dans le serveur DNS et par conséquent de se passer d'encodage du serveur REST une fois la résolution effectuée.
Pour valider la pertinence de ces approches, nous avons réalisé un démonstrateur basé sur l'itinérance passive de LoRaWAN. Il montre comment optimiser l'architecture d'itinérance passive entre opérateurs IoT plus précisément les opérateurs LoRaWAN, permettant ainsi aux Objets connectés mobiles de passer de leur zone de couverture à une autre. Il est à noter que ce travail a été présenté devant la communauté de la LoRa Alliance et aux journées LPWAN de France.

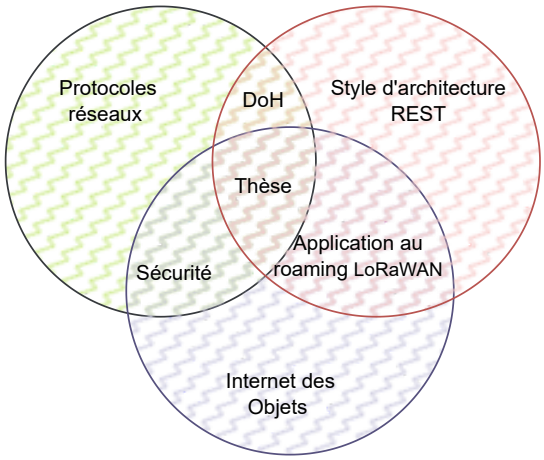


Figure 1.2: Domaines de recherche de cette thèse

1.3 Plan de la thèse

Ce manuscrit de thèse détaille les différentes contributions que nous avons présentées et s'organisent en huit chapitres regroupés en trois parties. Dans la première partie, après le présent chapitre consacré à l'introduction, le chapitre 2 présente de manière détaillée le DNS, l'état de l'art sur les travaux portant sur les applications du DNS à d'autres domaines et ceux relatifs à l'impact de la RESTification sur le DNS. Les différentes techniques d'encapsulation du DNS sur d'autres protocoles sont présentées ainsi que l'étude des formats de représentation des données DNS. Nous poursuivons au chapitre 3 par la présentation

du paradigme REST, son impact sur l'infrastructure DNS. Enfin le chapitre 4 analyse et compare les différents formats binaire d'encodage des messages, les plus populaires.

Dans la deuxième partie de cette thèse, nous présentons les principales contributions réalisées portant sur la réduction de l'impact de la RESTification sur le DNS grâce aux techniques de l'Internet des Objets ainsi que leurs implémentions. Au chapitre 5, nous rappelons les détails du format classique d'un message DNS et expliquons comment les données DNS sont représentées. Ensuite, nous décrivons le nouveau format e-CBOR proposé et sa mise en œuvre avec des exemples d'application. Au chapitre 6, nous présentons notre architecture DNS-Broker et la gestion des certificats SSL lors de la résolution.

Dans la troisième partie, nous proposons un démonstrateur qui nous permet d'appliquer nos contributions dans le domaine de la mobilité des appareils IoT. Pour cela, nous commençons au chapitre 7 par étudier des différentes architectures d'itinérance entre opérateurs IoT qui existent dans la littérature et les différents types d'itinérance IoT. Au chapitre 8, nous décrivons les différents modes d'activation d'un objet connecté en itinérance dans le réseau LoRaWAN. Nous poursuivons par l'optimisation et l'implémentation de l'architecture LoRaWAN qui facilite l'itinérance des objets connectés mobiles. Le chapitre 8 analyse les performances de notre solution et la compare à IoTRoam. Enfin, le chapitre 9 conclut la thèse et présente quelques perspectives.

Chapter 2

DNS : Domain Name System

It is more complicated than you think.

— **RFC 1925, Nr.8**
(The Twelve Networking Truths)

Sommaire

2.1 Introduction

Aujourd'hui, le système de noms de domaine (en anglais DNS: *Domaine Name System*) [104, 105, 102, 103] est d'une importance capitale pour le fonctionnement de l'Internet. Il est chargé de traduire les noms de domaine généralement plus facile à mémoriser en adresses IP permettant ainsi de simplifier grandement l'utilisation de l'internet. Le concept de la mise en correspondance des adresses IP avec les noms de domaine remonte à l'époque du *Advanced Research Projects Agency Network (ARPANET)* [56]. Un fichier HOST.TXT contenant pour chaque machine son nom et l'adresse IP associée, était utilisé. Chaque fois qu'un hôte voulait accéder à un service extérieur, il devait télécharger le fichier pour s'assurer qu'il disposait de la version la plus récente. Au fur et à mesure qu'Internet s'est développé, une telle solution non évolutive est devenue impraticable. C'est pourquoi, en 1983, Paul Mockapetris a proposé les Request For Comments (RFC) 882 [104] et 883 [105] contenant la première architecture d'un DNS décentralisé et qui a été finalement normalisée par l'IETF [125] dans les RFC 1034 [102] et 1035 [103].
Bien que le DNS soit très stable et présente de nombreux avantages pour l'Internet, ses fonctionnalités n'ont pas beaucoup évolué depuis 1983. De plus, toute tentative d'extension de ces fonctionnalités à d'autres usages possibles que la gestion de noms reste aujourd'hui délicate. Récemment dans le RFC 8484, les premières tentatives d'intégration du DNS dans le paradigme REST ont ouvert la voie à l'extension du DNS. Diverses propositions basées sur différents schémas d'encapsulation telles que ceux présentés à la Figure 1.1 ont conduit à une véritable "***RESTification***" du DNS.
Dans ce chapitre nous rappelons d'abord brièvement l'architecture du DNS en indiquant les propriétés fondamentales que sa conception permet de garantir. Nous montrons que le DNS est naturellement intégré dans la majorité des applications Internet et notamment celles massivement utilisés de nos jours. Nous abordons par la suite la caractéristique modulaire native du DNS qui a permis la prise en charge de nouveaux besoins par l'adjonction de nouveaux enregistrements ou regroupements de nouveaux enregistrements en nouveaux services. Nous abordons également les nouvelles approches d'encapsulation protocolaires qui tout en permettant au DNS d'hériter de nouvelles propriétés de sécurité, ont ouvert la voie à sa restification. Finalement nous présentons les travaux portant sur les applications du DNS à d'autres domaines que la traduction de noms.

2.2 Fondamentaux du DNS

Cette section qui rappelle le fonctionnement du DNS, n'a pas pour but de présenter ces concepts de manière détaillée et exhaustive, mais plutôt de préciser la base conceptuelle nécessaire à la compréhension de cette thèse.

2.2.1 Structure arborescente

Le système de noms de domaine [103] utilise une structure arborescente qui est constituée de deux éléments principaux : les domaines et les zones. Le domaine est considéré comme un sous-arbre de l'arbre DNS. Comme le montre la Figure 2.1, pour obtenir le nom de domaine, il faut concaténer le label de tous les nœuds en partant de la racine du sous-arbre jusqu'à la racine de l'arbre complet du DNS. Cette construction de l'arbre garantit l'unicité de nom dans la zone. Chaque domaine est composé d'une ou plusieurs zones, qui représentent les unités administratives de l'arbre DNS. Les serveurs qui gèrent ces zones sont responsables de toutes les informations qui y sont contenues. Les administrateurs de ces serveurs ont la responsabilité de maintenir à jour et de rendre disponibles les informations contenues dans leur zone respective.

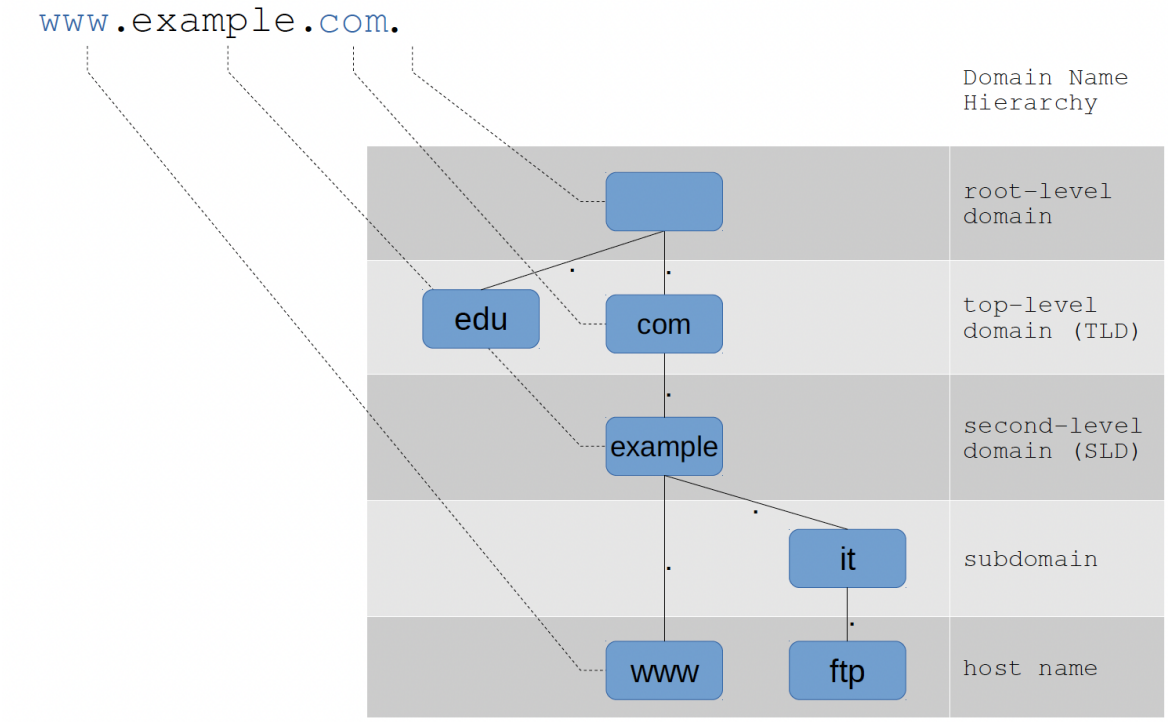


Figure 2.1: Vue d'ensemble de l'espace de noms de domaine.

Au sommet de cet arbre[1], on trouve le domaine racine qui ne compte que 13 serveurs de noms racine, dont chacun est nommé à l'aide de lettres de "A" à "M", déployés sur environ 500 sites à travers le monde. L'étiquette du domaine racine est en fait intégrée dans chaque requête DNS qui contient une étiquette vide invisible à la fin du nom de domaine actuel.

[1]L'arbre au milieu représente la structure hiérarchique et logique de l'espace de noms de domaine. Les noms de domaine illustrés sont : "edu", "www.example.com" et "ftp.it.example.com". La partie droite représente la hiérarchie des noms de domaine qui sépare les niveaux de l'arbre.

Par exemple, "www.example.com." indique l'emplacement exact et est appelé nom de domaine absolu ou, plus formellement, nom de domaine entièrement qualifié (FQDN).

2.2.2 Architecture

Le DNS est composé de trois éléments qui gèrent et utilisent les informations contenues dans le fichier zone. Il s'agit des serveurs de noms faisant autorité, des résolveurs locaux et des résolveurs récursifs [103].

1. *Serveurs de noms faisant autorité*: Ce sont des serveurs de noms chargés d'administrer une ou plusieurs zones DNS. Ils détiennent donc une copie autoritative des enregistrements DNS pour une zone de domaine spécifique et sont responsables de répondre aux requêtes pour ladite zone de domaine.
2. *Resolveurs Recursifs*: Les résolveurs récursifs, également appelés serveurs caches récursifs, ont largement contribué à la résilience du système DNS et par conséquent à son passage à l'échelle. Ces serveurs n'administrent aucune zone. Leur rôle est de recevoir des requêtes en provenance des résolveurs locaux et de leur répondre en utilisant les informations précédemment stockées en mémoire. S'ils ne trouvent pas de réponse, ces requêtes sont envoyées aux serveurs faisant autorité les plus proches et pouvant détenir la réponse. L'utilisation des résolveurs récursifs permet de réduire le temps de résolution.
3. *Resolveurs locaux*: Le résolveur local est chargé de transmettre la requête d'un client ou d'une application à un serveur faisant autorité via le résolveur récursif. Lorsque la réponse à cette requête arrive, il l'envoie au client.

2.2.3 Principe de la résolution de noms

La résolution de noms représente l'opération d'association d'un nom de domaine à un enregistrement de ressource ou *Resource Record (RR)*. Cette résolution peut être itérative ou récursive. La figure 2.2 illustre ces deux méthodes [103].
Dans la résolution itérative, le serveur ne répondra qu'avec les informations qu'il a en sa possession. S'il a les informations demandées, il les fournira, sinon il donnera des informations sur le serveur de noms le plus apte à donner la réponse. Pour trouver ce serveur, il examinera ses délégations et sélectionnera celle qui a le suffixe commun le plus long avec le nom demandé.
En revanche, dans une résolution récursive, le serveur de noms contactera tous les serveurs faisant autorité jusqu'à obtenir la réponse recherchée. En d'autres termes, il effectuera une résolution complète.
De manière générale, les serveurs faisant autorité sont configurés en mode itératif pour économiser du temps et des ressources, tandis que les resolveurs sont configurés en mode récursif pour obtenir un maximum d'informations afin de répondre rapidement

aux demandes. Sur la figure 2.2, le résolveur est en mode récursif parce qu'il passe par plusieurs serveurs faisant autorité tels que les serveurs Root et serveurs generic top level domain (gTLD) avant d'avoir la réponse à sa demande.

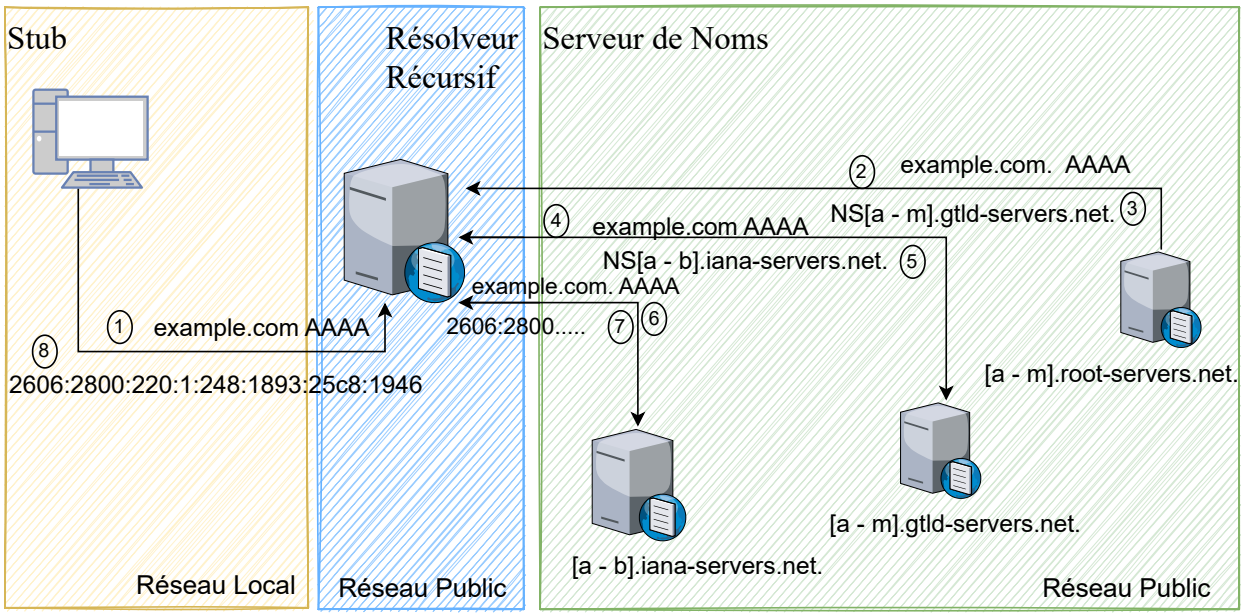


Figure 2.2: Processus de résolution DNS

2.3 Propriétés, applications et évolutions du DNS

Après avoir rappelé les principes fondamentaux du DNS, l'objectif de cette section est de montrer brièvement la richesse du DNS en faisant le point sur les principales propriétés garanties par le DNS, quelques exemples d'applications Internet massivement utilisées et s'appuyant sur le DNS, les capacités natives d'extensions horizontales du DNS pour faire face à de nouveaux besoins, les avancées en termes de sécurité protocolaire ainsi que les premières propositions d'application du DNS à de nouveaux usages.

2.3.1 Propriétés du DNS

2.3.1.1 Performance de la résolution de noms

Afin de garantir l'évolutivité de l'espace de nommage, DNS gère les noms de domaine de façon hiérarchique, le niveau supérieur déléguant les noms au niveau inférieur. La résolution d'un nom de domaine suit également cette relation de délégation entre le nom de niveau supérieur et le nom de niveau inférieur, on peut le voir sur la figure 2.2. Le client envoie une requête DNS à son résolveur récursif qui vérifie son cache pour voir s'il

est possible de répondre à la requête. On a deux cas possibles, si le cache ne contient pas la réponse (réponse négative), le résolveur envoie la demande du client aux serveurs faisant autorité, niveau par niveau, jusqu'à obtenir une réponse complète.
En supposant que le résolveur conserve chaque nouvelle réponse dans son cache et que le nom de résolution se trouve au moins sous le TLD en tant que nom de second niveau, on peut ainsi modéliser la latence [161, 158] d'une résolution DNS de la manière suivante:

$$L_{DNS} = p \times T_a + p \times (1-p) \times (T_a + T_b) \tag{2.1}$$

$$+ p \times \sum_{k=2}^{n-1} (1-p)^k \times (k \times T_b + T_a) \tag{2.2}$$

$$+ (1-p)^n \times (n \times T_b + T_a) \tag{2.3}$$

On note ici que p représente la probabilité qu'un résolveur DNS retourne une réponse positive au client, Ta représente le temps d'aller-retour (RTT) entre le client et le résolveur, et Tb représente le temps d'aller-retour (RTT) entre le résolveur et le serveur faisant autorité. Et n est la longueur (nombre de labels) du nom de domaine.

En se basant sur ce modèle mathématique, la figure 2.3 montre la latence moyenne pour une résolution DNS. Nous fixons $Ta = 30ms$ et $Tb = 40ms$, les trois courbes représentant les noms de domaine de longueur deux, trois et quatre. Comme on peut le constater, la latence augmente lorsque le résolveur récursif ne trouve pas la réponse dans son cache et lorsque la longueur du nom augmente [67, 158].

En concrétisant les très bonnes performances du DNS la figure 2.3 montre également que ***si l'on souhaite étendre ses fonctionnalités à d'autres usages, la latence devient un défi majeur***, surtout pour des applications ayant des exigences strictes en termes de délai. Cette figure se limite à la latence de résolution du protocole DNS de base, mais si l'on utilise IPv6 ou des protocoles comme DNSSEC ou DANE, davantage de procédures et d'échanges sont nécessaires impactant de fait les performances du DNS.

2.3.1.2 Fiabilité

En ce qui concerne la fiabilité, le DNS est soumis à une forte charge de trafic et doit être capable de répondre rapidement à des requêtes de résolution de noms de domaines. Des pannes du DNS peuvent entraîner des interruptions de service et des perturbations pour les utilisateurs. Pour garantir la fiabilité du DNS, des serveurs DNS redondants et des protocoles de récupération d'erreurs sont généralement mis en place [115].

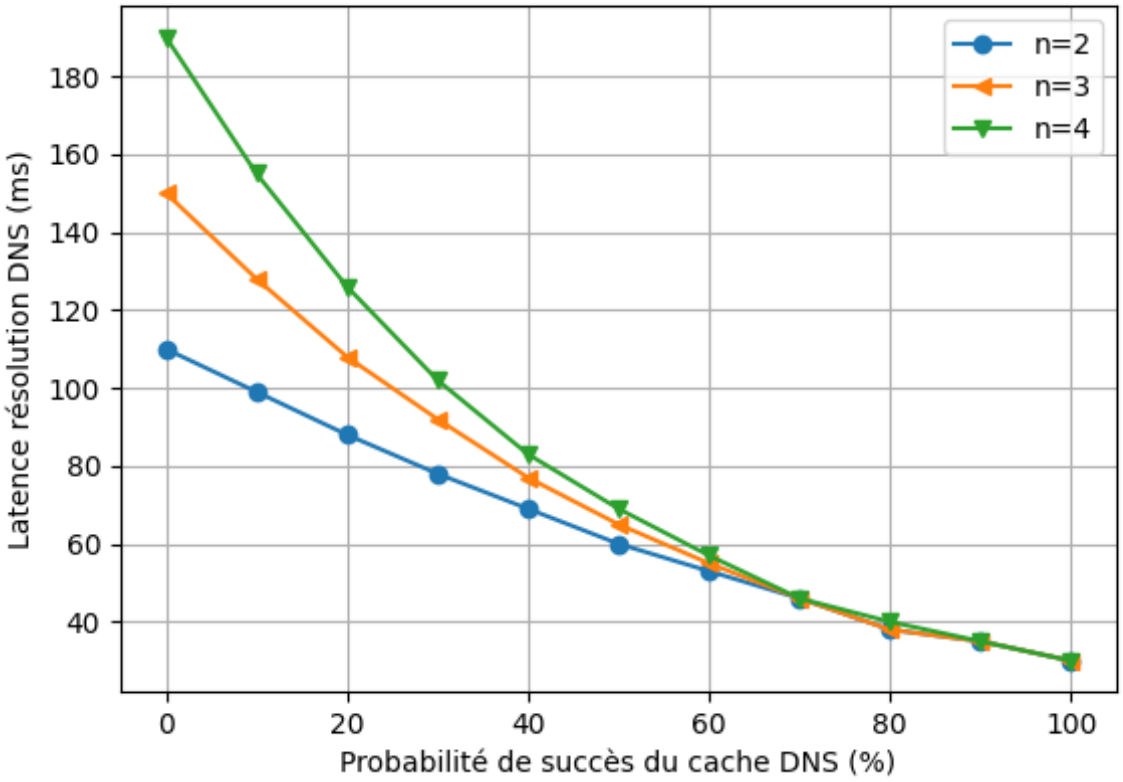


Figure 2.3: La latence de résolution du DNS

2.3.1.3 Intégrité

L'intégrité des données DNS garantit que les informations contenues dans le système de noms de domaine sont exactes et n'ont pas été altérées ou manipulées par des tiers non autorisés et/ou malveillants.

Si l'intégrité des données DNS est compromise, cela peut entraîner des perturbations du fonctionnement d'Internet et des risques pour la sécurité des utilisateurs. Cette intégrité du DNS peut être compromise par différentes formes d'attaques, notamment les attaques de redirection DNS, les attaques de modification DNS. Les attaques de redirection DNS impliquent la modification des enregistrements DNS pour rediriger les utilisateurs vers des sites Web malveillants, tandis que les attaques de modification DNS impliquent la modification des enregistrements DNS pour rediriger le trafic vers des serveurs malveillants. Pour protéger l'intégrité du DNS, des mesures de sécurité peuvent être mises en place, telles que l'utilisation de la validation DNSSEC (Domain Name System Security Extensions) [72].

Avec DNSSEC [72], l'intégrité des données DNS est assurée par l'utilisation de clés et de signatures numériques qui permettent aux résolveurs DNS de vérifier l'authenticité des

données reçues. Si les signatures numériques ne correspondent pas ou si les données ont été modifiées en transit, une erreur sera détectée et le résolveur DNS pourra prendre des mesures pour protéger l'utilisateur. Cependant, il est important de noter que DNSSEC ne fournit pas de confidentialité des données DNS. Les requêtes et les réponses DNS peuvent toujours être observées et lues par des parties tierces. Pour assurer la confidentialité des données DNS, des mécanismes supplémentaires tels que le chiffrement DNS (DNS over TLS ou DNS over HTTPS) doivent être utilisés.

2.3.2 Exemples d'applications Internet utilisant le DNS

La fonction de traduction offerte par le DNS a très rapidement été intégrée à des services internet pour usagers, comme par exemple le courrier électronique, pour faciliter un accès transparent, rapide et fiable à un ensemble décentralisé de ressources internet impliquées dans la mise en oeuvre de ce service. L'avènement du Web dans les années 1990 a amplifié cette approche pour faire du DNS un pilier incontournable de tous les grands services emblématiques de l'internet (cf. figure 2.4). Nous esquissons dans ce qui suit un rapide panorama de quelques uns de ces services.

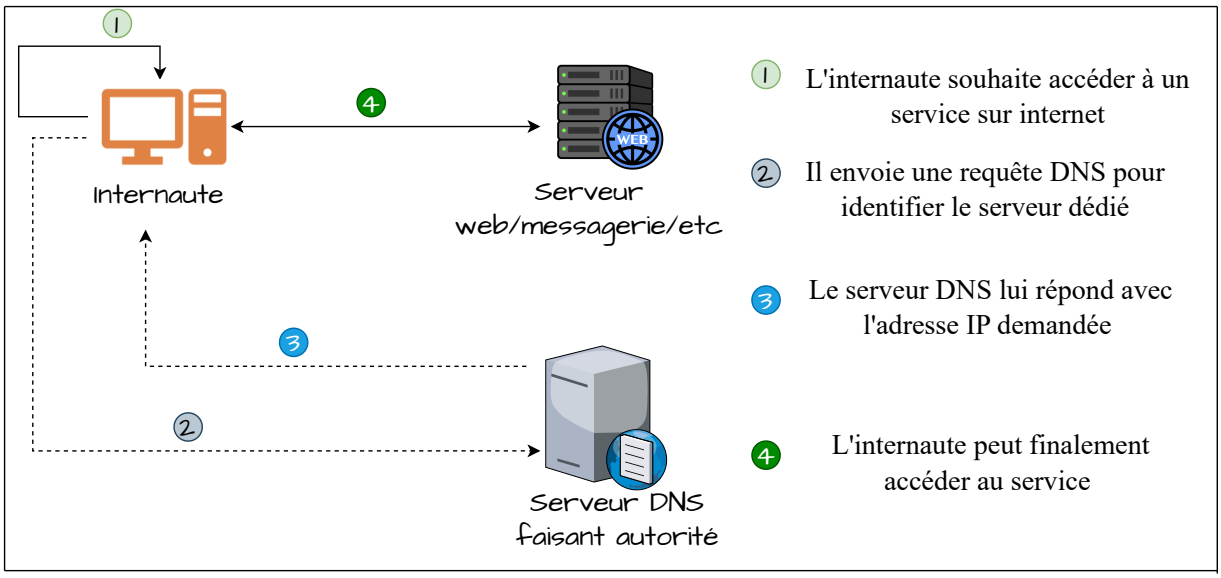


Figure 2.4: Interaction entre le DNS et les autres services (web, messagerie, etc)

2.3.2.1 E-mail

L'e-mail, tel que nous le connaissons aujourd'hui, a été développé dans les années 1960. À cette époque, le e-mail était utilisé pour envoyer des messages entre des ordinateurs connectés à un réseau, tel que le réseau ARPANET [56]. Les messages étaient envoyés

en utilisant des protocoles de messagerie propriétaires tels que SNDMSG [134] et MAILBOX [131]. Les adresses e-mail étaient généralement composées d'un nom d'utilisateur et d'un nom de domaine, similaires aux adresses e-mail actuelles. Au fil du temps, les protocoles de messagerie électronique ont été normalisés et standardisés, et de nombreux nouveaux protocoles ont été développés pour faciliter l'envoi et la réception de courriels. Parmi les protocoles les plus courants figurent Simple Mail Transfer Protocol (SMTP) [147] pour l'envoi de courriels, Post Office Protocol (POP) [147] et Internet Message Access Protocol (IMAP) [32] pour la récupération de courriels.

L'e-mail est un moyen de communication très populaire auprès du grand public. Cette popularité a été possible grâce à l'introduction dans le DNS de l'enregistrement Mail Exchange (MX) [102, 103] introduit en 1983. Cet enregistrement est utilisé pour spécifier le serveur de messagerie responsable de recevoir les e-mails destinés à un domaine particulier. Il contient le nom de domaine du serveur de messagerie et une valeur de priorité, qui permet de déterminer l'ordre de traitement des serveurs de messagerie en cas de défaillance du premier serveur. L'enregistrement MX a rapidement été adopté et publié dans le RFC 974 [124] intitulé "Mail Routing and the Domain System".

2.3.2.2 Streaming

Lorsque nous souhaitons accéder à un service de communication en temps réel ou de streaming en ligne, comme Netflix ou YouTube avec des technologies telles que WebRTC, HTTP Live Streaming (HLS) [90] et Dynamic Adaptive Streaming over HTTP (DASH) [73], notre appareil a besoin des adresses IP des serveurs qui les hébergent. Pendant aussi la diffusion en continu, le serveur envoie les contenus vidéos ou audio sous forme de paquets qui sont acheminés sur Internet jusqu'à notre appareil. Le DNS est utilisé à chaque étape du processus, depuis la résolution du nom de domaine jusqu'à l'acheminement des paquets de données. Les services de streaming utilisent souvent des réseaux de distribution de contenu (CDN) [69] pour améliorer la vitesse et la fiabilité du streaming en ligne. Les CDN sont des réseaux de serveurs distribués dans le monde entier, qui permettent de stocker le contenu vidéo en plusieurs endroits pour minimiser la distance et le temps de transmission des paquets de données. Le DNS est également utilisé pour résoudre les noms de domaine des serveurs CDN en adresses IP, ce qui permet de sélectionner le serveur CDN le plus proche pour fournir le contenu de manière plus efficace.

2.3.2.3 Messagerie instantanée

Comme tous des services Web ou applicatifs, la messagerie instantanée ne peut pas fonctionner sans le DNS. Un des protocoles les plus populaires en messagerie instantanée est Extensible Messaging and Presence Protocol (XMPP) [33]. C'est un protocole de

communication ouvert et extensible utilisé pour les services de messagerie instantanée, de présence en ligne, de collaboration en temps réel et d'autres applications de communication en temps réel. Il est basé sur eXtensible Markup Language (XML) [110] et suit une approche client-serveur. Il permet ainsi aux utilisateurs de communiquer entre eux en envoyant des messages texte, des fichiers et d'autres types de données en temps réel. Il prend également en charge la gestion de la présence, ce qui signifie que les utilisateurs peuvent indiquer leur disponibilité en ligne, leur statut et leurs activités actuelles. Une caractéristique importante de XMPP est sa décentralisation. Au lieu de passer par un seul serveur centralisé, chaque utilisateur peut se connecter à son propre serveur XMPP ou choisir parmi différents serveurs existants. Cela favorise l'interopérabilité et permet aux utilisateurs de communiquer avec d'autres utilisateurs sur différents serveurs XMPP. Cette décentralisation a été possible grâce au DNS, car il permet de localiser et de se connecter facilement aux serveurs appropriés pour établir les communications. Le DNS joue donc un rôle essentiel dans l'infrastructure sous-jacente de ce protocole.

2.3.3 Modularité structurelle du DNS

Dans sa conception initiale, le DNS était destiné à faire uniquement la traduction des noms de domaines en adresses IP. L'enregistrement RR en constituant la structure de données fondamentale du DNS, lui a cependant offert une modularité native qui a très simplement ouvert la voie à un principe d'extension horizontale. De nouveaux types d'enregistrement ont ainsi été introduits par la suite dans la spécification du DNS afin de répondre aux nouveaux besoins des utilisateurs d'Internet. Nous rappelons ci-dessous, par ordre d'apparition chronologique, certains des RRs les plus utilisés ainsi que de nouveaux services du DNS.

2.3.3.1 RR SRV: Service

L'enregistrement SRV [52] a été introduit dans les spécifications du DNS en 1999 dans le RFC 2782 et est utilisé pour spécifier les services disponibles dans un domaine. Il permet de définir les serveurs et les ports associés à un service particulier. Il est couramment utilisé pour la découverte de services tels que la localisation des serveurs de messagerie web (webmail) ou des services de voix sur IP (VoIP). Il fournit une méthode standardisée pour décrire l'emplacement des services réseaux dans le DNS, facilitant ainsi l'interopérabilité et la configuration des applications réseaux.

2.3.3.2 RR NAPTR : Name Authority Pointer

L'enregistrement NAPTR [94] a été introduit dans les spécifications du DNS en 1999 via la RFC 2915. Il a été développé pour faciliter la mise en place de services tels que la numérotation téléphonique, la découverte de services et d'autres applications où une transformation du nom de domaine est nécessaire.

2.3.3.3 RR TXT: Text

L'enregistrement TXT [128] est décrit dans le RCF 1464, publié en mai 1993. Il permet de stocker des informations de texte arbitraires associées à un nom de domaine. Il est couramment utilisé pour diverses fins, notamment la vérification de domaine, les enregistrements SPF (Sender Policy Framework) pour la validation des e-mails, les clés de sécurité, les informations de géolocalisation, et d'autres données textuelles personnalisées. Depuis son introduction, l'enregistrement TXT est largement utilisé pour stocker des informations supplémentaires et des attributs spécifiques à un domaine. Il fournit une flexibilité pour ajouter des données textuelles arbitraires dans le DNS afin de répondre à divers besoins d'information et de configuration.

2.3.3.4 SPF: Sender Policy Framework

Le SPF [70] n'est pas un enregistrement mais un service qui a été proposé en 2003 par Meng Weng Wong et a été largement adopté par la suite. Il permet aux propriétaires de domaines de spécifier quels serveurs de messagerie sont autorisés à envoyer des e-mails au nom de leur domaine. Cela aide à lutter contre le spam et l'usurpation d'identité en fournissant une méthode de validation de l'expéditeur des e-mails. Pour mettre en œuvre SPF, les propriétaires de domaines ajoutent un enregistrement TXT spécial dans le DNS de leur domaine, qui contient des informations SPF [70] spécifiques. Cet enregistrement TXT [128] indique les serveurs de messagerie autorisés à envoyer des e-mails au nom du domaine. Les serveurs de messagerie destinataires peuvent ensuite vérifier ces enregistrements SPF pour valider l'authenticité de l'expéditeur. Ainsi, bien que SPF ne soit pas un enregistrement distinct, il utilise le mécanisme des enregistrements TXT du DNS pour spécifier les politiques de validation d'e-mails.

2.3.3.5 DNSSEC: Domain Name System Security Extensions

La version actuelle de DNSSEC [72] est décrite dans les RFC 4033 [15], RFC 4034 [16] et RFC 4035 [17], publiées en mars 2005. Ces RFCs décrivent les détails techniques de DNSSEC, y compris les nouveaux types d'enregistrements tels que les resource records SIG (Signature), KEY (Clé publique) et NXT (Next Secure). C'est une extension du DNS qui vise à renforcer la sécurité et l'intégrité des informations de résolution DNS. Il

utilise des mécanismes de cryptographie pour signer numériquement les données DNS, permettant ainsi aux clients de vérifier l'authenticité et l'intégrité des réponses DNS reçues. Avec DNSSEC, les enregistrements DNS sont signés à l'aide de clés cryptographiques. Les serveurs DNS faisant autorité signent les enregistrements avec une clé privée, et les clients peuvent vérifier ces signatures en utilisant la clé publique correspondante. Cela permet de détecter toute altération des données DNS en transit ou sur les serveurs. DNSSEC offre ainsi plusieurs avantages en matière de sécurité DNS. Il permet de prévenir les attaques telles que le DNS spoofing, le cache poisoning et les attaques de redirection de trafic. Il renforce également la confiance dans les réponses DNS en fournissant une vérification cryptographique de l'authenticité des données.

2.3.3.6 DMARC: Message Authentication

DMARC [74] est un service comme SPF qui utilise le mécanisme des enregistrements TXT (Text) existants du DNS pour spécifier les politiques de gestion des e-mails. Il a été introduit en 2012 par l'Internet Engineering Task Force (IETF) pour améliorer l'authentification des e-mails et lutter contre le phishing, le spam et l'usurpation d'identité. Il permet aux propriétaires de domaines de définir des politiques de contrôle des e-mails envoyés au nom de leur domaine. Pour mettre en œuvre DMARC, les propriétaires de domaines ajoutent un enregistrement TXT [128] spécial dans le DNS de leur domaine, qui contient les informations DMARC spécifiques. Cet enregistrement TXT définit les politiques de gestion des e-mails, y compris les actions à prendre pour les e-mails non conformes, les rapports à générer et les adresses de contact à utiliser. Les enregistrements DMARC dans le DNS aident les serveurs de messagerie destinataires à évaluer l'authenticité des e-mails en vérifiant les politiques DMARC spécifiées par le domaine émetteur.
Ces enregistrements ont été développés pour répondre aux besoins spécifiques des services Web et améliorer la gestion des e-mails, de la résolution DNS et des autres fonctionnalités associées au Web.

2.3.4 Sécurisations du DNS par encapsulations protocolaires

Avec une forte expérience dans la conception et le développement des protocoles et technologies standardisées pour l'Internet, l'IETF a mis en place des nombreux groupes de travail (voir Figure 2.5) relatifs à la standardisation des techniques d'intégration du DNS à d'autres protocoles (TLS, DTLS), parmi lesquelles les plus significatives sont listées ci-dessous. Ces travaux prennent également en compte la sécurisation du canal de communication via des protocoles de sécurité tels que TLS et DTLS.
Les encapsulations du DNS dans des protocoles ayant des caractéristiques fortes en termes de sécurité, permettent au DNS une sécurisation par héritage. Cette section présente

les principaux standards ayant contribuer à étendre les propriétés de sécurité du DNS ouvrant par ailleurs la voie à sa RESTification.

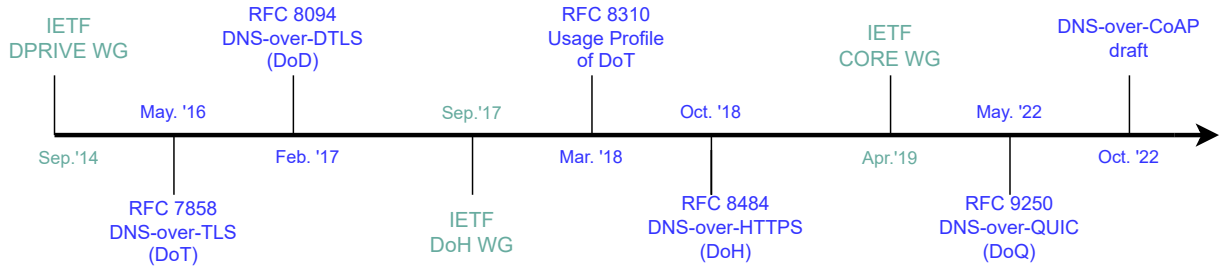


Figure 2.5: Chronologie de standardisation des différentes techniques de transport du DNS sur d'autres protocoles

2.3.4.1 DNS over TLS (DoT)

Dans le RFC 7858 normalisé par IETF en 2016 [60], afin de résoudre des problèmes du DNS portant sur la confidentialité et l'intégrité de ses communications, les chercheurs ont proposé le protocole DoT. Il permet d'établir une session TLS sur une connexion TCP via le port 853. Son but est de s'assurer que les messages DNS sont transmis sur un canal sécurisé afin de réduire l'écoute et la divulgation des données. Depuis sa normalisation, DoT a reçu un très large soutien en ayant par exemple été adopté sur les appareils Android en tant que " DNS privé " depuis Android 9 (août 2018) [106]. De même, Apple l'a intégré sur ses appareils et services, en particulier sur IOS 14 en septembre 2020 et sur macOS Big Sur en novembre 2020 [135]. Grâce à ce RFC, la communauté scientifique a compris qu'il était désormais possible d'étendre les fonctionnalités du DNS en le transportant sur un autre protocole. Malheureusement DoT s'appuie sur TCP et utilise le port 853 qui fait constamment l'objet de blocage par certains pare-feux.

2.3.4.2 DNS over DTLS (DoTLS)

Pour contourner l'utilisation de TCP évoquée précédemment dans le cas de DoT, des chercheurs de CISCO ont proposé le protocole DoDTLS [61] en 2017, qui a été normalisé dans le RFC 8084. Il utilise UDP afin de minimiser la latence durant une résolution. La section 3 de sa spécification stipule que les clients et les serveurs DoDTLS utilisent le port 853 pour communiquer, tout comme DoT. Avant d'envoyer une requête DNS, le client peut déterminer si le serveur prend en charge le DNS over DTLS en envoyant un message DTLS ClientHello à ce port. Dans ce cas, nous avons les scénarios suivants :

1. Si le serveur répond et qu'une session DTLS est établie, le client DoDTLS authentifie le serveur en utilisant les mêmes méthodes que TLS, en suivant les meilleures pratiques de sécurité TLS décrites dans le RFC 7525 [136]et les profils d'authentification DoT décrits dans le RFC 8310 [122]. Une fois que tout cela est fait, les requêtes et les réponses DNS sont maintenant protégées contre toute attaque malveillante,
2. Si le serveur ne répond pas, le client peut réessayer ou abandonner en utilisant DTLS. Et s'il abandonne pour revenir au DNS habituel, il doit changer le port 853 car il est interdit d'utiliser ce port pour transmettre des messages DNS en clair.

A noter que l'utilisation de DTLS peut entraîner une fragmentation des paquets DNS, car les paquets DTLS sont souvent plus grands que les paquets DNS. Les problèmes de fragmentation peuvent aussi entraîner des problèmes de performances, de sécurité et de compatibilité avec les équipements réseaux.

2.3.4.3 DNS over QUIC (DoQ)

Les utilisateurs ont de plus en plus besoin d'une connectivité Internet ininterrompue pour accéder parfois de manière simultanée à plusieurs ressources sur le web. Le protocole QUIC a été proposé pour répondre à ces besoins. Comme TCP, le protocole QUIC est un protocole de couche transport du modèle ISO, mais construit sur UDP. Comme son nom l'indique, il permet de transmettre de façon plus rapide des paquets de données entre client et serveur ou entre serveurs. Cela est dû au fait que QUIC n'a pas besoin d'un protocole de niveau supérieur tel que TLS pour fournir les fonctionnalités de sécurité comme le chiffrement et l'authentification lors des échanges. Contrairement à ses prédécesseurs, qui effectuent une poignée de main en deux aller-retours (établissement d'une connexion TCP et chiffrement TLS), QUIC le fait en un seul aller-retour.

Le transport de DoQ a été normalisé en mai 2022 par IETF dans le RFC 9250 [31]. L'objectif de ce RFC est d'exploiter les avantages qu'offre QUIC, pour transporter les requêtes DNS de manière sécurisée et rapide. DoQ vient ainsi apporter une solution au problème inhérent au rythme de vie extrêmement rapide. Par exemple, on se déplace constamment et en étant connecté à l'Internet : le matin, un utilisateur peut se connecter avec son téléphone à la box de la maison pour suivre l'actualité, une fois qu'il quitte la maison, il passe du Wi-Fi à la Fifth Generation Mobile Radio (5G) pour se connecter parfois à un site web, et une fois au bureau, son smartphone doit se connecter au WI-FI du bureau pour lire ses mails. Aujourd'hui, les normes telles que DoT, DoDTLS et DoH ont du mal à gérer ce passage d'une adresse IP à une autre, un évènement qu'on appelle "migration de connexion" mais DoQ serait capable de le faire.

Nous avons vu dans la section précédente que DoH présentait plusieurs limites dues à sa dépendance à HyperText Transfer Protocol (HTTP). Ces limites peuvent être améliorées grâce aux standards tels que HTTP/3 et DoQ parce qu'ils utilisent le protocole de

transport QUIC, qui est conçu pour améliorer la performance et l'efficacité des communications sur Internet.

Bien que DoQ et HTTP/3 ont été standardisés récemment par l'IETF et présentent de nombreux avantages, leur adoption reste encore très limitée car ce sont des protocoles relativement nouveaux. Les fournisseurs de services DNS doivent donc les mettre en oeuvre sur leurs serveurs pour que les utilisateurs puissent en bénéficier. De plus, DoQ ne suit pas le style d'architecture RESTful car il n'utilise pas les méthodes HTTP pour accéder à une ressource.
Nous présentons dans le tableau 2.1 une analyse comparative et synthétique des différentes méthodes d'encapsulation des messages DNS.

2.3.5 Le DNS comme outil de stockage d'informations critiques

Dans les Section 2.3.3 et Section 2.3.4, nous avons présenté de manière chronologique quelques enregistrements et services qui ont été ajoutés au DNS depuis sa mise en oeuvre en 1983. Nous pouvons voir que la fonction principale du DNS reste la traduction. Bien que d'autres enregistrements tels que MX [103], SRV [52], NAPTR [94] et TXT [128] ont été introduits sans changer son architecture, le DNS demeure un annuaire de l'Internet dont le rôle essentiel est d'associer le nom de domaine à un enregistrement spécifique. Dans cette section, nous nous intéressons aux approches qui utilisent le DNS pour stocker d'autres informations que celles liées à l'annuaire. Il s'agit à notre connaissance de DANE (DNS-based Authentification of Named Entities) et DKIM (Domain Keys Identified Mail) [9].

2.3.5.1 DANE: DNS-based Authentification of Named Entities

Depuis de nombreuses années [55], nos échanges sur internet sont sécurisés par une infrastructure PKI. Cependant, ce système repose sur des autorités de certification (AC) chargées d'émettre et de maintenir l'intégrité de ces certificats. Malheureusement, ce modèle basé sur les AC présente des défauts majeurs. En effet, les navigateurs web intègrent une liste de plusieurs dizaines d'AC de confiance, allant de grandes entités telles que DigiCert [140] et VeriSign [150] à des organismes moins connus comme Tubitak (gouvernement turc) [145].

La confiance accordée à une seule AC racine expose le système tout entier à des risques. En effet, n'importe quelle AC peut émettre des certificats pour n'importe quel nom de domaine, y compris des certificats frauduleux pour des sites Web légitimes. Malgré cela, les navigateurs ne signaleront pas ces certificats illégitimes, cela offre aux attaquants la possibilité de subvertir la sécurité de nombreux sites Web. Au cours des dernières années, nous avons été témoins de tentatives malveillantes de compromission de certificats pour des noms de domaine populaires tels que google.com ou microsoft.com. Ces attaques visaient surtout à espionner ou à falsifier des sites Web protégés par le protocole TLS.

DANE offre la possibilité aux clients de rechercher une deuxième source de vérification. C'est un protocole de sécurité qui s'appuie sur le DNS pour renforcer la sécurité des communications, notamment les connexions Transport Layer Security (TLS). Il utilise ainsi le DNS pour *stocker et vérifier les certificats numériques* utilisés dans les connexions sécurisées.

Avec DANE, le propriétaire du domaine va tout d'abord générer un certificat numérique pour le service (par exemple, HTTPS) qu'il souhaite sécuriser. Ce certificat peut être auto-signé ou émis par une AC. Une fois le certificat créé, un enregistrement TLSA est ajoutée à la zone DNS du domaine correspondant. Il contient des informations sur le certificat, telles que l'algorithme de hachage, le type de clé publique et d'autres informations. Ainsi, lorsqu'un client souhaite établir une connexion sécurisée avec le domaine, il effectue premièrement une requête DNS pour récupérer l'enregistrement TLSA correspondant. Le client spécifie, dans ce cas, le type de service et le nom de domaine dans la requête DNS. Après l'obtention de la réponse DNS, le client utilise les informations de l'enregistrement TLSA pour valider le certificat reçu du serveur web en comparant l'empreinte digitale du certificat avec celle fournie dans l'enregistrement TLSA pour vérifier l'authenticité du certificat.

2.3.5.2 DKIM: Domain Keys Identified Mail

DKIM [9] est une méthode d'authentification des e-mails qui utilise le DNS pour stocker et récupérer les informations cryptographiques nécessaires. Le DNS joue un rôle crucial dans DKIM en stockant les clés publiques et d'autres enregistrements associés. Son mode de fonctionnement est le suivant :

1. L'expéditeur génère une paire de clés cryptographiques : une clé privée et une clé publique. La clé privée est gardée secrète et utilisée pour signer les e-mails sortants, tandis que la clé publique correspondante est publiée dans le DNS.
2. La clé publique est généralement stockée dans un enregistrement DNS TXT associé au domaine de l'expéditeur. Cet enregistrement TXT contient la clé publique DKIM, ainsi que des informations supplémentaires telles que le sélecteur DKIM et l'algorithme cryptographique utilisé.
3. Lorsqu'un e-mail est envoyé, le serveur de messagerie de l'expéditeur ajoute une signature DKIM à l'en-tête de l'e-mail. Cette signature est créée à l'aide de la clé privée et inclut un hachage du contenu de l'e-mail, parmi d'autres informations.
4. Dès que le serveur de messagerie du destinataire reçoit l'e-mail, il récupère la clé publique de l'expéditeur à partir du DNS en utilisant le sélecteur DKIM et le nom de domaine. Le serveur de messagerie utilise ensuite la clé publique pour vérifier la signature DKIM. Il recalcule le hachage du contenu de l'e-mail et le compare à la signature décryptée. S'ils correspondent, l'e-mail est considéré comme authentique.

En stockant la clé publique DKIM dans le DNS, tout utilisateur qui souhaite accéder à un enregistrements DNS doit premièrement s'authentifier. Cela permet aux destinataires des e-mails de vérifier l'authenticité des e-mails signés par DKIM, qu'ils reçoivent.

2.3.6 Le DNS comme annuaire de l'IoT

Plusieurs chercheurs ont utilisé l'architecture DNS pour résoudre un certains nombres de problèmes dans le domaine de l'Internet des Objets. Bien que notre problématique porte sur la réduction de l'impact de la RESTification sur le DNS en utilisant les concepts de l'IoT afin de l'étendre à des domaines variés, nous avons pensé judicieux d'aborder ces différents travaux dans cette section, tout en notant que cette liste n'est pas exhaustive.

Pour Zhiwei Yan et al [159], la vision de l'internet des objets couvre la connectivité omniprésente d'objets dans le monde physique et les informations correspondantes dans le monde virtuel. Avec ses nombreux domaines d'application tels que les villes intelligentes, les transports urbains intelligents et les installations optimisées d'élimination des déchets, on a besoin d'un nom unique pour chaque objet et une résolution de nom sécurisée et efficace qui donnera accès à toutes les informations importantes sur tout objet grâce à son nom. Pour cela, les auteurs ont choisi d'utiliser le DNS parmi des systèmes de nommages existants. Sachant que le DNS, malgré ses fonctionnalités et ses performances, n'a pas été conçu à l'origine pour les applications IoT, la question que les auteurs se sont posés dans leurs travaux est la suivante : le DNS actuel peut-il fournir un service de noms adéquat pour l'IoT à l'avenir ? Ils ont ainsi analysé les forces et les faiblesses du DNS lorsqu'il est utilisé pour gérer l'IoT. Cette analyse se focalise sur cinq caractéristiques, à savoir la **S**écurité, la **M**obilité, l'**I**ndépendance vis-à-vis de l'infrastructure, la **L**ocalisation et l'**E**fficacité, qu'ils ont appelé collectivement SMILE.

Roberto Perdisci et al ont proposé IoTFinder [116], qui est un cadre efficace utilisé pour l'identification des appareils IoT. Les informations DNS passives dispersées sont collectées par IoTFinder et sont utilisées pour mettre en œuvre un cadre d'approche basé sur l'apprentissage automatique. Ce cadre vise à distinguer avec précision les différents types d'appareils IoT, en se basant exclusivement sur leurs noms de domaine. Le cadre est autonome et fonctionne indépendamment du fait que les appareils IoT se basent sur une traduction d'adresse réseau (NAT) ou d'autres dispositifs intermédiaires. Il est également indépendant de l'adresse IP attribuée, qu'il s'agisse d'une adresse IPv4 ou IPv6. Le cadre peut être utilisé selon différents cas d'utilisation. Un ensemble de données sur le trafic DNS et IoT a été collecté à des fins d'analyse et de détection [116].

Lee et al [79] ont développé une architecture pour les services de nommage DNS pour les dispositifs IoT. Cette architecture, baptisée DomainName System Name Autoconfiguration (DNSNA), a été mis en œuvre pour les réseaux IPv4 et IPv6. Le nombre d'appareils IoT augmente rapidement d'année en année et il est très difficile d'attribuer manuellement des noms DNS à ces appareils. Ces fonctionnalités génèrent automatiquement des noms DNS pour les appareils IoT. Il permet également la découverte de services et d'autres appareils sur le réseau. Le protocole de configuration dynamique de l'hôte (DHCP) est utilisé pour les appareils IPv4, tandis que le protocole de découverte du voisin (ND) est

utilisé pour les appareils IPv6. Le suffixe DNS des deux protocoles permet de déterminer le nom DNS. Le téléphone d'un utilisateur peut automatiquement authentifier un appareil IoT à l'aide d'un service sans contact utilisant la communication en champ proche (NFC). Le DNSNA réduit le volume global des paquets transmis.

L'auteur de [68] propose une architecture DNS pour l'Internet des objets. Le DNS est utilisé pour faire la traduction des identifiants uniques (URI) d'objets physiques en adresse réseaux concrètes, à partir desquelles des informations telles que leur état, leur emplacement sur ces objets peuvent être extraites. L'approche a été appelée dans le domaine de la logistique des transports et les résultats préliminaires indiquent que le DNS pourrait être une proposition faisable pour réaliser le suivi des objets connectés dans le domaine du transport.

2.4 Limitations de la RESTification du DNS

La RESTification du DNS introduit par DoH a ouvert la voie à l'extension future du DNS à d'autres usages. Il s'agit d'un ensemble de mécanismes mis en œuvre pour transporter les messages DNS sur le protocole HTTP lors de la résolution de noms. Cette RESTification du DNS soulève plusieurs problèmes. Premièrement, le DNS utilise actuellement un protocole basé sur UDP, qui est optimisé pour les transactions rapides et légères, avec peu de données échangées entre les serveurs et les clients. Cependant, sa RESTification requiert l'utilisation du protocole HTTP s'exécutant sur TCP, qui est beaucoup plus lourd, consomme beaucoup de ressources et génère des échanges de données plus importants. Cela peut entraîner une augmentation significative de la latence et une diminution des performances du DNS, en particulier pour les requêtes multiples.

Du fait de toutes ces limitations, quelques travaux évoqués précédemment ont été proposé par l'IETF dans la littérature portant sur la RESTification du DNS. Ces travaux présentent les différentes techniques d'encapsulation protocolaires du DNS. Les méthodes les plus connues sont: DoH, DoQ, DoC. Ils proposent aussi les différents formats de représentation des messages DNS. Il est clair que ces différentes approches ont apporté des améliorations significatives au DNS, elles n'ont cependant pas abordé les problèmes de latences.

2.5 Conclusion

En résumé, ce chapitre a établi les principes fondamentaux du DNS, nous permettant ainsi de saisir son importance, de comprendre son mécanisme de fonctionnement, ainsi que ses multiples applications et extensions. À travers ces notions, nous avons pu constater à quel point le DNS est essentiel au bon déroulement de nombreuses activités en ligne, telles que l'accès aux sites web, la livraison des e-mails, la téléphonie, la messagerie

Table 2.1: Comparaison des techniques d'encapsulation des messages DNS

Catégories	Noms	DoT	DoD	DoH	DoQ	DoC
DNS public	Google	✓	-	✓	✓	-
	Cloudfare	✓	-	✓	✓	-
	Quad9	✓	-	✓	-	-
	OpenDNS	-	-	✓	-	-
	Yandex.DNS	-		✓	-	-
	adGuard	✓	-	✓	✓	-
	Alternate DNS	-	-	✓	-	-
	OpenNIC	-	-	✓	-	-
	Tenta	✓	-	✓	-	-
	PowerDNS	✓	-	✓	-	-
	Dyn	-	-	✓	-	-
	Verisign	✓	-	✓	-	-
	SafeDNS	✓	-	✓	-	-
	CleanBrowsing	✓	-	✓	-	-
Serveurs DNS	Unbound	✓	✓	✓	✓	-
	BIND	✓	-	✓	-	-
	Knot Res	✓	-	✓	-	-
	dnsdist	✓	-	✓	-	-
	CoreDNS	✓	-	✓	-	-
	Cisco Registrar	✓	-	✓	-	-
	MS DNS	✓	-	✓	-	-
	AnswerX	✓	-	✓	-	-
Clients DNS	Ldns(drill)	-	-	✓	-	-
	Stubby	✓	-	✓	-	-
	BIND(dig)	-	-	✓	-	-
	Go DNS	-	-	✓	-	-
	Knot(kdig)	✓	-	✓	-	-
Navigateurs	Firefox	✓	-	✓	-	-
	Chrome	✓	-	✓	-	-
	IE	-	-	-	-	-
	Safari	-	-	✓	-	-
	Yandex	-	-	✓	-	-
	Tenta	-	-	✓	-	-
Systèmes d'exploitation	Androïd	✓	-	✓	-	-
	Linux (systemd)	✓	-	✓	-	-
	Windows	✓	-	✓	-	-
	macOS	✓	-	✓	-	-

instantanée, et bien d'autres fonctions cruciales. Son rôle est donc vital au sein de l'infrastructure Internet, et une compréhension approfondie de son fonctionnement nous permet d'explorer certaines extensions avancées.
Dans le chapitre suivant, nous nous penchons sur le rôle de l'architecture REST au sein de l'infrastructure DNS, en analysant son impact sur ses performances. Nous présentons également certains protocoles du domaine de l'IoT basés sur le modèle de communication REST, qui s'avéreront pertinents pour notre étude.

Chapter 3

Application du paradigme REST au DNS

Every old idea will be proposed again with a different name and a different presentation, regardless of whether it works.

— **RFC 1925, Nr. 11**
(The Twelve Networking Truths)

Sommaire

3.1 Introduction

Aujourd'hui, le paradigme Representational State Transfer (REST) est largement utilisé dans la conception d'API Web et offre un ensemble de principes et de conventions pour créer des services Web évolutifs et faciles à utiliser. L'application de ces principes au DNS peut le rendre compatible avec d'autres services Web reposant sur le paradigme REST.

Dans ce chapitre, nous étudierons le rôle et l'influence de l'architecture REST sur le DNS. Nous examinerons également la manière dont les méthodes HTTP, telles que GET, POST, PUT et DELETE, peuvent être employées pour interagir avec les ressources DNS. Nous explorerons les travaux de recherche qui ont introduit les principes de REST dans le DNS, tout en abordant leurs limitations. Enfin, nous présenterons les protocoles IoT qui utilisent le modèle REST pour échanger des données, en évaluant comment ils pourraient être utilisés pour atténuer l'impact potentiellement négatif du paradigme REST sur les performances du DNS.

3.2 Style architectural REST

Au cours de ces dernières années, le style d'architecture REST [156] a beaucoup influencé la conception et le développement de l'architecture web actuelle. Pour mieux comprendre le fonctionnement de REST, il est nécessaire de lire le chapitre 5 de la thèse de doctorat du Dr. Roy Fielding [43], son concepteur. Dans cette thèse, il définit le paradigme REST comme un "ensemble coordonné de contraintes architecturales qui restreignent les rôles/caractéristiques des éléments architecturaux et les relations autorisées entre ces éléments au sein de toute architecture conforme à ce style". On voit bien qu'il présente REST, non pas comme une technologie à part entière, mais comme un ensemble de contraintes à respecter lors de la conception des services Web:

- *Client-serveur*: Il faut strictement séparer le client et le serveur, ce qui leurs permettra d'évoluer de manière indépendante.
- *Sans état*: La communication entre le client et le serveur doit être sans état. Seul le client peut conserver les informations échangées avec le serveur lors d'une session, toutes les requêtes envoyées au serveur par le client doivent être indépendantes. Le serveur ne doit converser aucun état.
- *Mise en cache*: Le serveur a la possibilité d'utiliser une réponse antérieure mise en cache pour une durée définie pour répondre à une nouvelle requête du client.
- *Interface uniforme*: L'interface uniforme facilite la conception d'API REST en permettant aux clients et aux serveurs de communiquer de manière cohérente et standardisée, indépendamment des langages de programmation, des plates-formes et des architectures sous-jacentes. Cette cohérence et cette standardisation permettent également de rendre les systèmes plus évolutifs et interopérables, car les modifications

apportées à un côté de l'interface peuvent être gérées de manière plus transparente par l'autre côté.
- *Système hiérarchisé*: Une ressource peut être divisée en sous-ressources. Ainsi, l'accès à une ressource ne donne pas l'accès à tout son contenu, mais juste à une partie, le reste étant accessible par un URL. Un client peut accéder à la ressource qu'il recherche sans nécessairement connaître le serveur HTTP qui la détient.
- *Code de demande*: Il est possible pour un client d'étendre ses fonctionnalités en fonction de ses besoins.

L'architecture RESTful est basée sur des protocoles web standards tels que HTTP, ce qui la rend facile à comprendre et à utiliser. Elle utilise des verbes HTTP tels que GET, POST, PUT, DELETE pour représenter des opérations sur des ressources, ce qui facilite de fait l'accès aux ressources du web. Elle peut être facilement étendue pour gérer de grandes quantités de trafic en ajoutant simplement des composants supplémentaires. L'architecture RESTful utilise les fonctionnalités standards de sécurité du protocole HTTP basé sur SSL/TLS pour assurer la sécurité des données. Afin de mieux représenter les données transférées sur le réseau, REST prend en charge plusieurs types de formats de données tels que JSON et XML.

Une définition plus concrète de REST a été proposée par Richardson et Ruby [126]. Les auteurs proposent une architecture qui met en œuvre le style d'architecture REST reposant sur l'architecture orientée resources (en anglais : Resource-Oriented Architecture (ROA)). Fondamentalement, ROA est une mise en œuvre de REST et utilise donc les URL, HTTP et XML. Selon eux, l'identification d'une ressource doit se fait à partir des Universal Resource Identifiers (URI). L'URI représente le nom et l'adresse donnés à une Ressource. Une information qui n'a pas d'URI, n'est pas une ressource et on ne peut pas l'obtenir directement sur le web, car c'est l'URI qui fournit une adresse unique pour l'identifier. Il est alors important que les URI aient une correspondance claire avec les ressources qu'ils identifient. Dans ROA, les méthodes fondamentales HTTP sont ainsi utilisées pour les opérations les plus courantes afin de répondre aux besoins des interfaces uniformes. Ces opérations sont appelées CRUD (Create, Retrieve, Update, Delete).

3.3 RESTification du DNS

Nous abordons dans cette section les travaux qui ont ouvert la voie à l'intégration du DNS au paradigme REST (DNS RESTification), ainsi que leurs limitations.

Definition 3.3.1. La RESTification du DNS désigne le processus d'intégration du paradigme REST dans l'architecture DNS afin de faciliter ses évolutions vers des nouveaux domaines émergents.

3.3.1 DNS over HTTPS (DoH)

DoH [57] est un protocole proposé en 2018 et qui permet de transporter les requêtes/réponses DNS sur le protocole HTTPS (HTTP-over-TLS) afin de fournir une communication chiffrée entre le résolveur local du client et son résolveur récursif. Actuellement, le serveur faisant autorité ne prend pas en charge DoH.

Le fonctionnement de DoH est le suivant : i) l'application (ou le client final) envoie sa requête au résolveur local qui supporte DoH, ii) ce résolveur local la transmet ensuite au résolveur récursif de confiance (serveur DoH) situé sur l'internet, iii) et ce résolveur récursif envoie finalement la requête au serveur faisant autorité. Le principe de DoH est d'utiliser le protocole HTTP tel qu'il a été développé. Les requêtes et réponses DNS peuvent conserver leur encodage binaire ou utiliser le JSON, la requête est placée dans le chemin de l'URL ou dans le corps d'une requête HTTP (RFC 9113), et la réponse sera dans le corps de la réponse HTTP. Toute la sécurité (intégrité et confidentialité) de ces messages est assurée par TLS (RFC 8446), via HTTPS.

Le client DoH [57] utilise les méthodes GET ou POST pour transporter la requête DNS. Lorsque GET est utilisée, une variable est définie pour stocker le contenu de la requête DNS, suivant l'encodage conventionnel des données DNS , cette variable est ensuite sur-encodé en Base64, plus précisément la variante base64url standardisée dans la RFC 4648. Et, lorsqu'il s'agit de la méthode GET, le corps de la requête est vide (section 4.3.1, RFC 7231). Lorsque le client DoH utilise POST, la requête DNS se trouve dans le corps de la requête du protocole HTTP. Dans les réponses renvoyées au client DoH, le serveur DoH [57] doit mettre un code de retour HTTP (RFC 7231). 200 signifie que la requête HTTP a été traitée avec succès. Mais cela ne signifie pas que la requête DNS a été traitée avec succès. S'il a reçu une erreur DNS NXDOMAIN (nom introuvable) ou SERVFAIL (échec de la requête), le code de retour HTTP sera toujours 200, indiquant qu'il y a une réponse DNS, même si elle est négative.

Lorsqu'un client DoH reçoit ce code 200, il doit analyser le message DNS et trouver le code de retour DNS tels que : i) NOERROR, si la requête a été traitée avec succès et la reponse contient la ressource demandée, ii) NXDOMAIN, si le nom de domaine utilisé dans la requête n'existe pas, iii) SERFAIL, si le serveur DNS a rencontré une erreur lors de la résolution de la requête et n'a pas pu fournir de réponse, iv) REFUSED, le serveur DNS a refusé de répondre à la requête pour une raison quelconque. Le serveur DoH n'affichera un code d'erreur HTTP que s'il n'a aucune réponse DNS à renvoyer.

Il mettra ainsi le code 403 s'il refuse de servir ce client DoH, 429 si le client fait trop de requêtes (RFC 6585), 500 si le serveur DoH a un problème technique, et bien sûr, 404 si le serveur HTTP ne peut rien trouver à l'URL donnée, par exemple, parce que le service a été supprimé. Dans tous ces cas, aucune réponse DNS n'est incluse. La sémantique de ces codes de retour et le comportement attendu du client suivent les règles

habituelles de HTTP. Actuellement, DoH est pris en charge par les principaux résolveurs, notamment Cloudflare [50], Google [51] et Quad9.

Le protocole sous-jacent TCP utilisé sur DoH est l'un des protocoles de transport les plus populaires sur le web au cours de ces dernières décennies. Bien que ce protocole ait énormément fait ses preuves dans le web, il présente aussi plusieurs inconvénients, on peut citer entre autres le blocage de tête ligne communément appelé *HOL blocking*.

Le problème avec TCP est qu'il transmet les paquets de données par segment. Ainsi, lorsqu'un client envoie une série de paquets pour effectuer une demande de connexion, le serveur lui répond avec une autre série de paquets qui lui est propre pour accuser réception. Ces paquets sont regroupés en fonction de leur numéro de séquence. Les paquets de données ayant les numéros de séquence les plus récents ne peuvent pas être traités tant que les plus anciens ne le sont pas. Ce qui veut dire que si l'un des paquets de réponse se perd à cause d'une dégradation de la connexion entre le client et le serveur, les paquets ayant un numéro de séquence inférieur devront attendre la retransmission de ce paquet. Cela peut avoir des conséquences très néfastes sur la latence des communications. De plus, le volume des échanges TLS permettant de sécuriser le canal entre le client et serveur reste aussi important sur DoH.

3.3.2 Impacts de la RESTification

Nous nous intéressons aux travaux de Chhabra et al [34] qui réalisent une étude approfondie sur les performances de DoH sur des données récoltées dans 224 pays et territoires et auprès de 22052 clients. À notre connaissance, ces travaux [34] présentent l'étude la plus vaste et la plus complète sur plusieurs zones géographiques. L'architecture utilisée présentée à la Figure 3.1 est composée d'un client DoH qui souhaite effectuer la requête. Ce client se connecte à Super Proxy BrightData, qui est un proxy HTTP, un nœud de sortie est utilisé pour relayer les requêtes vers les pays désirés.

3.3.2.1 Durée d'une résolution DoH

Dans les architectures implémentant DoH, le temps d'aller-retour (ou RTT) d'une résolution de noms se fait généralement pour la première fois en trois phases :

Etape (1-8): Etablissement de connexion TCP

Dans cette première phase, le client cherche à établir une connexion TCP avec le serveur DoH afin de sécuriser le canal de communication. Ainsi à l'étape (1-2), on a une ouverture de la connexion à travers Super proxy. À l'étape (3-4) le nœud de sortie transmet la requête au résolveur récursif pour avoir l'adresse IP du serveur de noms faisant autorité. Après avoir obtenu cette adresse, le nœud effectue une poignée de main TCP avec le serveur DoH (etape 5-6).

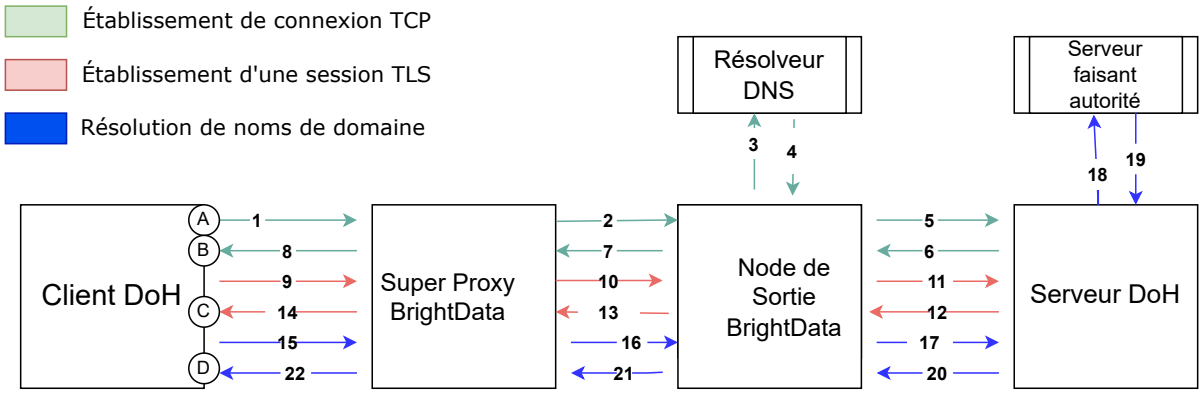


Figure 3.1: Evaluation des effets du REST sur une architecture DNS traditionnelle selon [34]

Une fois le poignée de main effectuée, à l'étape (7), le nœud de sortie répond au Super Proxy en lui envoyant les en-têtes de réponse HTTP contenant des informations temporelles utiles : le temps nécessaire à la résolution du nom de domaine du serveur DoH (t3 + t4) et l'heure de la poignée de main TCP (t5 + t6). À l'étape (8), le Super Proxy renvoie un "200 OK" au client DoH, établissant ainsi le tunnel TCP.

Etape (9-14): Etablissement d'une TLS Le message clientHello est envoyé par le client pour etablir une session TLS à l'étape (9-11). Le serveur DoH repond avec un message ServerHello à l'étape (12-14).

Etape (15-22): Résolution de noms de domaine

À l'étape (15-17), le client envoie une requête Finished et une requête HTTP GET au résolveur DoH pour résoudre le nom de domaine cible "<UUID>.a.com". Le résolveur DoH résout ensuite le nom de domaine en contactant le serveur de noms faisant autorité à l'étape (18-19). À l'étape (20-22), le serveur DoH crypte l'adresse IP résolue et la renvoie à notre client, achevant ainsi la résolution DoH.

Le temps de résolution dans ce scénario est :

$$t_{DoH} = (t_3 + t_4 + t_5 + t_6) + (t_{11} + t_{12}) + (t_{17} + t_{18} + t_{19} + t_{20}) \tag{3.1}$$

Avec un RTT stable entre le client DoH et le noeud de sortie on a :

$$RTT = (t_1 + t_2 + t_7 + t_8) + (t_9 + t_{10} + t_{13} + t_{14}) + (t_{15} + t_{16} + t_{21} + t_{22}) \tag{3.2}$$

Si on calcule le temps qui s'écoule entre D et C, on a :

$$T_D - T_C = \sum_{i=9}^{22} t_i \tag{3.3}$$

De même, le temps entre B et A est :

$$T_B - T_A = \sum_{i=1}^{8} t_i + t_{BrightData} \tag{3.4}$$

Ainsi, en se basant sur les équations précédentes, le temps de résolution est :

$$t_{DoH} = (T_D - T_C) - 2 \times (T_B - T_A) + 3 \times (t_3 + t_4 + t_5 + t_6) + t_{BrightData} \tag{3.5}$$

Ce qui implique finalement :

$$t_{DoH} = (\sum_{i=9}^{22} t_i) - 2 \times (T_B - T_A \sum_{i=1}^{8} t_i + t_{BrightData}) + 3 \times (t_3 + t_4 + t_5 + t_6) + t_{BrightData} \tag{3.6}$$

A travers ces équations, Chhabra et al [34] montrent ainsi dans leurs travaux que le temps de résolution DNS dans un environnement implémentant DoH dépend fortement du temps de connexion TCP (Etape 1-8) et du temps d'établissement de session TLS instaurés (Etape 9-14) par HTTPS.

Bien que DoH ait introduit les principes de REST dans le DNS et grâce à lui, les échanges DNS peuvent désormais être encapsulés dans le protocole HTTPS qui offre des mécanismes de sécurisation des informations des utilisateurs sur le Web, sa mise en oeuvre a un impact important sur les performances du DNS notamment en ce qui concerne la durée d'une résolution.

3.3.3 Etude comparative entre REST et DNS

	DNS	**REST**
Domaine d'application	Traduction des noms de domaine afin de faciliter la communication entre les appareils sur Internet.	Style architectural utilisé dans la conception des services web.
Protocoles de transport	Fonctionne sur UDP	Fonctionne sur HTTPS
Performances	Dépendent essentiellement de la rapidité avec laquelle les serveurs DNS répondent aux requêtes de résolution	Peut offrir des performances rapides grâce à la réduction du nombre de messages et à la prise en charge de la mise en cache.
Capacité de passage à l'échelle	La structure hiérarchique du DNS facilite son passage à l'échelle	Facile à dimensionner et sans état.
Sécurité	Les messages DNS sont non chiffrés	Prend en charge le chiffrement
Format des données	Prise en charge du format binaire et de JSON	Prise en charge des formats textuels tels que XML, JSON et HTML

3.3.4 Différences entre DoH et DoQ

Comme nous l'avons présenté dans les deux sections présentes, DoH et DoQ sont deux protocoles de résolution de noms de domaine qui visent à sécuriser les échanges DNS. La principale différence entre les deux protocoles est le protocole de transport sous-jacent utilisé pour transmettre les requêtes et les réponses DNS. DoH utilise le protocole HTTPS pour encapsuler les requêtes DNS, tandis que DoQ utilise le protocole de transport QUIC pour transmettre les requêtes et les réponses DNS. Nous identifions les différences importantes suivantes :

1. **Performances** : DoQ est généralement considéré comme plus rapide que DoH car il utilise le protocole de transport QUIC pour améliorer les performances des connexions Internet en réduisant les temps de latence et en augmentant le débit. Cela est possible grâce à des techniques telles que le multiplexage, le chiffrement intégré, la correction d'erreurs et l'utilisation de connexions persistantes lors de la résolution.
2. **Complexité** : DoQ est plus récent et moins répandu que DoH, ce qui peut rendre sa mise en œuvre plus complexe. Cependant, DoH peut être aussi plus complexe à mettre en œuvre sur les serveurs DNS, car il nécessite un serveur Web pour gérer les requêtes HTTPS.

3. **Déploiement**: DoH est plus couramment utilisé que DoQ à l'heure actuelle. Les navigateurs Web populaires comme Firefox et Chrome prennent en charge DoH, tandis que la prise en charge de DoQ est encore très limitée.

Bien que l'approche DoQ soit plus rapide que DoH, elle n'est pas basée sur le style architectural REST qui facilite la gestion des ressources de l'Internet à travers ses verbes HTTP. On peut aussi noter que DoQ ne constitue pas en soi une amélioration de DoH car les deux protocoles suivent des voies complètement différentes. L'IETF a normalisé en juin 2022, HTTP/3.0, qui est un protocole basé sur QUIC. A notre connaissance, il existe aujourd'hui deux approches qui pourraient réduire l'impact de la RESTification du DNS. Cela pourrait ainsi ouvrir la voie à un usage du DNS comme base de données distribuée de ressources Internet manipulables à travers des méthodes REST (GET, PUT, POST, DELETE,..) :

1. Utiliser HTTP/3.0 mais sa normalisation étant très récente il est encore très peu répandu.
2. Utiliser les protocoles développés pour IoT qui offrent une certaines flexibilité.

Dans cette thèse, nous avons choisi la deuxième approche qui sera développée dans la section suivante.

3.4 Extension de l'Internet aux environnements contraints

En 2012, l'IETF a créé le groupe de travail Constrained RESTFul Environment (CoRE). Son but est d'étendre l'architecture WEB aux applications Machine to machine (M2M) [65] et à l'IoT. Il est donc chargé de développer des protocoles de transport et de sécurité plus légers ainsi que des formats d'encodage de données plus compacts et plus souples que ceux utilisés traditionnelement sur le Web.

L'un des premiers travaux de ce groupe portait sur le développement du protocole CoAP (Constrained Application Protocol) [137] qui a été publié dans le RFC 7252 en juin 2014. CoAP est un protocole de communication générique optimisé pour les architectures contraintes. Il est principalement destiné aux appareils dont les ressources sont très limitées. Pour ces appareils, les protocoles tels que HTTP et TCP sont trop restrictifs. Bien qu'il soit basé sur le style architectural REST, il utilise une approche qui est légèrement différente de celle de HTTP, en utilisant des codes d'erreur plus petits et plus spécifiques pour minimiser la quantité de données transférées. De plus, CoAP utilise UDP comme protocole de transport, ce qui le rend plus léger et plus adapté aux environnements IoT. Sa proximité avec HTTP facilite le développement de passerelles entre le monde des objets connectés et le Web actuel.

La figure 3.2 présente l'interaction entre le web et un environnement contraint basé sur le paradigme REST. Cette architecture permet à un client utilisant le protocole CoAP d'accéder à une ressource d'un serveur qui utilise le protocole HTTP et vice-versa. Un proxy HTTP-COAP est nécessaire. Les travaux de Colitti et al [156] décrivent la conception et la mise en œuvre d'un tel système.
Le groupe CoRE a aussi proposé le format CBOR, qui est un format de sérialisation binaire recommandé pour le protocole CoAP dont les objectifs sont différents des formats binaires tels que ASN.1 [1] et MessagePack [97].

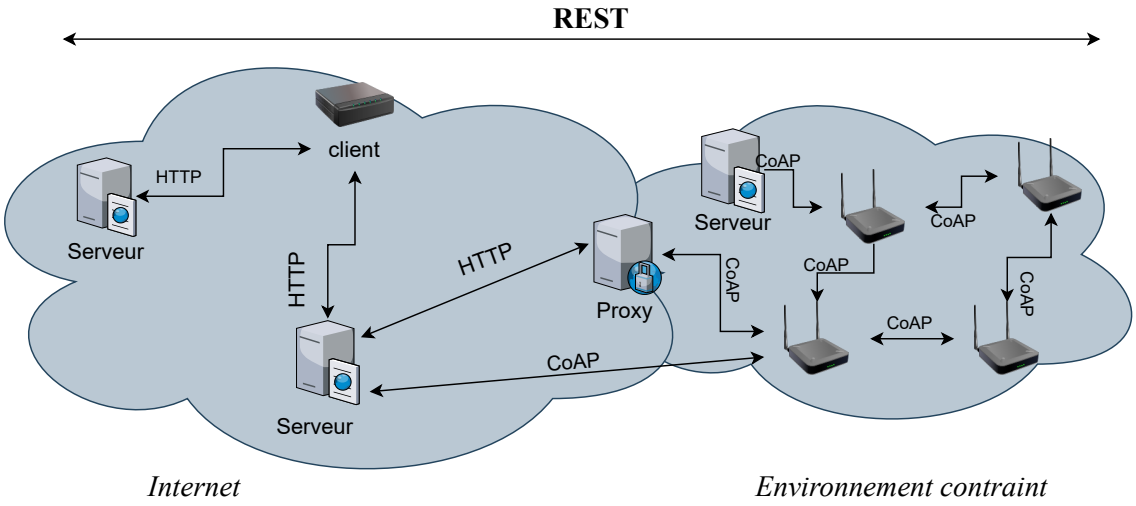


Figure 3.2: CoRE: Environnement Contraint RESTful [65]

3.5 Protocoles de l'IoT

3.5.1 Quelques avantages des protocoles de l'IoT

Les protocoles de l'IoT offrent de nombreux avantages qui contribuent à la diffusion et à l'adoption de cette technologie. Nous en rappelons dans cette section certains points forts de façon non exhaustive.

3.5.1.1 Interoperabilité

Dans le contexte de l'Internet des objets, l'interopérabilité fait référence à la capacité des différents appareils, systèmes et services IoT à fonctionner ensemble de manière transparente, malgré leurs différences de fabrication, de plateforme, de protocole ou de technologie. Cela signifie que les appareils IoT peuvent communiquer, échanger des données et collaborer efficacement, quel que soit leur fabricant ou leur fournisseur.

L'interopérabilité est essentielle pour le déploiement et l'adoption généralisée de l'IoT car elle permet la création de systèmes complexes et interconnectés, où différents appareils et services peuvent travailler ensemble pour offrir des fonctionnalités avancées et des solutions intégrées. Elle favorise également l'ouverture, la flexibilité et la possibilité de choix pour les utilisateurs, en évitant la dépendance à l'égard d'un fournisseur spécifique.

On distingue aujourd'hui plusieurs niveaux d'interopérabilité dans l'IoT. Le premier niveau concerne l'Interopérabilité des appareils IoT, dans ce cas, les appareils IoT doivent être capables de se découvrir, de s'authentifier mutuellement et d'établir une communication fiable, indépendamment de leurs différences matérielles, de leurs capacités de traitement ou de leurs protocoles de communication. On a aussi l'interopérabilité des données IoT, les données générées par les appareils IoT à cette étape doivent pouvoir être comprises, échangées et utilisées par d'autres appareils ou services IoT. Cela nécessite des formats de données normalisés (CBOR, JSON,..etc), des protocoles de transfert de données compatibles et des mécanismes d'intégration de données efficaces. Le dernier niveau est l'interopérabilité des services IoT, elle permet d'interconnecter et combiner plusieurs services afin d'ouvrir des fonctionnalités plus avancées. Cela nécessite ainsi des interfaces standardisées, des protocoles de découverte de services et une architecture flexible permettant la composition et la gestion de services hétérogènes.

3.5.1.2 Flexibilité

En ce qui concerne la flexibilité, elle représente la capacité des systèmes, des appareils et des services IoT à s'adapter et à répondre efficacement aux changements de conditions, de besoins ou d'exigences. Elle implique la possibilité de modifier ou d'ajuster rapidement les configurations, les fonctionnalités ou les paramètres des composants de l'IoT pour répondre aux demandes changeantes.

La flexibilité dans l'IoT revêt une grande importance en raison des caractéristiques évolutives et dynamiques des déploiements IoT, ainsi que des besoins changeants des utilisateurs et des environnements. Au niveau des données, cette flexibilité implique la possibilité de gérer et de traiter des volumes importants de données générées par les appareils IoT, tout en fournissant des mécanismes d'accès et de requête flexibles. Cela peut comprendre des outils d'analyse et de traitement de données en temps réel, des protocoles de transfert de données flexibles et des mécanismes d'intégration de données avec d'autres systèmes.

3.5.1.3 Faible Latence

Elle représente le temps écoulé entre l'envoi d'une demande ou d'une action et la réception de la réponse correspondante. La faible latence est cruciale dans de nombreux scénarios IoT où une réponse rapide et en temps réel est essentielle. Dans les environnements industriels par exemple, une basse latence est nécessaire pour permettre une communication rapide

entre les capteurs, les actionneurs et les systèmes de contrôle, afin de garantir des opérations synchronisées et de prendre des décisions en temps réel. Elle peut être aussi nécessaire dans les systèmes de transport intelligents qui exigent une basse latence pour fournir des réponses rapides aux conditions de la circulation, aux dangers potentiels et aux changements dans l'environnement routier.

Pour assurer cette faible latence, certains travaux de recherche proposent de déployer les capacités de calcul et de traitement au niveau de la périphérie du réseau, proche des appareils IoT. Des protocoles de communication IoT tels que Message Queuing Telemetry Transport (MQTT) ou Constrained Application Protocol (CoAP), peuvent être aussi utilisés.

3.5.2 CoAP

CoAP [137] est un protocole de communication conçu pour les appareils et les réseaux contraints, tels que les objets connectés et les réseaux basse consommation. Il a été spécifié par l'IETF dans la RFC 7252. Il est conçu pour fonctionner sur les réseaux IP (Internet Protocol) et utilise le protocole UDP (User Datagram Protocol) comme couche de transport. Cela permet à CoAP d'être utilisé efficacement dans des environnements à faible bande passante et à faible consommation d'énergie.

De plus, il utilise deux couches, une couche de messagerie CoAP utilisée pour gérer le manque de fiabilité d'UDP et la nature asynchrone des interactions, et une couche de requête/réponse héritée de HTTP (requêtes GET, POST, PUT, DELETE et autres catégories de réponses). La taille de la charge utile d'un message CoAP ne doit pas être supérieure à 1024 octets, mais il est possible de faire un transfert par blocs CoAP, ceci permet d'envoyer différents fragments du même message, chaque fragment étant considéré comme un message. La sémantique des requêtes et des réponses COAP est véhiculée dans les messages COAP qui contiennent des verbes HTTP, soit un code de retour tels que 404, 205, etc. Le jeton utilisé permet de faire correspondre une requête CoAP à une réponse. Une demande peut être transmise dans un message CON (Confirmable) ou NON (Non-confirmable) :

1. Si la requête du client CoAP est transmise dans un message CON et que la réponse du serveur est disponible immédiatement, elle sera transmise dans un message ACK (Acknowledgement). Si le serveur ne peut pas répondre immédiatement à une requête (par exemple, un GET) contenue dans un message CON, il réagit en envoyant simplement un message ACK sans contenu. Lorsque la réponse est prête, le serveur l'envoie dans un nouveau message CON (dont le client doit accuser réception par un message ACK sans contenu),
2. Si une requête est envoyée dans un message NON (Non-confirmable), la réponse est envoyée dans un nouveau message NON. Lorsque le serveur CoAP n'est pas en

mesure de traiter un message NON (ni même de fournir une réponse d'erreur), il répond par un message RST (Reset) au lieu du message ACK.

Comme le montre la Figure 3.3, le protocole CoAP est divisé en deux couches principales. Une couche pour les requêtes/réponses et une autre pour les messages. Dans la section suivante, nous allons présenter ces deux couches.

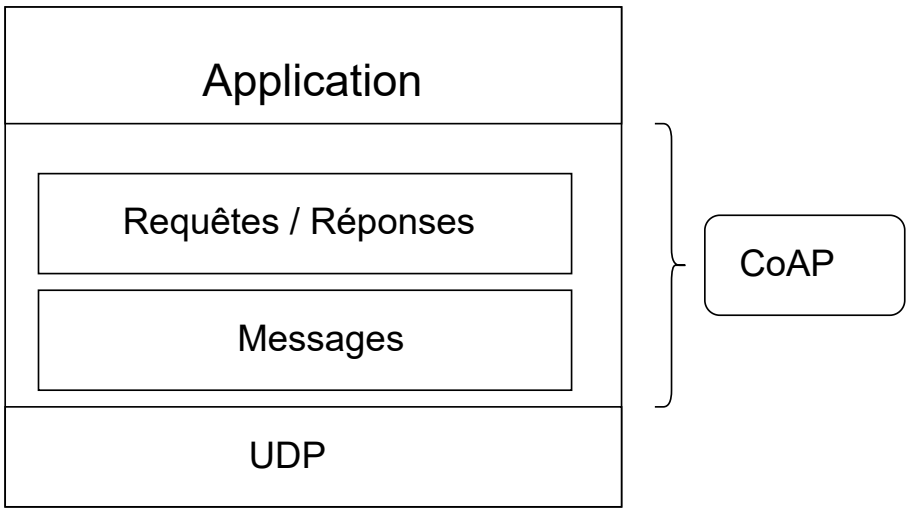


Figure 3.3: Les couches du protocole CoAP [137]

3.5.2.1 La Couche Messages

Pour encoder les données échangées, le protocole CoAP utilise un format de message qui comprend un en-tête de 4 octets de taille fixe, une valeur token de longueur variable, des options et une charge utile, comme le montre la figure 3.4.
L'en-tête du message CoAP comprend les éléments suivants :

- *Version (Ver) :* Ce champ est un entier non signé de 2 bits qui indique la version CoAP.
- *Type (T) :* Ce champ est un entier non signé de 2 bits qui indique le type de message. Il existe quatre types de messages différents pour CoAP : le message confirmable (CON), le message non confirmable (NON), le message d'accusé de réception (ACK) et le message de réinitialisation (RST). Les messages ACK et RST sont des types de réponse du serveur CoAP où le message ACK reconnaît l'arrivée d'un message confirmable particulier et le message RST indique l'absence d'un certain contexte dans les messages CON et NON. Les messages CON et NON sont des méthodes de requête/réponse CoAP. CON sert à montrer la fiabilité d'un message et nécessite

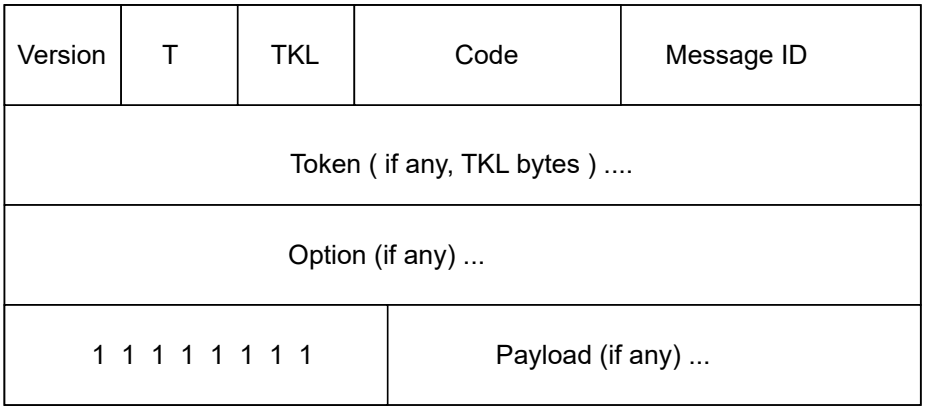


Figure 3.4: Le format de message CoAP [137]

un ACK. NON est un message qui n'a pas besoin d'une transmission fiable ni d'un ACK, mais qui comporte une identification de duplication Message-ID.

- *Token Length (TKL) :* Ce champ est un entier non signé de 4 bits qui indique la longueur du champ Token (0-8 octets).
- *Code :* Ce champ est un entier non signé de 8 bits, qui est expliqué dans la partie portant sur la couche Message .
- *Message ID :* Ce champ est un entier non signé de 16 bits qui est utilisé pour faire correspondre les messages de type CON/NON avec ACK/RST et pour détecter la duplication des messages.

3.5.2.2 La Couche Requête/Réponse

Au niveau de la couche requête/réponse, le client CoAP envoie une ou plusieurs requêtes au serveur CoAP. Lorsque le serveur reçoit la requête, il répond en fonction du type de message. Les différentes requêtes et réponses sont échangées de manière asynchrone. Le message CoAP contient un code de méthode ou un code de réponse. Le message CoAP contient également des informations facultatives sur la requête et la réponse décrites précédemment, telles que l'URI, le type de média de la charge utile et le jeton permettant de faire correspondre les requêtes et les réponses.

La requête d'un client CoAP peut être transmise soit dans un message CON (Confirmable), soit un message NON (NON - Confirmable). Pour une requête de type CON, si côté serveur la réponse est immédiatement disponible, elle sera transmise dans un message ACKnowlegement(ACK). Dans le cas où le message ACK n'est pas transmis correctement, le client CoAP retransmettra son message CON. La figure 3.5 montre des exemples portant sur une requête utilisant la méthode GET transportée via un message CON dont la réponse retournée par un serveur CoAP sera dans un message ACK.

Dans ce scénario, si le serveur est incapable d'envoyer une réponse immédiatement après la réception de la requête de type CON du client, il envoie juste une réponse ACK sans payload en attendant la disponibilité de la réponse. Si la réponse existe, il répond via le message CON contenant le code du succès (2.05).

Dans le scénario présenté à la figure 3.6, si la requête du client CoAP est envoyé dans

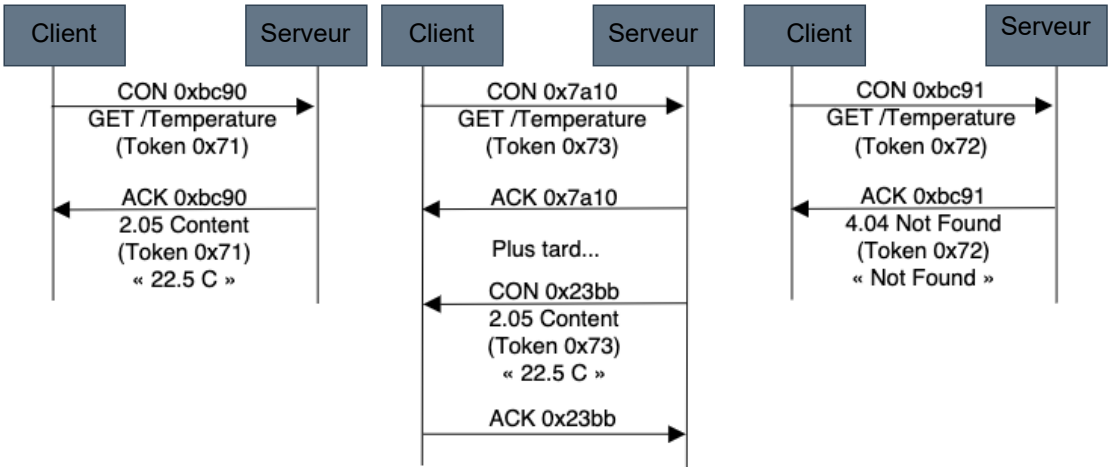


Figure 3.5: Message de transmission CoAP de type confirmable(CON) [137]

un message de type NON-confirmable (NON), alors la réponse retournée sera dans un nouveau message de type NON.

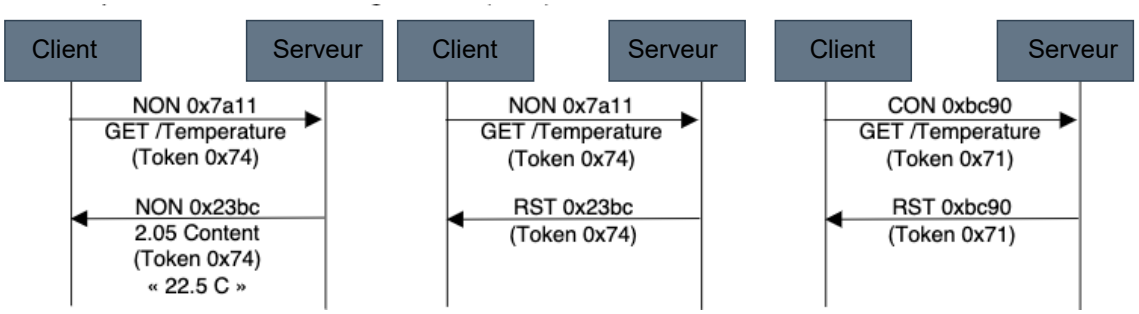


Figure 3.6: Message de transmission CoAP de type NON-confirmable(CON) [137]

3.5.2.3 Le protocole CoAP dans le modèle OSI

Le protocole CoAP est aujourd'hui bien intégré dans le modèle OSI (Open Systems Interconnection) à cinq couches : physique, liaison de données, réseau, transport et

application. Au niveau de la couche 7, le protocole permet de communiquer comme HTTP avec des applications REST. Les messages sont encapsulés dans UDP au niveau de la couche transport, permettant ainsi d'économiser la bande passante et de diminuer la latence.

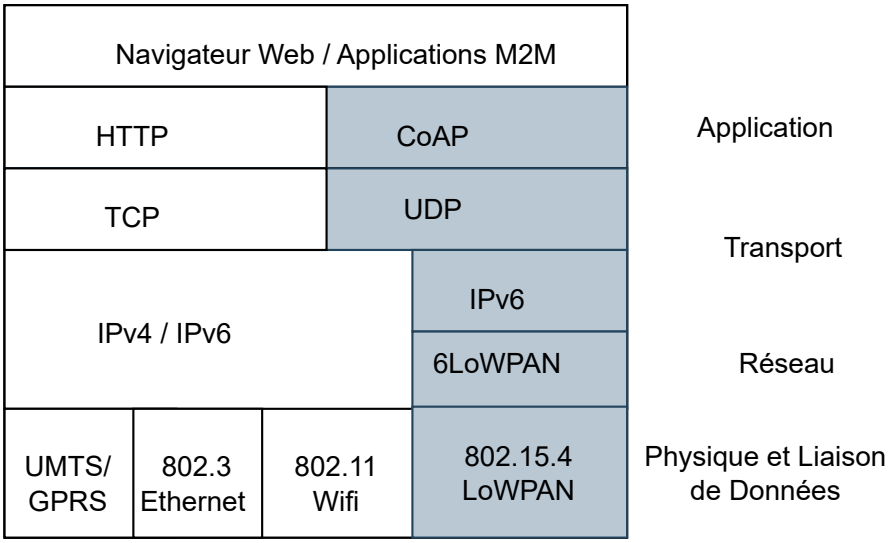


Figure 3.7: Le protocole CoAP et le modèle OSI [137]

3.5.3 DTLS

Datagram Transport Layer Security (DTLS) [153] suit de près TLS [123] en terme de comportement et de garanties de sécurité. Contrairement à son parent orienté flux, DTLS ajoute des facilités pour fonctionner dans des environnements de datagrammes non fiables. Il apporte une couche d'enregistrement modifiée qui tolère la perte de paquets et le réordonnancement des messages. Pour briser les dépendances entre enregistrements, DTLS interdit les algorithmes de chiffrement de flux et utilise des numéros de séquence explicites dans les datagrammes. Un contexte cryptographique s'étend donc sur un seul enregistrement. UDP est le transport le plus courant dans les déploiements IoT. Comparé à TCP, il ne présente pas de surcharge protocolaire substantielle et permet des implémentations à faible empreinte mémoire. Aujourd'hui DTLS est très utilisé pour sécuriser des données transportées par certains protocoles IoT tels que CoAP et MQTT-SN. La figure 3.8 présente les échanges de poignées de main entre un client et serveur qui implémentent le protocole DTLS. On remarque que DTLS utilise quasiment les mêmes communications de poignées de main et de flux que TLS, à l'exception de trois modifications essentielles :

1. Pour éviter les dénis de service, un échange de cookies sans état a été ajouté dans DTLS.

2. Des modifications ont été apportées à l'en-tête de la poignée de main DTLS pour gérer la perte de message, le ré-ordonnancement et la fragmentation IP.
3. Pour gérer la perte de messages, un délai de retransmission a été ajouté dans DTLS. En plus des exemples mentionnés ci-dessus, le flux et la logique des formats de messages DTLS sont similaires à ceux de TLS.

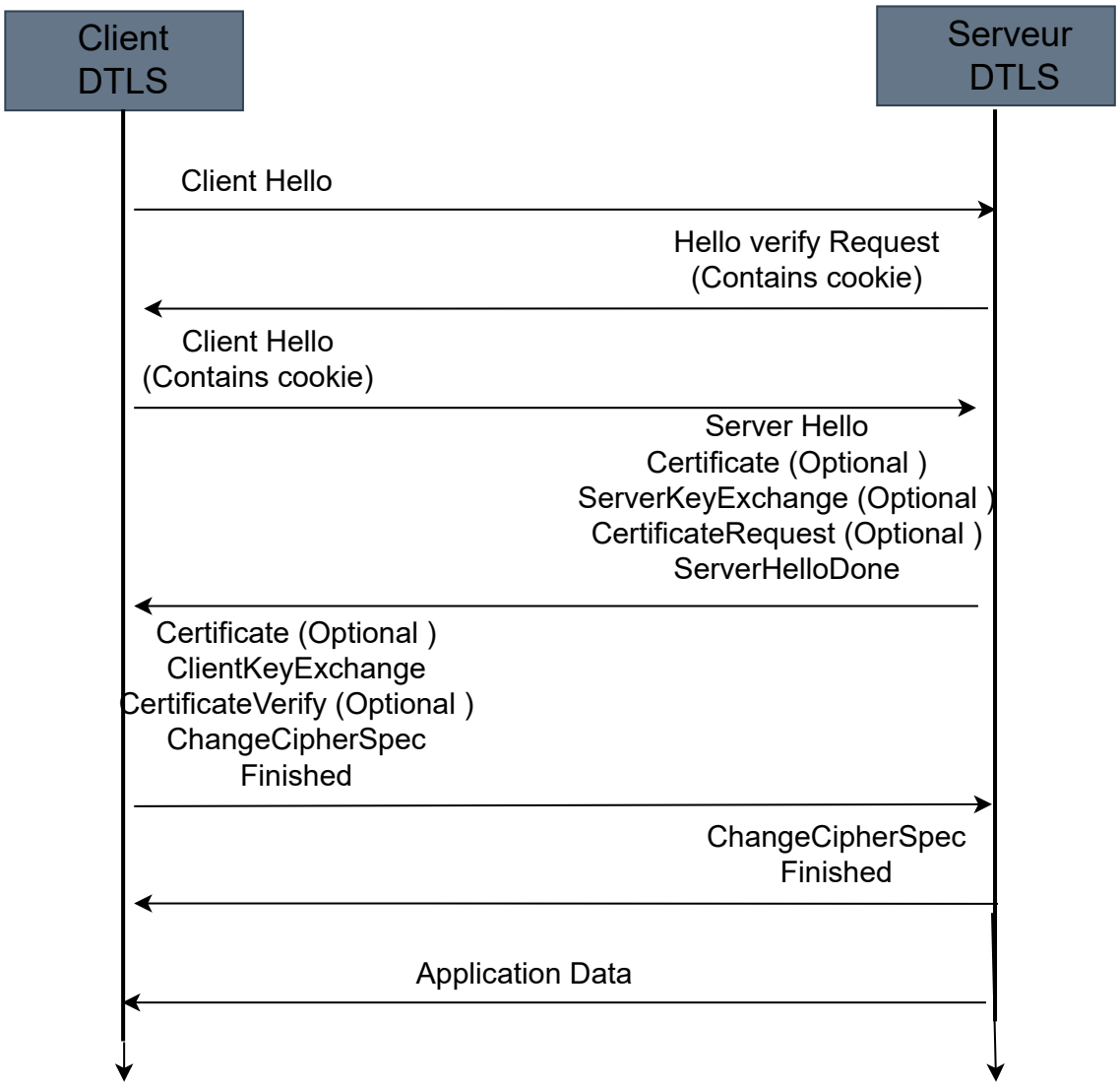


Figure 3.8: Poignée de main avec DTLS [153]

3.5.4 OSCORE

Le protocole Object Security for Constrained RESTful Environments (OSCORE) [132] est une méthode de sécurité de la couche application pour le protocole CoAP [137] au moyen

de CBOR Object Signing and Encryption (COSE) [130]. CBOR [152] est un format de données conçu pour un code et un message de petite taille, qui a modifié le modèle de données JavaScript Object Notation (JSON) [143] en autorisant, entre autres, les données binaires. CBOR est utilisé pour le codage compact dans OSCORE. La structure COSE organise tous les messages de sécurité sur la base du type de tableau CBOR, qui est utilisé pour les structures de chiffrement et de dérivation des clés. La figure 3.9 ci-dessous donne un aperçu schématique des échanges de messages d'OSCORE lorsqu'il est utilisé par CoAP.

OSCORE peut être utilisé à la fois pour le transport fiable et non fiable, car ces méthodes ne diffèrent que dans la couche de messagerie CoAP, qui n'est pas protégée par OSCORE. En outre, OSCORE protège les interactions RESTful telles que la méthode de requête, la ressource demandée et la charge utile du message, comme le montre la figure 3.11. OSCORE ne protégeant que les informations pertinentes de la couche d'application, la surcharge de messages est minime.

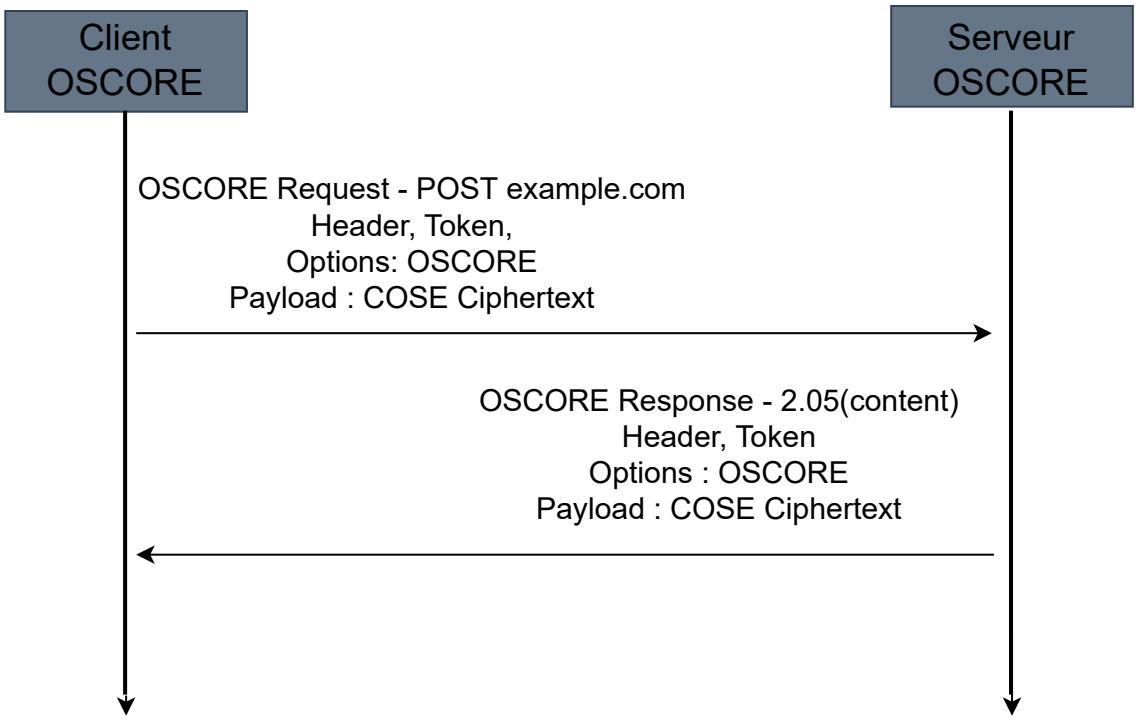


Figure 3.9: Poignée de main avec OSCORE [132]

3.5.5 EDHOC

Ephemeral Diffie-Hellman Over COSE (EDHOC) [133] est un protocole d'échange de clés Diffie-Hellman authentifiées très compact et léger permettant d'établir un secret partagé

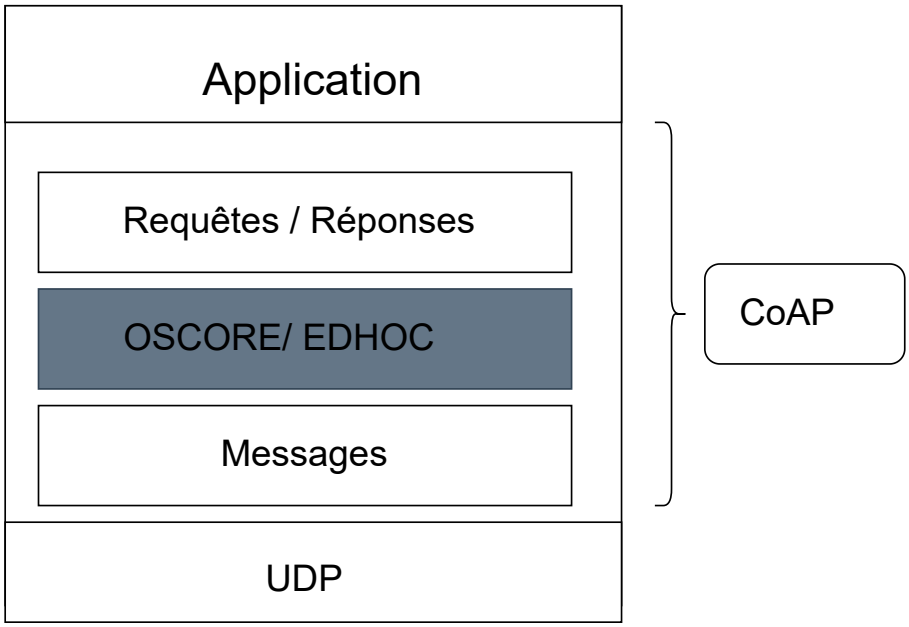


Figure 3.10: Le protocole CoAP avec OSCORE/EDHOC [133, 132]

sur la base d'une clé éphémère, offrant un secret parfait, une protection de l'identité et une authentification mutuelle sur la base d'un justificatif d'identité établi hors bande. Le protocole est décrit dans [86] comme un projet de l'IETF en cours d'élaboration, récemment adopté par le groupe de travail LAKE de l'IETF.

3.5.6 Une première optimisation : DNS over CoAP (DoC)

Dans [82], les auteurs ont proposé en 2021 une technique d'encapsulation des données DNS sur le protocole CoAP, baptisée DNS over CoAP (DoC) pour effectuer une résolution DNS sur un équipement IoT. Leur méthode permet aussi de sécuriser le canal de communication entre l'appareil IoT et le serveur DNS faisant autorité pendant cette résolution. Le protocole CoAP utilisé suit le modèle RESTful en utilisant des URIs pour identifier les ressources et des méthodes de requête similaires à celles d'HTTP (comparables à GET, POST, PUT, DELETE) pour manipuler ces ressources. Cependant les méthodes de requête CoAP sont optimisées pour les réseaux contraints en bande passante et en énergie, ce qui les rend plus légères et plus efficaces que les méthodes HTTP standard.
Pour évaluer les performances de leur modèle, les auteurs ont réalisé 50 requêtes DNS, à raison de 5 requêtes par seconde, pour obtenir les adresses IPv6 associées aux noms de domaine de 24 caractères. L'objectif était d'évaluer le temps de résolution et la taille des paquets DNS que le protocole CoAP est susceptible de transporter sur un réseau contraint. Pour ce faire, 10 tests ont été effectués en utilisant un système Cortext - M3

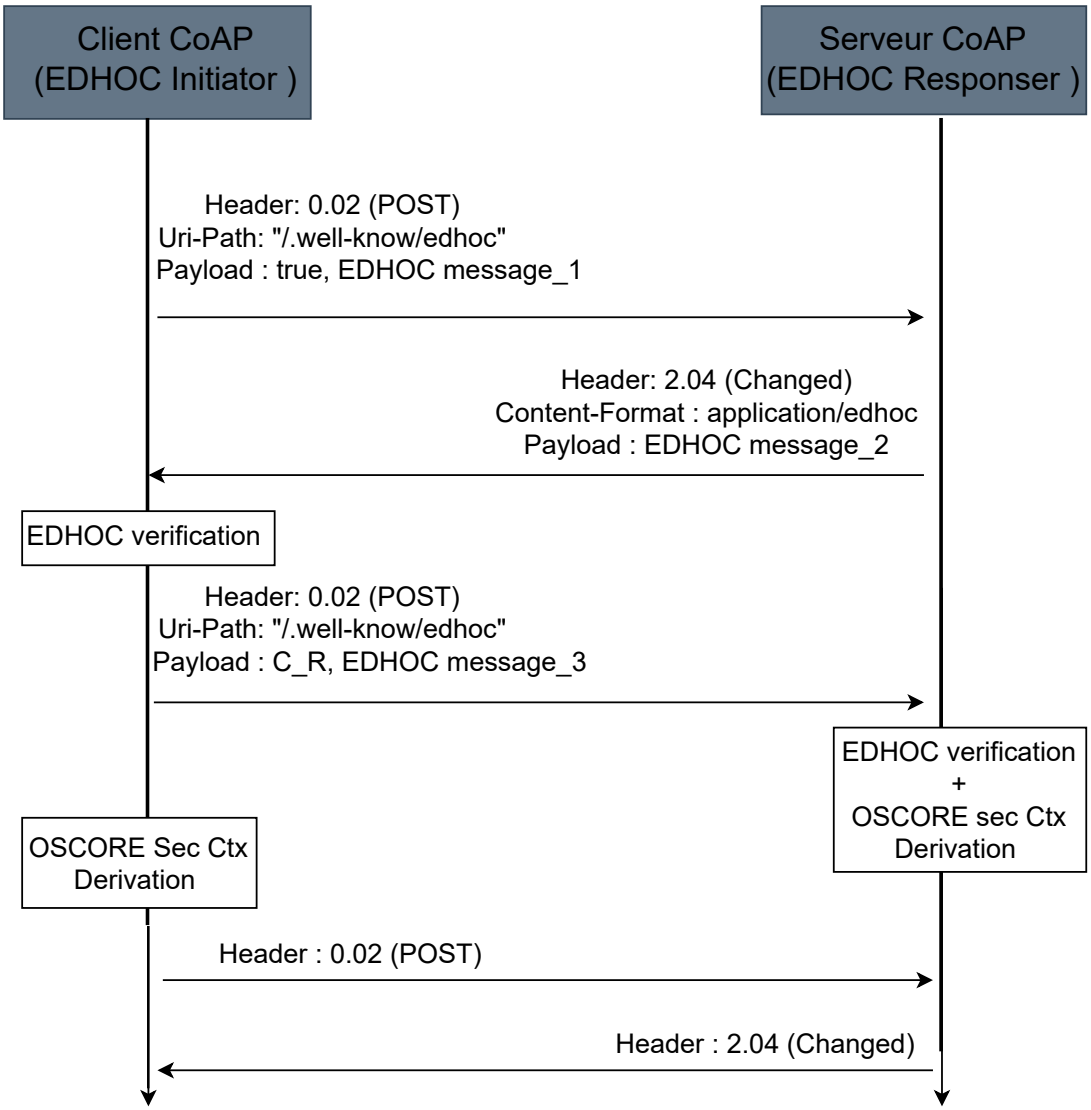


Figure 3.11: Execution de EDHOC sur le protocole CoAP [114]

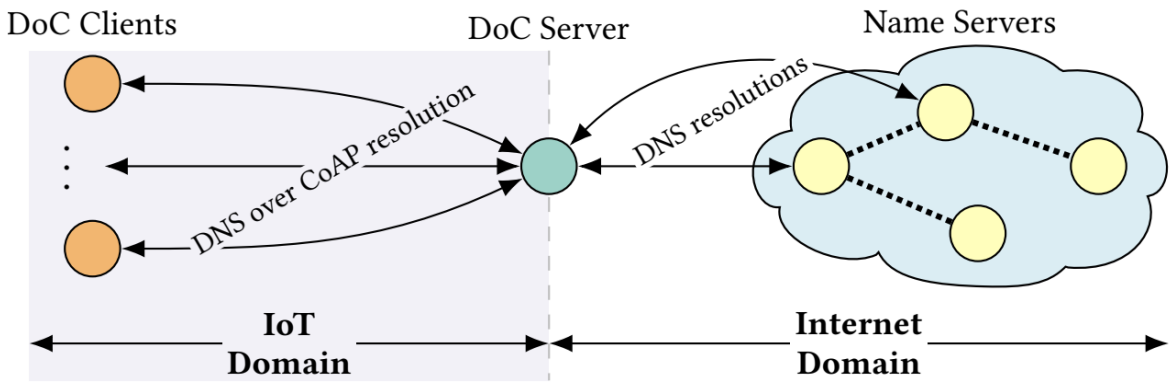


Figure 3.12: Architecture de DoC [81][82][83]

équipé d'une antenne radio IEEE 802.15.4 qui joue le rôle d'un équipement IoT. Les expériences ont été séparées en trois groupes :

1. La première expérience a été réalisée uniquement en résolution classique via UDP.
2. Dans la deuxième expérience, des méthodes CoAP telles que FETCH et POST ont été ajoutées et CoAP et UDP sont utilisés comme moyens de transport.
3. Dans la troisième expérience, les requêtes DNS sont transportées seulement par CoAP et les échanges sont sécurisés par DTLS et OSCORE.

Les auteurs ont constaté que dans le premier groupe d'expériences où seul UDP est utilisé comme moyen de transport, il n'y a pas de fragmentation des messages DNS. Alors que dans le deuxième groupe utilisant UDP et CoAP, la requête DNS n'était pas fragmentée, par contre la réponse renvoyée par le serveur DNS l'était. Enfin, requête et réponse étaient toutes deux fragmentées dans le groupe 3 qui utilise seulement CoAP avec des protocoles de sécurité supplémentaires tels que DTLS et OSCORE.

Bien que DoC ait réduit le volume de données échangées lors de la résolution DNS en remplaçant HTTP par un protocole IoT plus léger comme CoAP, les auteurs ont indiqué que leur approche souffre d'un problème de fragmentation des messages DNS. Ce qui peut avoir un impact significatif sur les performances particulièrement dans un environnement où les ressources sont limitées.

3.6 Conclusion

Dans ce chapitre, nous avons examiné divers travaux concernant l'intégration du DNS dans le paradigme REST ainsi que leurs limitations. Nous avons également présenté quelques protocoles IoT. Le modèle de représentation et de transfert de données basé sur REST reste très répandu dans le domaine du Web aujourd'hui. Il offre une accessibilité aisée aux ressources Internet via les verbes HTTP. L'incorporation de ce modèle architectural dans

le DNS a été une avancée majeure, renforçant non seulement sa sécurité, mais ouvrant également la possibilité de l'utiliser dans de nouveaux domaines où une faible latence lors de la communication est cruciale. Cependant, l'utilisation de REST présente également des inconvénients, notamment en ce qui concerne ses performances dans le DNS. Le protocole TCP utilisé dans HTTP peut consommer d'importantes ressources, ce qui va à l'encontre de la nécessité d'une faible latence pour le DNS.
Aujourd'hui, les technologies développées pour l'IoT offrent de nombreux avantages en raison de leur efficacité en termes de consommation de ressources. Ainsi CoAP repose sur les mêmes principes que HTTP qui est actuellement utilisé par le standard DoH. Le remplacement de HTTPS par le protocole CoAP pourrait contribuer à réduire l'impact du protocole TCP sur la latence du DNS mais selon les auteurs[82], cela doit passer par la définition d'un format de représentation des messages DNS plus flexible, compatible avec le format existant, et surtout adapté au protocole CoAP.
Dans le chapitre suivant, nous examinons en détail les différents formats de représentation binaire afin de déterminer lequel sera retenu pour représenter les informations dans le protocole CoAP.

Chapter 4

Analyse des formats d'encodage des données utilisés par le DNS

Every networking problem always takes longer to solve than it seems like it should.

— **RFC 1925, Nr. 9a**
(The Twelve Networking Truths)

Sommaire

4.1 Introduction

Au début années 1980 [102, 103], après avoir créé le système de noms de domaines pour résoudre les problèmes de croissance et de gestion des tables d'hôtes du réseau ARPANET [56], il fallait également un moyen standardisé de représenter les messages DNS pour faciliter les échanges entre les clients et les serveurs DNS. Le "DNS wireformat" a été développé pour répondre à ce besoin. Il utilise une représentation binaire des messages DNS permettant une transmission efficace sur les réseaux. La spécification de l'encodage binaire des messages DNS a été documentée dans le RFC 1035, publiée en novembre 1987.

Bien que le "DNS wireformat" soit largement utilisé depuis les débuts du DNS en raison de son efficacité et de sa compacité, il reste très difficile à lire et à comprendre pour les personnes non familières avec sa structure. Par ailleurs cette dernière étant fixe, l'ajout de nouvelles fonctionnalités et extensions sans modification de la spécification du protocole reste très difficile à réaliser aujourd'hui.

Avec l'évolution des technologies et des besoins, de nouvelles formes de représentation des données ont émergé, dont le format JSON. Ce dernier est un format léger et facilement lisible par les humains, ce qui en fait un choix populaire pour représenter des données structurées. Son adoption dans le domaine du DNS offre une alternative plus conviviale à la représentation binaire des messages DNS. Le passage du "wireformat" à JSON a également été stimulé par l'adoption de nouvelles technologies telles que les services DNS basés sur le cloud et les API RESTful. Ces technologies ont encouragé l'utilisation de formats de données plus flexibles et interopérables.

Dans ce chapitre, nous nous intéressons aux différents formats de représentation des messages DNS existant dans la littérature. Nous nous intéressons également aux formats de sérialisation binaire avec et sans schéma pour aboutir à une comparaison entre ces deux types formats afin de choisir celui qui sera adapté à nos travaux.

4.2 Formats des messages DNS

Dans cette partie, notre étude est concentrée sur les différents formats existants utilisés actuellement par le DNS ainsi que sur leurs limitations. L'approfondissement du format DNS obtenu nous aidera à déterminer comment nous pouvons le rendre plus flexible.

Definition 4.2.1. Un format de sérialisation binaire sans schéma est une méthode de représentation d'objets ou de données sous forme binaire sans inclure explicitement un schéma séparé pour décrire la structure des données.

4.2.1 Messages DNS en "wireformat"

L'un des avantages de DoH [57] présenté à la Section 3.3.1, est qu'il permet à un client HTTP qui encapsule le message DNS lors de l'opération de résolution, de spécifier au serveur HTTP le type de format de données qu'il supporte. La représentation actuelle par défaut des données DNS transportées sur HTTP est appelée base64url [139], qui est un encodage Base64 dont l'alphabet a été modifié pour permettre l'encodage d'URL. Avec le codage Base64, le flux binaire du message DNS est décomposé en groupes de six caractères et chacun de ces groupes est traduit en nombres qui ont une représentation textuelle en American Standard Code for Information Interchange (ASCII). Ainsi grâce à DoH, il est possible de changer la représentation DNS de binaire en textuelle, qui peut ensuite être envoyée au serveur HTTP. Ce nouveau format est appelé le "wireformat" et reste pour le moment le format par defaut de DoH. Une autre approche décrite dans un RFC informatif [57], utilise JSON pour représenter les messages DNS. Il convient de noter que Google et Cloudfare ont également apporté une contribution qui fait référence au format JSON sur DoH, mais qui est différente de [57] car elle ne prend pas en compte tous les champs des messages DNS décrits dans le standard RFC 1035 [103]
Bien que DoH permette d'introduire de nouveaux formats de représentation du message DNS, le système d'encodage de l'information qu'il utilise par défaut est celui de l'ASCII.

Le listing 4.1 montre un exemple d'une requête DNS pour `www.example.com` utilisant la méthode HTPP POST envoyée par le client DoH [50] [57] :

```
1
2  :method = POST
3  :scheme = https
4  :authority = cloudflare-dns.com
5  :path = /dns-query
6  accept = application/dns-message
7  content-type = application/dns-message
8  content-length = 33
9
10 <33 bytes represented by the following hex encoding>
11 00 00 01 00 00 01 00 00  00 00 00 00 03 77 77 77
12 07 65 78 61 6d 70 6c 65  03 63 6f 6d 00 00 01 00
13 01
```

Listing 4.1: Un exemple d'une requête DoH avec la méthode POST utilisant le "wireformat" pris dans la specification officielle [57]

```
1
2  :status = 200
3  content-type = application/dns-message
4  content-length = 64
5  cache-control = max-age=128
6
7  <64 bytes represented by the following hex encoding>
8  00 00 81 80 00 01 00 01  00 00 00 00 03 77 77 77
9  07 65 78 61 6d 70 6c 65  03 63 6f 6d 00 00 01 00
10 01 03 77 77 77 07 65 78  61 6d 70 6c 65 03 63 6f
11 6d 00 00 01 00 01 00 00  00 80 00 04 C0 00 02 01
```

Listing 4.2: Une réponse DoH de la méthode POST utilisant le "wireformat"

4.2.2 Messages DNS en JSON

JSON [143] est un format de données sans schéma largement utilisé pour l'échange de données sur le web. Il est apparu au milieu des années 2000 et a depuis gagné en popularité en raison de sa simplicité et de sa lisibilité pour les humains. Il a été inventé par Douglas Crockford dans le but de fournir une alternative plus simple au format XML [110] pour l'échange de données; il a été standardisé par l'IETF en juillet 2006 dans le RFC 8259. C'est un format étroitement associé au langage de programmation JavaScript, mais indépendant de tout langage; il peut être utilisé avec une grande variété de langages de programmation.

```
1
2 { "ID": 19678, "QR": 0, "Opcode": 0,
3       "AA": 0, "TC": 0, "RD": 0, "RA": 0, "AD": 0, "CD": 0, "RCODE": 0,
4       "QDCOUNT": 1, "ANCOUNT": 0, "NSCOUNT": 0, "ARCOUNT": 0,
5       "QNAME": "example.com", "QTYPE": 1, "QCLASS": 1
6    }
```

Listing 4.3: Exemple d'une requete DNS de type A en JSON - RFC 8427

```
1  { "ID": 32784, "QR": 1, "AA": 1, "RCODE": 0,
2    "QDCOUNT": 1, "ANCOUNT": 2, "NSCOUNT": 1,
3     ARCOUNT": 0,
4     "answerRRs": [ { "NAME": "example.com.",
5                                              "TYPE": 1, "CLASS": 1,
6                                              "TTL": 3600,
7                                              "RDATAHEX": "C0000201" },
8                    { "NAME": "example.com.",
9                                              "TYPE": 1, "CLASS": 1,
10                                             "TTL": 3600,
11                                             "RDATAHEX": "C000AA01" } ],
12    "authorityRRs": [ { "NAME": "ns.example.com.",
13                                                 "TYPE": 1, "CLASS": 1,
14                                                 "TTL": 28800,
15                                                 "RDATAHEX": "CB007181" } ]
16
17 }
```

Listing 4.4: Exemple d'une reponse DNS de type A en JSON - RFC 8427

Comme le montre le Listing 4.3, JSON reste un format facile à comprendre par l'homme. Il utilise une syntaxe concise basée sur des paires clé-valeur et des structures de données simples. Une structure de données codée à l'aide de JSON est appelée document JSON. Les documents JSON sont généralement plus compacts que les documents XML équivalents, ce qui permet de réduire la bande passante nécessaire pour transférer les données sur le réseau. Il utilise une structure de données hiérarchique, avec des objets et des tableaux imbriqués. Il utilise des accolades pour définir les objets et des crochets pour définir les tableaux. Il peut prendre en charge plusieurs types de données primitifs tels que les chaînes de caractères, les nombres, les booléens, les valeurs nulles, ainsi que les tableaux et les objets. Plusieurs approches sont aujourd'hui proposées dans la littérature pour encoder les messages DNS en JSON.

4.2.2.1 L'approche de Hoffman

Selon [57] les systèmes conformes à DoH qui effectuent la résolution de noms de domaines doivent encoder les messages DNS en utilisant les méthodes HTTP GET ou POST avec un corps de message transmis encodé en base64url. Lors de l'encodage des messages DNS en JSON [141], les données DNS sont mieux structurées et plus lisibles qu'en base64url : ainsi chaque section du message DNS est représentée par un objet JSON et les éléments constitutifs de ses sections sont mis sous la forme de paires clé/valeur.

4.2.2.2 L'approche de Google et de Cloudflare API

Google [51] a proposé un ensemble d'API qui permettent d'interroger le serveur DNS sur les différents types d'enregistrements et d'obtenir une réponse au format JSON [143]. Cloudflare[50] a mis en œuvre DoH en prenant en charge le format JSON pour représenter les données DNS. Dans son approche, les requêtes DNS sont envoyées à l'aide d'une méthode GET de HTTP. Lorsqu'une requête DNS est effectuée à l'aide de la méthode GET, la requête est encodée dans l'URL. Le client doit inclure un champ d'en-tête de requête `HTTP Accept` avec un type MIME `application/dns-json` pour indiquer qu'il est en mesure d'accepter une réponse JSON du résolveur DNS via HTTPS.

4.2.2.3 Comparaison entre les approches Hoffman et, Google et Cloudfare

Le tableau 5.3 ci-dessous décrit les champs de la réponse DNS qui ont été pris en compte ou ignorés par les différentes approches [141, 51, 50] lors du passage au format JSON.

4.2.2.4 Limitations du format JSON utilisé par le DNS

Malgré sa popularité, JSON n'est pas le format de sérialisation de données le plus efficace en termes de temps d'exécution ou d'espace.

Complexité temporelle: Lorsqu'on est sur des systèmes à forte intensité de données, sérialiser ou désérialiser les données devient une tâche très difficile, surtout lorsqu'on dispose de ressources limitées. Dans les travaux [113], les auteurs indiquent que les applications big data peuvent consacrer jusqu'à 90 % de leur temps d'exécution à la désérialisation de documents JSON. Cela est dû au fait que la désérialisation des formats textuels tels que JSON est généralement coûteuse lorsqu'elle est effectuée à l'aide d'algorithmes d'analyse syntaxique traditionnels basés sur des machines d'état. Les articles [48] mettent en évidence l'impact de la vitesse de sérialisation et de désérialisation sur la consommation de la batterie des téléphones portables et des plateformes mobiles à ressources limitées. Pour résoudre ce problème, [26] proposent des encodeurs et des décodeurs JSON prometteurs qui déduisent les schémas d'utilisation JSON pendant l'exécution et s'auto-optimisent en générant dynamiquement un code de codage et de décodage. De plus, [77] propose

Table 4.1: Champs DNS utilisés en fonction des approches de Google, Cloudfare et Hoffman

Champs	Sous-champs	Format JSON	
		Hoffman	Google et Cloudfare
HEADER	ID	✓	✓
	FLAG	✓	✓
	QDCOUNT	✓	-
	ANCOUNT	✓	-
	NSCOUNT	✓	-
	ARCOUNT	✓	-
QUESTION	QNAME	✓	✓
	QTYPE	✓	✓
	QCLASS	✓	-
ANSWER	NAME	✓	✓
	TYPE	✓	✓
	CLASS	✓	-
	TTL	✓	✓
	RDLENGTH	✓	-
	RDATA	✓	✓
AUTHORITY		✓	-
ADDITIONAL		-	-

une nouvelle approche pour analyser efficacement les documents JSON en utilisant les instructions SIMD du processeur. Les auteurs de [85] affirment que les applications analysent des documents JSON entiers, mais qu'en réalité elles n'utilisent généralement que certains champs. Ils suggèrent donc un parseur JSON paresseux qui déduit les schémas des documents JSON pendant l'exécution et utilise ces schémas pour spéculer sur la position des champs nécessaires afin d'éviter de désérialiser des parties inutiles des documents JSON.

Complexité spatiale: Les services du cloud sont généralement accessibles via Internet. Par conséquent, ces types de services sont particulièrement sensibles aux performances insuffisantes du réseau. Par exemple, en 2007, Akamai a découvert qu'un retard de 100 millisecondes dans le temps de chargement d'un site web peut réduire les taux de conversion de 7% et qu'un retard de deux secondes dans le temps de chargement d'une page web augmente les taux de rebond de 103%. Par rapport à JSON, [120] a constaté que l'utilisation d'un format personnalisé de sérialisation binaire réduisait le trafic réseau global de 94%. [23] concluent que JSON n'est pas un format approprié pour les systèmes de communication à bande passante limitée, citant la taille des documents comme raison principale. Les auteurs de [121] affirment dans leurs travaux que le trafic réseau est l'une des deux principales causes de consommation de batterie sur les appareils mobiles et

que, par conséquent, un format de sérialisation efficace en termes d'espace pourrait avoir un impact positif sur les économies d'énergie.
Dans les sections suivantes, nous abordons d'autres formats de sérialisation binaire pouvant encoder les messages DNS en occupant le moins d'espace possible notamment par rapport à JSON.

4.3 Formats de serialisation les plus populaires

Definition 4.3.1. La sérialisation est un processus de traduction d'une structure de données en une chaîne de bits (une séquence de bits) à des fins de stockage ou de transmission sur un réseau. La structure de données originale peut être reconstruite à partir de la chaîne de bits en utilisant un processus appelé désérialisation.

Dans la littérature, on distingue les formats de sérialisation binaires compatibles JSON

1. **Sans schéma**: BSON, CBOR [152], FlexBuffers [112], MessagePack [97], Smile [4] et UBJSON [3].
2. **Avec schéma**: ASN.1 [153], Apache Avro [144], Microsoft Bond [98], Cap'n Proto [2], FlatBuffers [111], Protocol Buffers [5] et Apache Thrift [91].

Dans cette section, nous présentons de manière détaillée ces différents formats de sérialisation binaire de données compatibles avec JSON. Notre objectif est de suivre leur évolution depuis 1983 (cf. Figure 4.1), d'analyser et de comparer les cas d'utilisation les plus populaires. Une présentation plus exhaustive est fournie en annexe.

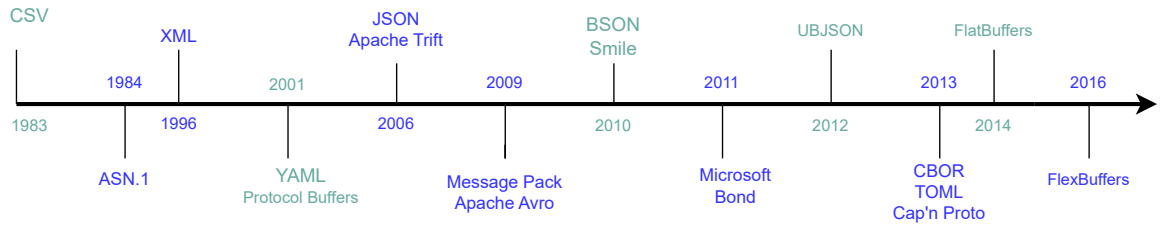


Figure 4.1: Chronologie des formats de données les plus populaires depuis 1983 [151]

4.4 Formats de serialisation binaire sans schéma

Dans ce type de format, les données sont simplement sérialisées en binaire sans aucune métadonnée supplémentaire détaillant comment ces données sont organisées. Lorsqu'un

programme souhaite désérialiser des données dans un format binaire sans schéma, il doit déjà connaître la structure des données à l'avance pour pouvoir les interpréter correctement. Nous présentons dans ce qui suit *CBOR*, *MessagePack* et *Smile*.

4.4.1 CBOR

CBOR [152] a été développé en réponse au besoin de disposer d'un format de données binaire plus compact et efficace que JSON.

```
A1                                      # map(1)
   6C                                   # text(12)
      676174657761795F636F6E66          # "gateway_conf"
   A3                                   # map(3)
      6A                                # text(10)
         676174657761795F4944           # "gateway_ID"
      70                                # text(16)
         33326533353831386632666238313262 # "32e35818f2fb812b"
      6E                                # text(14)
         7365727665725F61646472657373   # "server_address"
      71                                # text(17)
         6F7574696C732E706C69646F202E6E6574 # "outils.plido .net"
      67                                # text(7)
         73657276657273                 # "servers"
      81                                # array(1)
         A4                             # map(4)
            6E                          # text(14)
               7365727665725F61646472657373 # "server_address"
            70                          # text(16)
               6F7574696C732E706C69646F2E6E6574 # "outils.plido.net"
            6C                          # text(12)
               736572765F706F72745F7570 # "serv_port_up"
            19 06A4                     # unsigned(1700)
            6E                          # text(14)
               736572765F706F72745F646F776E # "serv_port_down"
            19 06A4                     # unsigned(1700)
            6C                          # text(12)
               736572765F656E61626C6564 # "serv_enabled"
            F5                          # primitive(21)
```

Listing 4.5: Sérialisation au format CBOR des données d'entrée JSON du listing A.1

Émergence du besoin : JSON est un format de données largement utilisé en raison de sa simplicité et de sa lisibilité. Cependant la représentation textuelle de JSON peut entraîner une surcharge en termes d'espace de stockage et de bande passante lorsqu'il est utilisé pour des applications nécessitant une efficacité accrue. Il est donc apparu le besoin de disposer d'une alternative binaire à JSON pour réduire la taille des données et améliorer les performances.

Développement de CBOR : Le format CBOR a été développé dans le cadre du groupe de travail CBOR de IETF, créé en 2010. Les participants au groupe de travail ont travaillé sur la définition d'un format binaire pour représenter les données de manière concise tout en maintenant une interopérabilité avec JSON.

Standardisation : La spécification officielle du format CBOR dans la RFC 7049, a été publiée en 2013. Ce document définit les règles précises d'encodage et de décodage des données CBOR, ainsi que la sémantique des types de données et des structures de données supportées.

Selon sa spécification, chaque élément CBOR qui contient un flot de données débute par un octet dont les trois premiers bits de poids fort indiquent le type major. Les cinq bits restants donnent les détails sur l'élément. Les types majors numérotés de 0 à 7 correspondent à la représentation en octet. Ces trois types sont décrits dans le tableau suivant :

Table 4.2: Une description des principaux types de CBOR, telle que spécifiée dans [152]

3 bits	Major Type	Description
000	0	Entier non signé entre 0 et $2^{64} - 1$
001	1	Entier négatif entre -1 et -2^{64}
010	2	Chaîne d'octets
011	3	Chaîne de texte UTF-8
100	4	Tableau
101	5	Une carte de paires d'éléments de données (ou objet en JSON)
110	6	Balise ("tag")
111	7	Les nombres à virgule flottante et les valeurs simples, ainsi que le code d'arrêt "break"

Avantages et adoption : Le format CBOR offre plusieurs avantages par rapport à JSON. Il permet une représentation binaire compacte des données, réduisant ainsi la taille des messages et des structures de données. CBOR prend en charge une variété de types de données, y compris les entiers de longueur variable, les nombres à virgule flottante, les chaînes de texte et les structures de données complexes. Grâce à sa compatibilité ascendante avec JSON, CBOR est devenu populaire dans divers domaines, tels que les protocoles de communication, les systèmes embarqués, l'IoT et les protocoles de sécurité.
Évolution continue : Depuis la publication de la spécification initiale, CBOR continue d'évoluer et de gagner en adoption. Des bibliothèques et des outils CBOR ont été développés pour faciliter l'encodage, le décodage et la manipulation des données CBOR dans différents langages de programmation.
Sécurité : CBOR inclut des mécanismes pour assurer l'intégrité et l'authenticité des données grâce à des signatures cryptographiques et des hachages.

Comme le montre la figure 4.2, aujourd'hui CBOR est considéré comme un format de données binaires efficace et polyvalent, offrant une alternative aux formats textuels comme JSON ou XML dans les scénarios où l'efficacité, la compacité et les performances sont des priorités.

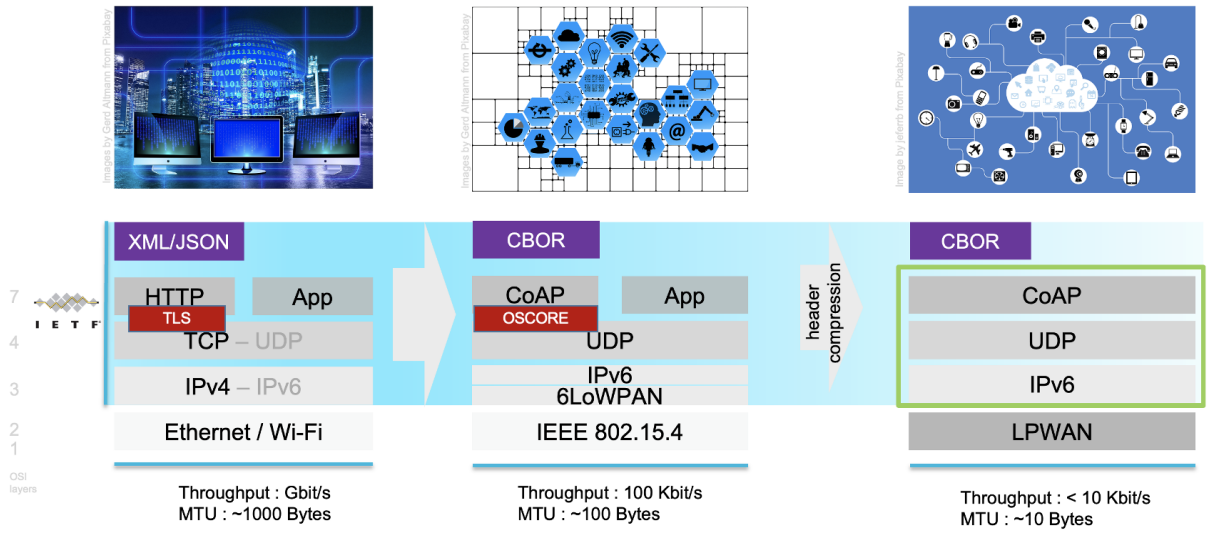


Figure 4.2: Utilisation du format CBOR dans les protocoles IoT

4.4.2 MessagePack

MessagePack [97] est un format de sérialisation binaire créé par Sadayuki Furuhashi en 2008. Il a été conçu comme une alternative compacte et rapide aux formats textuels tels que JSON et XML. Le format MessagePack utilise une représentation plus compacte pour encoder les données. Il prend en charge divers types de données, tels que les entiers, les flottants, les chaînes de caractères, les tableaux et les maps. Les données sont encodées de manière binaire, sans recours à la représentation textuelle, ce qui permet une taille réduite et des opérations de sérialisation et de désérialisation plus rapides.
Au fil du temps, MessagePack a gagné en popularité en raison de ses avantages en termes de vitesse et d'efficacité de stockage par rapport à JSON. De nombreux frameworks et bibliothèques l'ont intégré comme une option de sérialisation de données, et il est utilisé dans une variété d'applications, allant des systèmes distribués à grande échelle aux applications mobiles.

4.4.3 Smile

Le format de sérialisation SMILE (Simple Binary Encoding for Libraries and Extensions) [4] est un format binaire compact conçu pour représenter et échanger des données de manière efficace. Il a été créé par Chris Kasso en 2009 en tant qu'alternative binaire au format JSON. L'idée derrière SMILE était de fournir une sérialisation binaire qui conserve la simplicité et la lisibilité du JSON, tout en offrant une taille de données réduite et des performances accrues. Le format SMILE est basé sur une structure de données arborescente, où les données sont organisées sous forme de tableaux et d'objets.

Contrairement au JSON, qui est un format basé sur du texte, SMILE utilise une représentation binaire pour les différentes structures de données. Cela permet de réduire considérablement la taille des données sérialisées et d'améliorer les performances lors de l'encodage et du décodage des données. Le format utilise des codes d'opération spécifiques pour représenter différents types de données tels que les entiers, les flottants, les chaînes de caractères, les tableaux, les objets, les booléens, etc. Les données sont compressées et encodées de manière optimale pour minimiser la taille totale du message.
L'adoption de SMILE a été principalement limitée à certains domaines spécifiques, tels que les systèmes embarqués, les applications nécessitant une haute performance et une faible consommation de bande passante, et les environnements où la taille des données est critique. Il convient de noter que SMILE n'a pas atteint la même popularité que d'autres formats de sérialisation binaire tels que MessagePack ou BSON (Binary JSON). Ces formats ont été plus largement adoptés par la communauté des développeurs et ont bénéficié d'un plus grand soutien dans de nombreux langages de programmation.

4.4.4 Etude comparative

Le graphique 4.3 présente tous les formats de serialiasation binaires sans schéma compatibles avec le format JSON. On peut voir que CBOR et Message Pack sont les deux formats qui présentent une taille relativement petite.

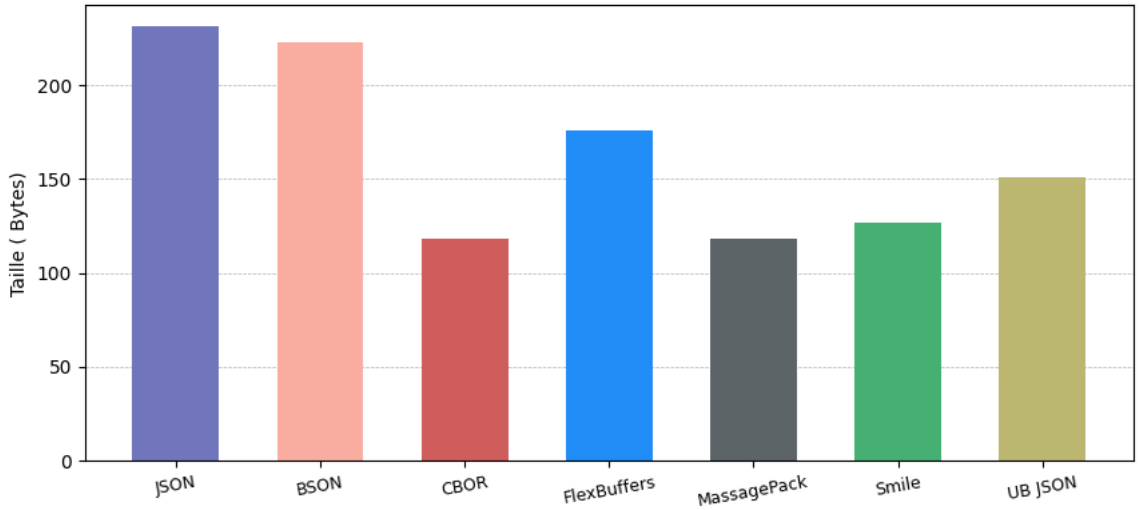


Figure 4.3: Comparaison entre les formats de sérialiasation binaires sans schéma et le format JSON [151]

Message Pack est un format très proche de CBOR, surtout dans ses objectifs et ses résultats. Il a même inspiré le développement de CBOR. Malheureusement, comme

la plupart de formats binaires sans schema, il souffre d'une absence d'interopérabilité. Plusieurs approches d'extension sont restées bloquées à cause de ce problème [24]. Et aujourd'hui le format n'est pas toujours normalisé par l'IETF.

De plus, CBOR est bien adapté à l'IoT en raison de sa compacité, de son efficacité, de son interopérabilité et de sa capacité à gérer l'évolution des données. Il offre une solution de sérialisation binaire efficace pour les appareils IoT avec des ressources limitées, contribuant ainsi à améliorer les performances, l'efficacité et la fiabilité entre les systèmes connectés de l'IoT.

En conclusion, MessagePack et CBOR sont deux formats de sérialisation binaire populaires, mais CBOR est souvent préféré dans les applications où l'interopérabilité, l'évolution des données et la normalisation sont des facteurs importants, tandis que MessagePack est souvent utilisé dans des cas où seules la compacité et la simplicité sont primordiales.

4.5 Format de sérialisation binaire avec schéma

Dans le format de sérialisation binaire avec schéma, la représentation d'objets ou de données se fait sous forme binaire, tout en incluant un schéma explicite pour décrire la structure des données. Le schéma permet aux programmes de lecture/désérialisation de comprendre comment interpréter correctement les données binaires et de les reconstituer en objets ou en données structurées. Le graphique 4.4 compare les tailles des différentes représentations binaires avec schéma des données.

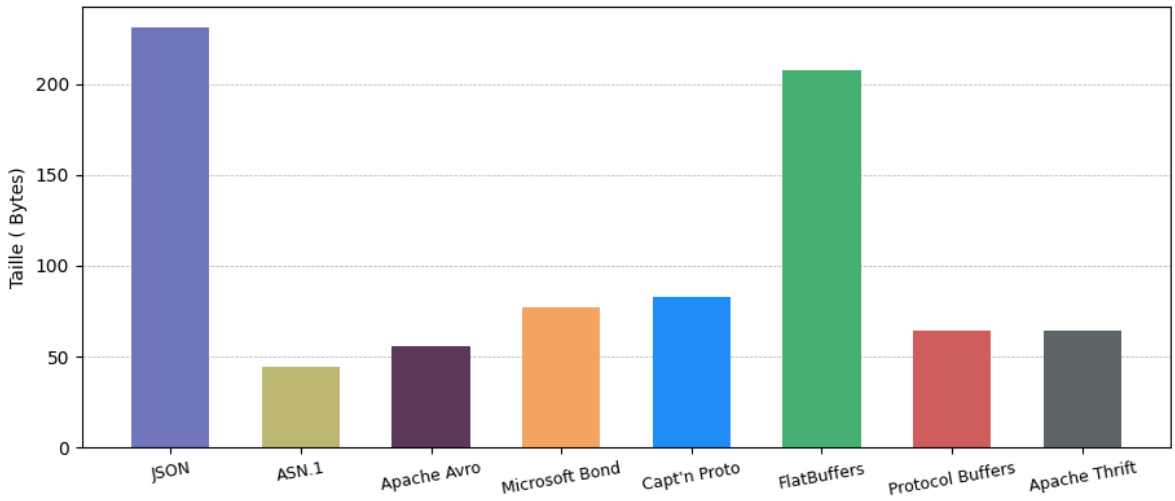


Figure 4.4: Comparaison entre les formats de serialiasation binaires avec schéma et le format JSON [151]

Bien que ces formats de sérialisation binaire avec schéma offrent une robustesse et une interopérabilité accrues, ils peuvent présentés aussi beaucoup d'inconvénients surtout dans un environnement contraint. Nous pouvons citer à cet égard les critères suivants:

1. **Taille des données** : La présence d'un schéma peut augmenter la taille des données sérialisées, car les métadonnées du schéma doivent être incluses aux données.
2. **Complexité** : La gestion d'un schéma peut ajouter de la complexité à la sérialisation et à la désérialisation des données, surtout lorsque les schémas sont très élaborés.
3. **Performance** : La lecture et la validation du schéma peuvent entraîner une surcharge de performance par rapport aux formats sans schéma.
4. **Dépendance du schéma** : L'utilisation d'un schéma signifie que les applications dépendent étroitement de la structure du schéma. Tout changement dans le schéma peut avoir des répercussions sur les applications existantes et nécessiter des ajustements supplémentaires.

4.6 Caractérisation des différents formats binaires

L'étude détaillée des formats de sérialisation binaire destinée aux messages DNS, nous a permis de tirer plusieurs enseignements essentiels:

Efficacité du format Binaire : Les formats de sérialisation binaire sont conçus pour être très efficaces en termes de taille de données et de vitesse de traitement. Ils minimisent la surcharge de données ajoutée par la sérialisation, ce qui est essentiel dans les applications où la bande passante et la vitesse de traitement sont critiques.
Interopérabilité : Certains formats de sérialisation binaire, tels que CBOR, Protocol Buffers (protobuf) et MessagePack, sont conçus pour offrir une interopérabilité entre différentes langages de programmation. Cela signifie que les données sérialisées dans un langage peuvent être désérialisées et utilisées dans un autre langage sans perte de structure ou de type.
Compactage de Données : Dans de nombreux scénarios, la sérialisation binaire permet de réduire considérablement la taille des données par rapport à d'autres formats, tels que le texte JSON, etc. Cela est particulièrement important dans les systèmes distribués où la transmission de données sur le réseau est coûteuse en termes de bande passante.
Choix du Format : Le choix du format binaire dépend des besoins spécifiques de chaque application/domaine. Il est important de prendre en compte des facteurs tels que la facilité d'utilisation, la performance, l'interopérabilité et les besoins en matière de sécurité lors de la sélection d'un format.

4.7 Conclusion

L'étude et l'analyse des formats de sérialisation binaire destinée à représenter les messages DNS de manière compacte et flexible, fait clairement apparaître CBOR comme le meilleur candidat. Le tableau 4.3 ci-dessous, montre par ailleurs que CBOR satisfait aux principales qualités qui pourraient être attendues d'un format de représentation de données :

Table 4.3: Etude comparative des différents formats de serialisation binaire compatible avec JSON

Formats	Standard	Robustesse	Compacité	Flexibilité	Interroperabilité
Formats binaire sans schéma					
JSON	✓	✓	✗	✗	✗
BSON	✗	✓	✗	✗	✗
CBOR	✓	✓	✓	✓	✓
FlexBuffers	✓	✓	✗	✓	✓
MessagePack	✗	✓	✓	✗	✗
Smile	✗	✗	✓	✗	✗
BSON	✗	✓	✗	✗	✓
Formats binaire avec schéma					
ASN.1	✓	✗	✗	✗	✗
Apache Avro	✗	✗	✗	✗	✗
Microsoft Bond	✗	✗	✗	✗	✗
Capt'n Proto	✗	✗	✗	✗	✗
FlatBuffers	✗	✗	✗	✗	✗
Protocol Buffers	✗	✓	✗	✓	✓

D'autre part CBOR dispose des atouts suivants :

- Les données peuvent être décodées sans description de schéma.
- La sérialisation est compacte.
- Le format s'adapte à la fois aux nœuds contraints et aux applications à fort volume.
- Le format prend en charge tous les types de données JSON.
- Le format est évolutif.

Dans ce chapitre qui conclue notre état de l'art, nous avons mené une étude des travaux existants qui nous conduit à faire l'hypothèse que les extensions performantes potentielles des fonctionnalités du DNS au delà de la résolution de noms de domaines, nécessite la définition d'un format d'encodage adaptable aux besoins des utilisateurs tout en restant pleinement compatible avec le format "wireformat" existant. Cette compatibilité avec le format original est essentielle pour préserver le DNS existant et amorcer une évolution efficace et sans accroc vers de nouvelles possibilités.

La prochaine section introduit une proposition de nouveau format nommé e-CBOR répondant aux exigences identifiées précédemment.

Part II

Mécanismes d'Extension du DNS

Chapter 5

e-CBOR : un format de représentation compacte et flexible

One size never fits all.

— **RFC 1925, Nr. 10**
(The Twelve Networking Truths)

Sommaire

5.1 Introduction

L'architecture Web et l'Internet ont rendu possible la recherche d'informations en ligne de toute nature, en utilisant des moteurs de recherche et des annuaires. L'extension importante de cette évolution technologique a été la possibilité de rechercher n'importe quel objet numérique grâce à son identifiant, quelle que soit sa localisation, en utilisant un simple URL.

Bien que cette identification des ressources sur le web repose sur le DNS, l'un des principaux obstacles qui freine l'extension de cette technologie à d'autres usages que la résolution de noms de domaines est l'absence d'un format de représentation et de transport des données plus flexible. Par exemple, le DNS peut être utilisé comme outils de stockages des informations de sécurité (clés, certificats, etc.) afin de renforcer la sécurité lors de l'identification des ressources sur le Web. De plus, l'extension du DNS peut jouer un rôle dans les environnements de calcul périphérique (edge computing), principalement en facilitant le routage efficace et la distribution de charge, ce qui peut améliorer la latence, réduire l'utilisation de la bande passante et améliorer les performances globales des applications et des services.

Dans ce chapitre, nous nous intéressons à la *réduction de l'impact du paradigme REST sur le DNS afin de permettre de futures applications efficaces du DNS à d'autres usages*. Pour cela, nous proposons une méthode, appelée Efficient Concise Binary Object Representation (e-CBOR) qui permet d'encoder les messages DNS grâce aux technologies de l'IoT. e-CBOR permet d'étendre le format de message DNS existant, de sorte qu'il peut être utilisé dans un environnement contraint en toute confiance sans modifier le format classique proposé par l'IETF. Dans e-CBOR, les messages de noms de domaine sont envoyés et reçus avec un format CBOR utilisant un encodage basé sur JSON qui permet une correspondance bidirectionnelle avec les encodages de transport traditionnels du système de noms de domaine. Cela garantit son interopérabilité tout en conservant son niveau actuel. Ainsi e-CBOR réduit de façon intéressante la taille des messages échangés lors d'une résolution DNS, notamment dans le cas des encapsulations induites par le modèle REST. e-CBOR vise également à améliorer la fiabilité et la sécurité DNS en facilitant l'introduction dans les Échanges DNS, des protocoles IoT tels que OSCORE, EDHOC (cf. figure 1.1(B)).

En traduisant les messages DNS en tables CBOR, le processus e-CBOR rend le message DNS plus compact et plus souple. Nos évaluations expérimentales montrent que pour les requêtes DNS et DNSSEC, nous obtenons une réduction du format de plus de 57 % par rapport au format original. Dans le cas des réponses nous obtenons jusqu'à 16 % de réduction de la taille des messages DNS/DNSSEC encodés dans le format traditionnel.

Dans ce qui suit nous rappelons le format classique d'un message DNS ainsi que sa représentation. Puis nous introduisons le nouveau format proposé et sa mise en œuvre. Nous illustrons e-CBOR à l'aide d'exemples d'application. L'analyse des performances

de e-CBOR comparé au DNS standard conclue ce chapitre.

5.2 Format classique des messages DNS

Nous décrivons dans cette section la structure d'un format de message DNS avec ses différents champs. Ensuite nous présentons les enregistrements (RR: ressource record en anglais) les plus utilisés par le DNS. Nous ajoutons enfin quelques exemples de représentation des données sur ce format. Ces notions permettront de mieux comprendre notre méthode de représentation des messages DNS qui sera proposée par la suite.

Definition 5.2.1. Un format de données est une structure ou une représentation spécifique utilisée pour organiser, stocker ou échanger des informations. Il définit la manière dont les données sont agencées, encodées et présentées pour qu'elles soient compréhensibles et utilisables par les systèmes informatiques ou les personnes. Les formats de données sont couramment utilisés pour décrire la disposition, la syntaxe, et parfois même le contenu des données, ce qui permet une interprétation cohérente et efficace des informations par les applications et les utilisateurs.

5.2.1 Structure d'un message DNS

Le RFC 1035 [103] a défini un format général qui permet à un client de construire sa requête DNS et au serveur faisant autorité d'envoyer une réponse DNS compréhensible par ce client. Ce format est illustré dans la Fig.5.1.

Cette figure 5.1 montre un message DNS composé de cinq sections : (i) Header (`ID`, flags, count) ; (ii) Question ; (iii) Answer ; (iv) Authority(RR) ; (v)Additional. La section `Header` contient des informations permettant de faire correspondre la réponse à la requête, la section `Question` contient la requête, la section `Answer` contient la réponse, `Authority(RR)` contient le RR et la section `Additional` contient les informations supplémentaires.

5.2.2 DNS Headers : en-tête

Les figures 5.2 et 5.3 illustrent respectivement un exemple de l'entête appelée Header d'une requête et d'une réponse DNS sur le nom de domaine `google.com`. Elle est constituée des champs suivants :

- `ID` - Cet élément représente un identifiant de 16 bits, qui peut être configuré selon les besoins. Lorsque le serveur renvoie sa réponse, il utilise le même identifiant que celui que nous avons spécifié ici. Cette fonction peut être utile pour associer les réponses aux requêtes, notamment si nous effectuons plusieurs requêtes en même temps. Cette correspondance est particulièrement pertinente car nous utilisons le protocole DNS via UDP.
- Flags :
  - `QR` : Il prend la valeur 0 en cas de demande et 1 en cas de réponse.

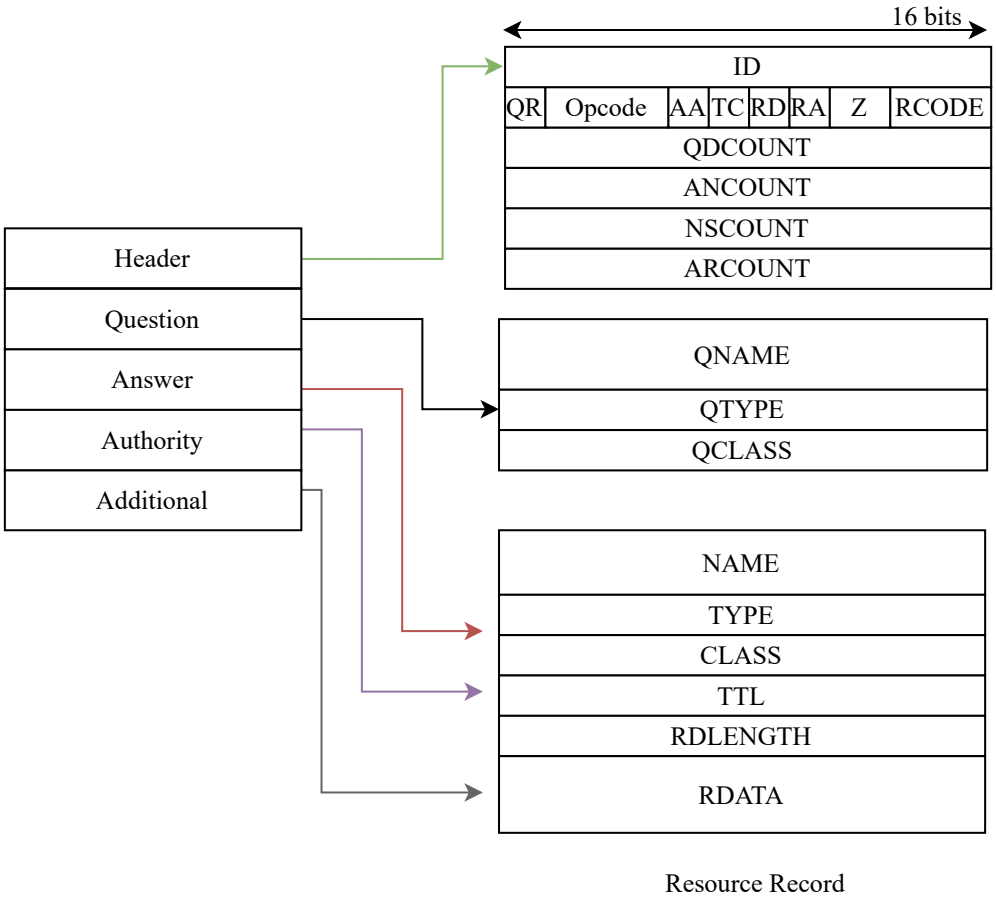


Figure 5.1: Format classique d'un message DNS[103]

- `Opcode` (Code Opération) : Ce champ indique le type de requête que nous souhaitons effectuer. ici nous traitons uniquement sur la valeur 0, qui correspond à une requête standard. Pour d'autres types de requêtes, il est possible de consulter la page 26 de la RFC 1035.
- `AA` (Bit d'Autorité Réponse) : Dans les requêtes, ce bit n'a pas de signification et doit être réglé à 0. Dans les réponses, il indique si le serveur qui répond est une autorité pour le domaine demandé ou non.
- `TC` (Bit de Troncation) : Ce bit spécifie si le message a été tronqué en raison de sa longueur excessive. Si c'est le cas, nous pouvons recevoir des messages supplémentaires pour obtenir le reste des données.
- `RD` (Bit de Récursivité Demandée) : Dans de nombreuses requêtes DNS, le premier serveur que nous interrogeons peut ne pas avoir l'adresse IP recherchée, mais il peut connaître un autre serveur de noms qui la possède. En définissant ce bit dans une requête, nous indiquons si nous souhaitons que le serveur transmette la requête au serveur de noms suivant de manière récursive jusqu'à ce qu'il trouve une réponse. Si le bit n'est pas défini, le serveur nous indiquera simplement le prochain serveur de noms à consulter. Dans l'en-tête de réponse,

ce bit est défini de la même manière qu'il l'était dans l'en-tête de la requête.
  - `RA` (Bit de Récursion Disponible) : Ce bit n'a pas de signification dans une requête et doit être réglé à 0. Dans une réponse, il indique si les requêtes récursives (telles que décrites ci-dessus) sont prises en charge ou non par le serveur de noms. Tous les serveurs de noms ne prennent pas en charge les requêtes récursives.
  - `Z` (Bit Réservé) : Ce champ est réservé pour une utilisation future et doit toujours être réglé sur 0.
  - `RCODE` (Code de Résultat) : Ce champ n'a pas de signification dans les requêtes et s'applique uniquement aux réponses. Il peut prendre l'une des valeurs suivantes :
    * 0 : Pas d'erreur.
    * 1 : Erreur de format (notre requête a un problème de formatage).
    * 2 : Panne du serveur (notre demande est correcte, mais il y a un problème avec le serveur).
    * 3 : Erreur de nom (valable uniquement pour les serveurs de noms faisant autorité, indiquant que le domaine demandé n'existe pas).
    * 4 : Non implémenté (le serveur ne prend pas en charge notre opération).
    * 5 : Refusé (le serveur pourrait répondre à votre demande, mais il refuse, peut-être en raison d'une autorisation insuffisante ou d'autres raisons).
    * 6-15 : Réservé pour une utilisation future.
- `QDCount` (Nombre de Sections de Questions) : Indique combien de sections de questions sont présentes (16 bits non signés).
- `ANCount` (Nombre de Sections de Réponses) : Indique combien de sections de réponse sont présentes (16 bits non signés).
- `NSCount` : Nombre d'enregistrements de serveurs de noms présents dans la section des enregistrements d'autorité (16 bits non signés).
- `ARCount` : Indique combien d'enregistrements de ressources sont présents dans la section des ressources supplémentaires (16 bits non signés).

5.2.3 DNS Questions : requête

Nous avons spécifié dans l'en-tête de notre example sur `google.com` que nous envoyons 1 question (dans QDCount). Chaque question correspond donc à une demande de nom et elle porte sur enregistrement de type A (IPv4).

La section des questions est beaucoup plus simple que la section d'en-tête. Le tableau donné par la RFC (encore une fois, chaque colonne fait 1 bit) comporte plus précisément les informations suivantes :

- QName correspond au nom de domaine que nous interrogeons.
- QType indique le type d'enregistrement que nous recherchons, comme A, CNAME, NS, ou d'autres types de recherches DNS similaires. Dans notre cas, nous effectuons une recherche de type A, ce qui est désigné par QType = 1 (conformément au RFC 1035, page 11).
- QClass permet de spécifier la classe de la requête, que ce soit pour Internet, CSNET, Chaosnet ou Hesoid. Étant donné que nous effectuons une recherche sur Internet

dans notre exemple, nous utilisons la valeur QClass = 1 (conformément au RFC 1035, page 12).

```
00000000 b0 0b 01 20 00 01 00 00 00 00 00 01 06 67 6f 6f |... .........goo|
00000010 67 6c 65 03 63 6f 6d 00 00 01 00 01 00 00 29 10 |gle.com.......).|
00000020 00 00 00 00 00 00 0c 00 0a 00 08 ff 32 fa f6 bc |............2...|
00000030 a1 41 f0                                       |.A.             |
```

Figure 5.2: Exemple d'encodage en hexadécimal d'une requête DNS[103]

Table 5.1: En-tête d'une requête DNS de type A sur le nom de domaine `google.com`

Champs	Sous-champs	Valeurs	Description
ID	`ID`	`0x0008`	La réponse doit avoir l'ID = `0x0008`
Flags	`QR`	`0`	Il s'agit d'une requête
	`OPCODE`	`0`	Requête standard
	`TC`	`0`	La requête est non tronquée
	`RD`	`1`	Récursion demandée
	`RA`	`1`	Non significatif pour la requête
	`Z`	`0`	Réservé
	`RCODE`	`0`	Non significatif pour la requête
QDCOUNT		`0x0001`	Pas de requête multiple
ANCOUNT		`0x0000`	Aucune réponse
NSCOUNT		`0x0000`	Aucun serveur DNS faisant autorité
ARCOUNT		`0x0001`	Enregistrement supplémentaire

Table 5.2: Exemple de la section `Question` d'une requête DNS de type A sur le nom de domaine `google.com`

Données (Question et Additionnal)	Description
`0x06`	Label de longueur 6
`0x676f6f676c65`	Chaîne de `google`
`0x03`	Label de longueur 3
`0x636f6d`	chaîne de `com`
`0x00`	Fin du nom de domaine
`0x0001`	Enregistrement de Type A
`0x0001`	Class IN(Internet address)
`0x2910...f32faf6bca141f0`	Enregistrement supplémentaire

5.2.4 DNS Answers : réponse

La section Answers contient des enregistrements de réponse aux questions. Chaque entrée comprend :

- Name : Nom de domaine auquel la réponse se réfère.
- Type : Type d'enregistrement DNS (A, AAAA, CNAME, etc.).
- Class : Classe de l'enregistrement (IN pour Internet).
- TTL : Durée de validité de l'enregistrement.
- RDlength : Longueur des données de l'enregistrement.
- Rdata : Données de l'enregistrement.

```
00000000 b0 0b 81 80 00 01 00 01 00 00 00 01 06 67 6f 6f |.............goo|
00000010 67 6c 65 03 63 6f 6d 00 00 01 00 01 c0 0c 00 01 |gle.com.........|
00000020 00 01 00 00 01 2c 00 04 8e fa b3 4e 00 00 29 02 |.....,.....N..).|
00000030 00 00 00 00 00 00 00                           |.......         |
```

Figure 5.3: Exemple d'encodage en hexadécimal d'une réponse DNS[103]

Table 5.3: En-tête d'une reponse DNS de type A sur le nom de domaine `google.com`

Champs	Sous-champs	Valeurs	Description
ID	`ID`	`0xb00b`	La réponse à la requête `0xb00b`
Flags	`QR`	`1`	Il s'agit d'une reponse
	`OPCODE`	`0`	Requête standard
	`TC`	`0`	La requête est non tronquée
	`RD`	`1`	Récursion demandée
	`RA`	`1`	Le serveur DNS peut gerer la recursivité
	`Z`	`0`	Réservé
	`RCODE`	`0`	Non significatif pour la requête
QDCOUNT		`0x0001`	Contient une seule requête
ANCOUNT		`0x0001`	Contient une réponse
NSCOUNT		`0x0000`	Aucun serveur DNS faisant autorité
ARCOUNT		`0x0001`	Aucun enregistrement supplémentaire

5.2.5 Limitation du format DNS

Les messages du DNS sont optimaux en taille grâce aux pointeurs arrière évitant de répéter plusieurs fois les mêmes séquences textuelles, mais l'ajout de nouveaux champs, même s'ils répondent à la même syntaxe que les autres est relativement pénible. Un exemple est l'utilisation du champ TXT pour supporter les évolutions, comme nous l'avons vu

Table 5.4: Exemple de la section `Question` d'une réponse DNS de type A sur le nom de domaine `google.com`

Données (Response et Additionnal)	Description
`0xc`	Le nom est un pointeur
`0x00c`	Pointeur est sur le nom à l'offset 0x00c
`0x0001`	Réponse à la requête de Type A (host address)
`0x0001`	Réponse est class IN (Internt address)
`0x0000012c`	Reponse valide pour 300 secondes
`0x0004`	Adresse IP de longueur 4 octets (IPv4)
`0x8efab34e`	Adresse IPv4 est 142.250.179.78
`0x00002902...00`	Enregistrement supplémentaire

dans les paragraphes précédents. La syntaxe utilisée dans ce champ n'est pas fortement standardisée, ce qui rend plus compliquée l'interopérabilité et surtout, elle ne permet pas de bénéficier des techniques de compression de texte, alourdissant de fait le poids des messages.

5.3 Proposition d'un nouveau format de message DNS

Cette section décrit les différentes sections du nouveau format, illustré dans les Fig 5.4, et Fig 5.9, ainsi que la manière dont elles sont codées en CBOR.

5.3.1 Structure de la section Header

La section d'en-tête ne contient plus que les champs `ID` et `FLAG`. Les champs `ARCOUNT`, `NSCOUNT`, `QDCOUNT` et `ANCOUNT` sont omis dans les requêtes DNS et certains sont utilisés dans la réponse renvoyée par le serveur DNS.
Pour coder ces informations en CBOR, le début du message doit contenir un octet représentant le nombre de sections DNS présentes. Pour une requête, l'en-tête est codé comme suit : `ID` (entier non signé codé sur 3 octets), `Flags` (entier non signé codé sur 3 octets). Pour une réponse, ces champs sont complétés par `ANCOUNT` (entier non signé de 1 octet indiquant le nombre de RR dans la section réponse), `NSCOUNT` (entier non signé de 1 octet représentant le nombre de RR dans la section enregistrement d'autorité) et `ARCOUNT` (entier non signé de 1 octet représentant le nombre de RR dans la section enregistrements supplémentaires).
Ce codage permet de passer d'un en-tête dont tous les éléments sont codés sur 12 octets à un nouvel en-tête dont les éléments sont codés sur seulement 3 octets pour une requête et 10 octets pour la réponse.

5.3.2 Structure de la section Question

Dans le message DNS classique, cette section contient `QNAME`, `QTYPE` et `QCLASS`. Nous utilisons un codage CBOR avec un tableau à deux entrées contenant respectivement `QNAME`

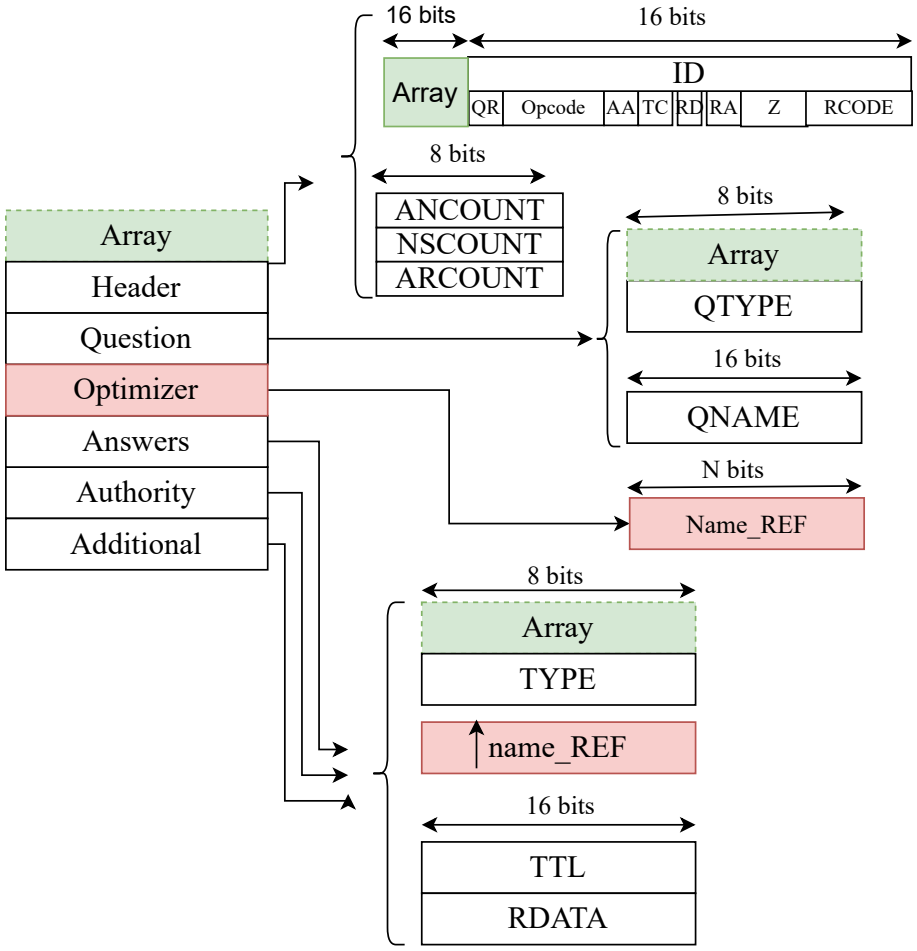


Figure 5.4: Format d'un message DNS utilisant la méthode e-CBOR

et `QTYPE` dans l'ordre inverse pour rester cohérent avec la structure CBOR. Contrairement à la structure DNS où la classe se trouve au milieu de la structure, dans e-CBOR, la classe est le dernier élément du tableau. Nous omettons "class" car sa valeur est toujours "IN". Nous allouons également 1 octet pour coder le nombre d'éléments dans le tableau CBOR, 1 octet pour `QTYPE` et 2 octets pour `QNAME`. Comme l'illustre la figure 5.9, notre approche nous permet de représenter les données d'une requête DNS en seulement deux champs, le premier étant `QTYPE` (entier non signé, 1 octet) et le second étant `QNAME` (chaîne de caractères sur 2 octets). Cela nous permet de réduire le nombre d'octets nécessaires pour représenter une requête DNS.

5.3.3 Description de la méthode e-CBOR

Pour réduire la longueur des messages DNS, "wireformat" utilise des pointeurs dans le message pour faire référence à un sous-domaine précédemment utilisé. Cette méthode de compression ne peut pas être utilisée directement dans le contexte CBOR.

Nous proposons d'introduire une structure de Type Map (ou Type 100) de CBOR qui joue un rôle similaire. La clé de map est un nombre et la valeur est soit une chaîne, soit un tableau contenant une chaîne suivie d'un entier faisant référence à une clé précédente. De cette façon, nous pouvons avoir une méthode de compression similaire, et comme les clés ont de petites valeurs, cela conduit généralement à une meilleure compression que le message original où les pointeurs sont sur 2 octets. Nous choisissons de placer cette structure après la requête, car elle n'est nécessaire que lorsqu'un nom apparaît à la fois dans la requête et dans la réponse.

L'un des avantages d'e-CBOR est la possibilité de revenir au message DNS classique après l'encodage.Les figures 5.5 et 5.6 illustrent l'arbre `name_ref` construit à partir des informations répétées dans une réponse DNS lors de la résolution du nom de domaine `fazenda.gov.br` sur l'enregistrement de ressource de type CNAME. Ce `name_ref` peut également prendre la forme suivante :

```
1  {    0: 'gov.br',
2       1: ['fazenda', 0],
3       2: ['serpro', 0],
4       3: ['bsa1', 2],
5       4: ['hostmaster', 2]
6  }
```

Listing 5.1: Representation de l'optimiseur des messages DNS : `name_ref`

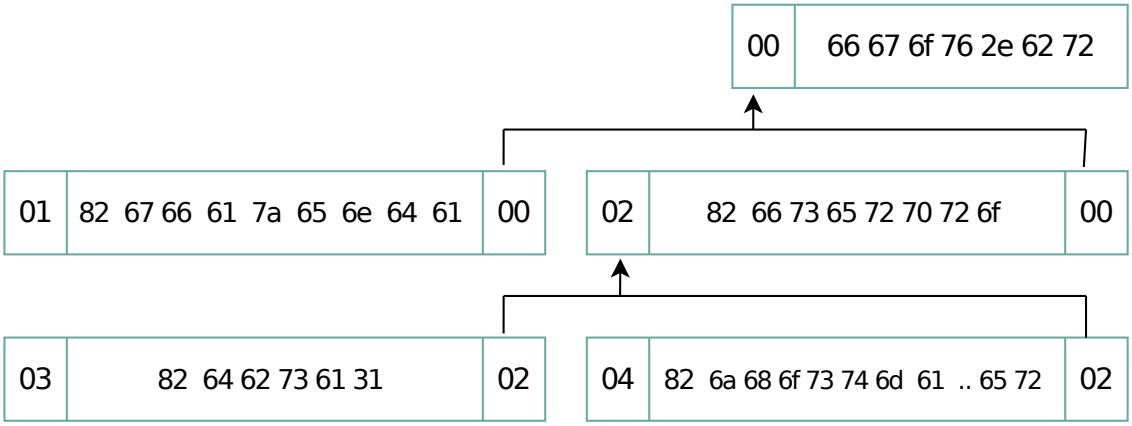


Figure 5.5: Exemple d'arbre e-CBOR pour une réponse DNS

Lors des échanges DNS, e-CBOR analyse l'ensemble du message et stocke les informations répétées dans un arbre, chaque nœud étant constitué d'une paire clé-valeur.

```
00000000  e7 e2 81 a0 00 01 00 00  00 01 00 01 07 66 61 7a  |.............faz |
00000010  65 6e 64 61 03 67 6f 76  02 62 72 00 00 05 00 01  |enda.gov.br......|
00000020  c0 0c 00 06 00 01 00 00  07 08 00 2f 04 62 73 61  |............/.bsa|
00000030  31 06 73 65 72 70 72 6f  c0 14 0a 68 6f 73 74 6d  |1.serpro...hostm|
00000040  61 73 74 65 72 c0 31 78  86 fa 39 00 00 38 40 00  |aster.1x..9..8@.|
00000050  00 0e 10 00 1b af 80 00  00 0e 10 00 00 29 02 00  |.............)..|
00000060  00 00 00 00 00 00                                 |......          |
```

a) Classic DNS Format

```
00000000  87 19 e7 e2 19 81 a0 82  05 01 a5 00 66 67 6f 76  |............fgov|
00000010  2e 62 72 01 82 67 66 61  7a 65 6e 64 61 00 02 82  |.br..gfazenda...|
00000020  66 73 65 72 70 72 6f 00  03 82 64 62 73 61 31 02  |fserpro...dbsa1.|
00000030  04 82 6a 68 6f 73 74 6d  61 73 74 65 72 02 80 81  |..jhostmaster...|
00000040  8a 06 01 03 04 1a 78 86  fa 39 19 38 40 19 0e 10  |......x..9.8@...|
00000050  1a 00 1b af 80 19 0e 10  19 07 08 81 4b 00 00 29  |............K..)|
00000060  02 00 00 00 00 00 00 00                           |........        |
```

b) DNS Format using e-CBOR

Figure 5.6: Relation entre les éléments du `name_ref` dans une réponse DNS

5.3.4 Transformation e-CBOR

Algorithme 1 : Algorithme de traduction de "wireformat" par e-CBOR

Entrées : Un "wireformat" $W = (H, Q, R, A, S)$
Output : Un format avec une structure tableau CBOR $W' = (H', Q', N, R', A', S')$
$H \longleftarrow [ID, Flag, nscount, ancount, arcount, adcount]$ // *Header section* ;
$Q \longleftarrow [QName, QType, QClass]$ // *Question section* ;
$R \longleftarrow [Name, Class, TTL, Rdlength, Rdata]$ // *Answer section* ;
$A \longleftarrow [R]$ // *Authority section* ;
$S \longleftarrow [R]$ // *Additionnal section* ;
tant que *true* **faire**
  si W *est vide* **alors**
    sortir de la boucle avec échec (aucune traduction n'existe)
  fin
  remplacer les champs d'en-tête nscount, ancount, arcount et adcount par une constante *none*;
  $H' \longleftarrow [ID, Flag, none]$;
  $N \longleftarrow$structure map CBOR pour stocker les informations répétitives dans W ;
  si W *est une requête* **alors**
    $Q' \longleftarrow [QType, QName]$;
    $W' \longleftarrow [H', Q', None]$;
  sinon
    $N \longleftarrow \{v :' label'\}$ // *N associe un entier aux étiquettes répétitives*;
    $Q' \longleftarrow [QType, v]$;
    $R' \longleftarrow [Type, v, TTL, Rdata]$;
    $W' \longleftarrow [H', Q', N, R', None, None]$;
  fin
fin

5.3.5 Format des enregistrements (RR)

Pour coder les enregistrements de ressources DNS (`RR`), nous utilisons la technique de représentation des données décrite dans la section précédente. En général, un enregistrement de ressources DNS se compose d'un nom de domaine (chaîne de 16 bits), d'un type (entier non signé de 16 bits), d'une classe (entier non signé de 16 bits), d'un TTL (entier non signé de 32 bits), de `RDLENGTH` (entier non signé de 16 bits) et de `RDATA` (chaîne d'octets de longueur variable).
Le codage e-CBOR s'effectue en deux étapes principales : nous commençons par la factorisation du nom de domaine, qui est ensuite stocké dans la variable `name_ref`. Cette variable stocke un ensemble de paires clé-valeur où la valeur est le nom de domaine et la clé un entier non signé. Cette technique permet de représenter le nom de domaine, qui est une chaîne de caractères, par un nombre. En remplaçant une chaîne de caractères par un nombre, on passe de 16 bits à 8 bits. Après avoir factorisé le nom de domaine, une table CBOR à 4 entrées est utilisée pour coder l'enregistrement de la ressource comme suit : le premier élément de la table contient le type (codé sur 8 bits au lieu de 16), le deuxième élément est le nombre qui remplace désormais le nom de domaine (codé sur 8 bits au lieu de 16), le troisième élément est le `TTL` (codé sur 8 bits au lieu de 16), le quatrième élément est le `RDATA` (la taille reste variable en fonction du `RR`). Les champs `CLASS` et `RDLENGTH` sont placés à la fin du tableau et restent facultatifs.

5.4 Exemples d'application de e-CBOR

5.4.1 Requête DNS

Dans cette section, nous illustrons l'utilisation de e-CBOR pour représenter les requêtes DNS, avec comme exemple une requête de type A pour le domaine `force.com`. La figure 5.7 compare e-CBOR et la représentation classique RFC 1035 :

```
00000000  c8 4e 01 20 00 01 00 00  00 00 00 01 05 66 6f 72  |.N. .........for|
00000010  63 65 03 63 6f 6d 00 00  01 00 01 00 00 29 10 00  |ce.com.......)..|
00000020  00 00 00 00 00 0c 00 0a  00 08 32 98 c6 46 f0 06  |..........2..F..|
00000030  c4 40                                             |.@              |
```

a) Classic DNS Format

```
00000000  83 19 c8 4e 19 01 20 82  01 6a 66 6f 72 63 65 2e  |...N.. ..jforce.|
00000010  63 6f 6d 2e                                       |com.            |
```

b) DNS Format using e-CBOR

Figure 5.7: (a) Représentation classique et (b) représentation e-CBOR d'une requête DNS

La figure 5.7 (a) représente une requête DNS classique composée d'un en-tête et de champs non vides (requêtes, enregistrements supplémentaires). L'en-tête se compose d'un `ID` (`0xc84e`), d'un indicateur (`0x0120`) et d'une séquence de compteurs (`Questions`, Answer `RRs`, Authority `RRs`, Additional `RRs`).

La figure 5.7(b) montre la représentation e-CBOR de cette requête DNS à l'aide d'un tableau CBOR. Dans l'en-tête, nous omettons les compteurs car ils peuvent être facilement dérivés du contenu des champs non vides présents dans la requête.
Dans la section de la requête, qui consiste en un nom (`0x 05 66 6f 72 63 65 03 63 6f 6d`), un type (`0x0001`) et une classe (`0x0001`), nous omettons également la sous-section "class" car sa valeur est toujours "IN". La section des réponses est évidemment vide puisqu'il s'agit d'une requête et qu'elle ne sera pas représentée.
La section Additional `RRs` peut, selon le client, fournir un mécanisme rétro-compatible pour étendre les messages DNS comme expliqué dans le RFC 6891 [37]. Elle contient généralement un `OPT` RR qui ne contient aucune donnée DNS. Nous l'omettons également.

La requête DNS peut alors s'écrire de la façon suivante:

```
[ 0xc84e,              # ID
  0x0120,              # Flag
    ["force.com", 1], # QNAME and QTYPE
    None               # Missing fields
 ]
```

Listing 5.2: Requête DNS prête à la transformation

Il devient alors possible de dériver la représentation suivante en utilisant e-CBOR :

```

83                            # array(3)
   19 C84E                    # unsigned(51278)
   19 0120                    # unsigned(288)
   82                         # array(2)
      01                      # unsigned(1)
      6A                      # bytes(10)
        666F7263652E636F6D2E # "force.com."
```

Listing 5.3: Requête DNS utilisant e-CBOR

Le tableau 5.5 montre que nous avons ainsi réduit la taille d'une requête DNS de 40 octets dans son format natif à 20 octets avec cette nouvelle représentation. En conséquence, nous pouvons obtenir un pourcentage de diminution de 50% pour ce scénario. Une requête DNS dans e-CBOR est donc un tableau à trois éléments composé d'un `ID`, d'un `Flag` et d'un sous-réseau à deux éléments composé d'un `QTYPE` et d'un `QNAME`.

5.4.2 Réponse DNS

Poursuivant l'exemple de la section précédente, examinons maintenant la représentation e-CBOR de la réponse à une requête DNS de type A pour le nom de domaine `force.com`. Cette réponse dans le DNS classique est illustrée dans la Fig. 5.8.

Table 5.5: Transformation et évaluation d'une requête DNS avec e-CBOR

Champs	Sous-champs	Valeurs		Taille(bytes)		
		DNS	e-CBOR	DNS	e-CBOR	δ
				($T1$)	($T2$)	($T1 - T2$)
	-	-	0x83	-	1	-1
HEADER	ID	0xc84e	0x19c84e	2	3	-1
	FLAG	0x81a0	0x1981a0	2	3	-1
	QDCOUNT	0x0001	-	2	-	+2
	ANCOUNT	0x0008	-	2	-	+2
	NSCOUNT	0x0000	-	2	-	+2
	ARCOUNT	0x0001	-	2	-	+2
QUESTION	-	-	0x82	-	1	-1
	QNAME	0x0566..6f6d00	0x6a666f.. 2e	11	11	0
	QTYPE	0x0001	0x81	2	1	+1
	QCLASS	0x0001	-	2	-	+2
ADDITIONAL		0x290200..00	-	23	-	+23
TOTAL				40	20	+20

(+ k) : signifie que notre approche a permis de réduire la taille d'un champ de k octets.

Lorsque nous l'examinons, nous remarquons que `force.com`, codé sur 10 octets, apparaît dans les huit réponses de la section des réponses, ainsi que dans la section des requêtes. Pour réduire la redondance et donc la taille du message, nous utilisons e-CBOR pour représenter le nom `force.com` comme suit :
La réponse à la requête qui en résulte, encodée avec e-CBOR, est la suivante :

```
[ 0xc84e,             # ID transaction
  0x81a0,             # Flags
  [1, 0],             # Query section
  {0: 'force.com'},   # name_ref
  [[1, 0, 0x17012384 , 0x0000001e], # rr1
   [1, 0, 0xb81f0382 , 0x0000001e], # rr2
   [1, 0, 0x17016a85 , 0x0000001e], # rr3
   [1, 0, 0x17016382 , 0x0000001e], # rr4
   [1, 0, 0x686d0b81 , 0x0000001e], # rr5
   [1, 0, 0xb819b384 , 0x0000001e], # rr6
   [1, 0, 0x686d0a81 , 0x0000001e], # rr7
   [1, 0, 0xb81f0a85 , 0x0000001e]],# rr8
  [],                  # Authority
  [0x2902000000000000]  # Additional
]
```

Listing 5.4: Réponse DNS utilisant e-CBOR

Le tableau 5.6 fournit une évaluation détaillée de cette transformation. Nous comparons chaque champ, sous-champ et leur représentation hexadécimale en utilisant le DNS classique et notre proposition e-CBOR, ainsi que la taille en octets des deux représentations. Enfin, nous comparons la différence (δ) de leurs tailles en octets pour identifier le pourcentage de gain ou de perte dans chaque cas. Voici un bref résumé du tableau :

- `0x87` : Utilisé pour indiquer un tableau CBOR de longueur 7.
- `Section d'en-tête` : `ID` et `FLAG` sont codés sur 3 octets, `QDCOUNT` n'est pas représenté, et `ANCOUNT`, `NSCOUNT` et `ARCOUNT` sont codés sur 1 octet.

```
▶ Domain Name System (response)
    Transaction ID: 0xc84e
  ▶ Flags: 0x81a0 Standard query response, No error
    Questions: 1
    Answer RRs: 8
    Authority RRs: 0
    Additional RRs: 1
  ▶ Queries
  ▼ Answers
    ▼ force.com: type A, class IN, addr 23.1.35.132
        Name: force.com
        Type: A(Host Address) (1)
        Class: IN (0x0001)
        Time to live: 30 (30 seconds)
        Data length: 4
        Address: 23.1.35.132
    ▶ force.com: type A, class IN, addr 184.31.3.130
    ▶ force.com: type A, class IN, addr 23.1.106.133
    ▶ force.com: type A, class IN, addr 23.1.99.130
    ▶ force.com: type A, class IN, addr 104.109.11.129
    ▶ force.com: type A, class IN, addr 184.25.179.132
    ▶ force.com: type A, class IN, addr 104.109.10.129
    ▶ force.com: type A, class IN, addr 184.31.10.133
  ▶ Additional records
      <Root>: type OPT
    [Request In: 129]
    [Time: 0.023158000 seconds]
```

Figure 5.8: Représentation classique d'une réponse DNS (Wireshark)

- `Section Question` : 1 octet est utilisé pour définir le tableau, 1 octet pour coder `QTYPE`, `QNAME` est codé en fonction des éléments de `name_ref`.
- `Réponses`, `Autorité` et `Autres` sections : 1 octet au début pour coder chaque sous-tableau, 1 octet pour `TYPE`, 1 octet pour le pointeur `name_ref`, 1 octet pour `TTL` ; `RDATA` variable de taille.

Dans cette nouvelle représentation, `0x87` (1 octet) est ajouté au début du message pour coder les 7 éléments du tableau, `0x82` (1 octet) également pour coder les deux éléments de la section de requête.
En examinant l'exemple décrit dans le tableau 5.6, nous pouvons voir que nous sommes passés d'un format DNS classique d'une taille de $T1 = 166$ octets à un format relativement plus petit d'une taille de $T2 = 118$ octets. Cela nous donne une différence de taille de $\delta = 48$, ce qui représente un gain de 28%.

Table 5.6: Transformation et évaluation du format de réponse DNS avec e-CBOR

Champs	Sous-champs	Valeur		Taille(bytes)		
		DNS	e-CBOR	DNS	e-CBOR	δ
				$(T1)$	$(T2)$	$(T1 - T2)$
	-	-	0x87	-	1	−1
HEADER	ID	0xc84e	0x19c84e	2	3	−1
	FLAG	0x81a0	0x1981a0	2	3	−1
	QDCOUNT	0x0001	-	2	-	+2
	ANCOUNT	0x0008	0x88	2	1	+1
	NSCOUNT	0x0000	0x80	2	1	+1
	ARCOUNT	0x0001	0x81	2	1	+1
QUESTION	-	-	0x82	-	1	−1
	QNAME	0x0566..6f6d00	0x00	11	1	+10
	QTYPE	0x0001	0x81	2	2	0
	QCLASS	0x0001	-	2	-	+2
name_ref		-	0x00a100696667..6d-		12	−12
ANSWER	-	-	0x84	-	$1 \times 8 = 8$	−8
	NAME	0xc0c0,..	0x00,..	$2 \times 8 = 16$	$1 \times 8 = 8$	+8
	TYPE	0x0001, ..	0x81	$2 \times 8 = 16$	$1 \times 8 = 8$	+8
	CLASS	0x0001, ..	-	$2 \times 8 = 16$	-	+16
	TTL	0x0000001e, ..	0x1e,..	$4 \times 8 = 32$	$1 \times 8 = 8$	+24
	RDLENGTH	0x0004, ..	-	$2 \times 8 = 16$	-	+16
	RDATA	0x17012384, ..	0x441701238418,..	$4 \times 8 = 32$	$6 \times 8 = 48$	−16
AUTHORITY				0	0	0
ADDITIONAL		0x290200..00	0x4b000029020..00	11	12	−1
TOTAL				166	118	+48

(+ k) : signifie que notre approche a permis de réduire la taille d'un champ de k octets.

5.5 Évaluation des performances

Cette section présente les résultats de l'évaluation des performances de notre solution. Le code source, ainsi que les fichiers contenant des exemples de mise en œuvre et l'implémentation de notre approche, sont disponibles ici : https://github.com/a20lemog/e-CBOR.

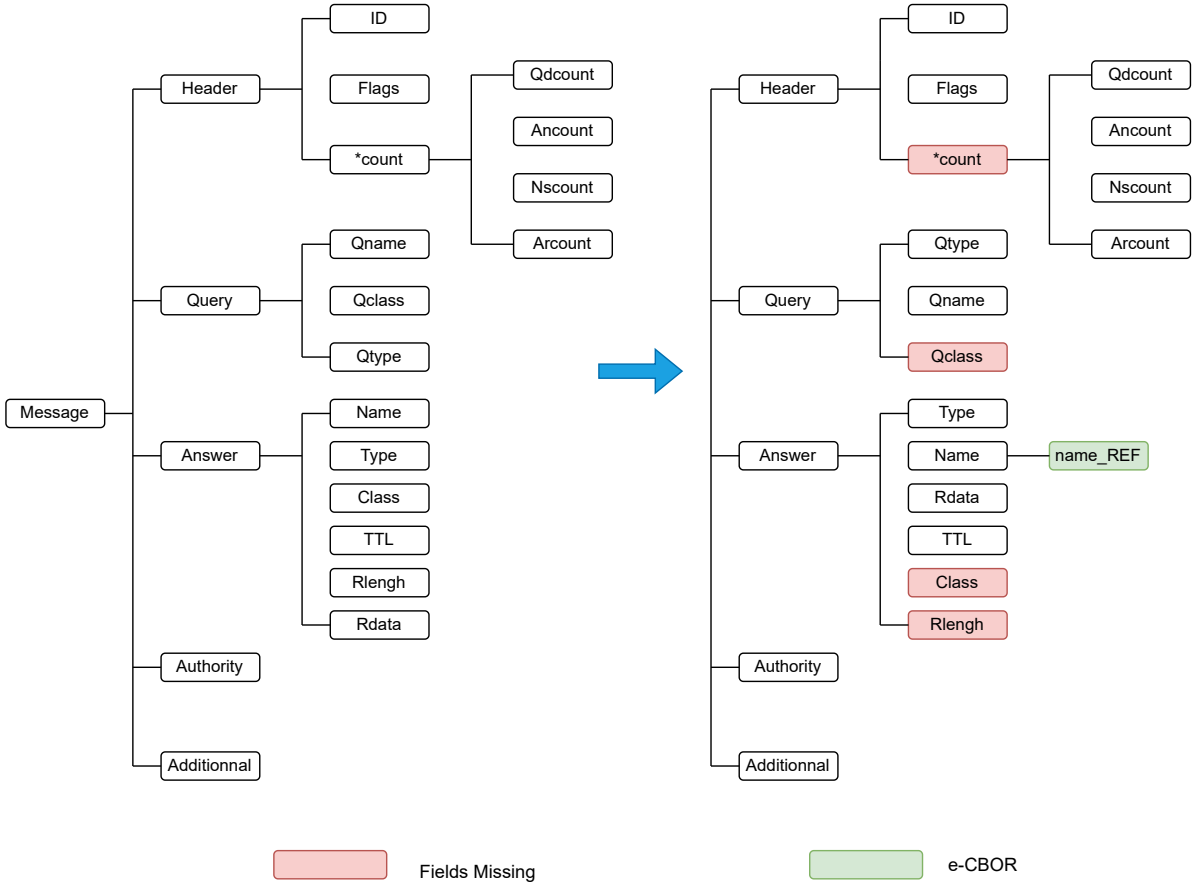


Figure 5.9: Les differents champs traités avec la méthode e-CBOR

5.5.1 Environnement de Test

Pour évaluer les performances de notre approche, un banc d'essai a été établi à l'aide de l'ensemble de données de [92] contenant 1000 noms de domaine opérationnels. Comme le montre la Fig. 5.10, un module Python a été développé pour analyser les noms de domaine dans un fichier CSV et pour effectuer des résolutions DNS de type `A`, `AAAA`, `CNAME` et `SOA`. L'outil Wireshark est utilisé pour capturer et stocker le trafic DNS pendant la résolution. Une fois que le fichier `.pcap` est généré par Wireshark, la méthode e-CBOR est appliquée à toutes les requêtes et réponses DNS.
Nous évaluons également notre approche dans un scénario où le serveur DNS met en œuvre DNSSEC, y compris les types d'enregistrements de ressources `DS`, `RRSIG`, `DNSKEY` et `NSEC`. Dans un scénario réel, ces requêtes DNS sont effectuées par les navigateurs web, et non par un module Python. Il convient également de noter que, dans tous les cas, les URL dont les résolutions aboutissent à des réponses non valides ont été éliminées de l'ensemble de données.

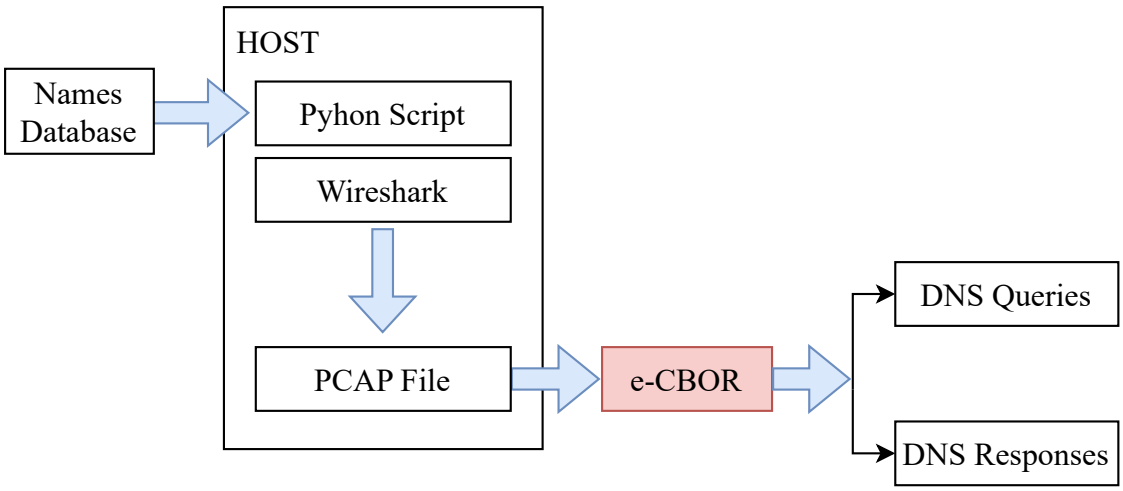


Figure 5.10: Processus de collecte des requêtes et des réponses DNS

5.5.2 Format de requête DNS

La Figure. 5.11 montre le pourcentage de réduction lors du passage du format DNS classique au format e-CBOR. Pour évaluer la taille du format proposé, nous analysons le trafic DNS généré après l'encodage de la requête à l'aide de l'approche e-CBOR. Pour ce faire, nous itérons à travers chaque réponse DNS et comparons la taille en octets de notre format proposé avec celle du format DNS classique, puis nous calculons la différence δ existant entre des tailles des deux formats.
Le pourcentage médian de réduction pour les requêtes est de 57%. En effet, pour encoder une requête DNS avec e-CBOR, nous avons besoin de : 1 octet pour spécifier un tableau CBOR avec 3 éléments, 3 octets pour encoder l'ID, 3 octets pour le `FLAGS`, 1 octet pour le sous tableau CBOR contenant les champs `QNAME` et `QTYPE`, n (en fonction de la longueur du nom) octets pour encoder `QNAME` et 1 octet pour encoder `QTYPE`.

5.5.3 Format de réponse DNS

La Fig. 5.12 montre les résultats des tests effectués sur l'ensemble de données présenté dans la section précédente. La Fig. 5.14 représente la taille résultante de `name_ref`. Nous considérons les types les plus courants : `A`, `AAAA`, `CNAME`, `SOA`.
Comme le montrent nos résultats dans les figures 5.12, 5.13 et le tableau 5.6, nous avons deux scénarios principaux en fonction du nombre d'éléments (L) présents dans le champ `name_ref` : (i) $\delta < 0$, représentant une meilleure représentation des données DNS par rapport au format DNS classique, et (ii) $\delta > 0$ indiquant une augmentation de la taille de l'e-CBOR.

$\delta < 0$ Nos observations sont les suivantes :

- Pour $L = 1$, notre approche présente un pourcentage de réduction allant jusqu'à 34% pour le type d'enregistrement de ressource `A`, 18% pour le type `AAAA`. Ici, nous

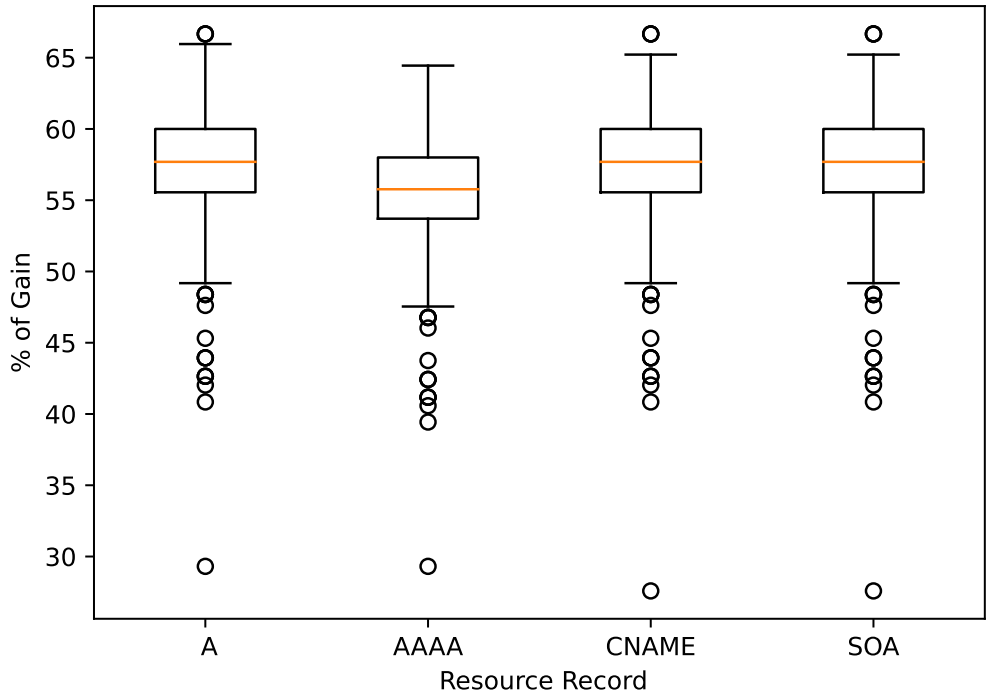


Figure 5.11: Pourcentage de réduction lors du passage des requêtes DNS classiques à e-CBOR

avons un `name_ref` avec un seul élément. Ce `name_ref` peut être écrit, par exemple, sous la forme : {0 : 'force.com'}

- Pour $L = 2$, c'est le cas lorsque le `name_ref` est composé d'au moins deux éléments, dont l'un est un tableau. Cela nous donne un taux de réduction qui peut aller jusqu'à 9% pour le type A, 5% pour le type `AAAA`, 3% pour les types `CNAME` et `SOA`. Exemple d'une référence nommée à deux éléments avec un tableau : {0 : 'tianya.cn', 1 : ['root', 0]}
- Pour $L = 3$, `nom_ref` composé d'au moins trois éléments non imbriqués. Le pourcentage de réduction peut atteindre 6% pour `A`, 4% pour `AAAA`, et 3% pour `CNAME` et `SOA` respectivement. Par exemple, `name_ref` avec trois éléments et deux tableaux a la forme {*0 : 'acs.org', 1 : ['dnsgrid', 0], 2 : ['network', 0]*}
- Pour $L = 4$, un `name_ref` de quatre éléments avec au plus trois tableaux. Contrairement au cas $L = 3$, où les éléments des tableaux sont des nombres, ici les éléments des tableaux peuvent toujours être des tableaux. On peut ainsi obtenir un taux de réduction allant jusqu'à 5% pour A, 3% pour `AAAA`, `CNAME` et `SOA`.

$\delta > 0$ Dans ce cas, la taille du format DNS proposé reste supérieure à celle du format DNS classique. Pour tout `name_ref` avec un $L >= 5$, notre approche présente une mauvaise représentation des données DNS. Nous pouvons affirmer que lorsque nous avons un `name_ref` à cinq éléments avec au moins quatre tables, notre approche n'est pas la meilleure solution.

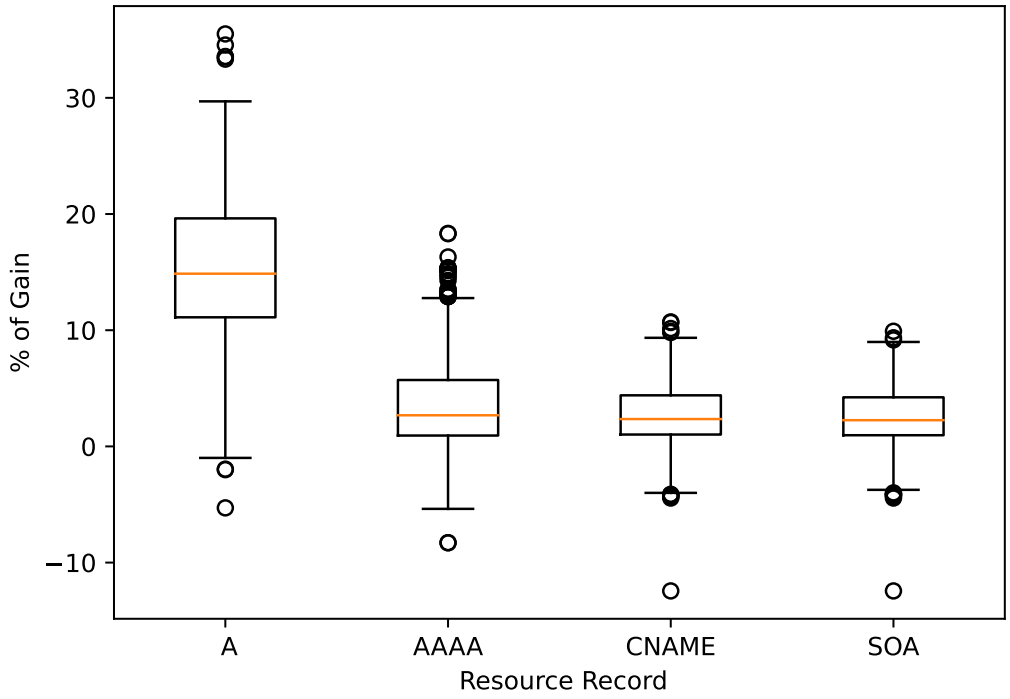


Figure 5.12: Pourcentage de réduction lors du passage des réponses DNS classiques à e-CBOR

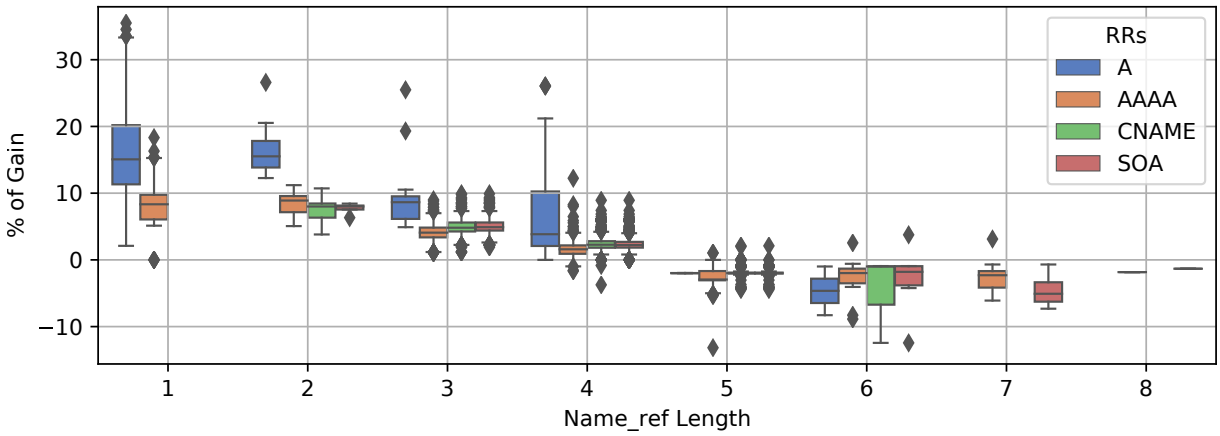
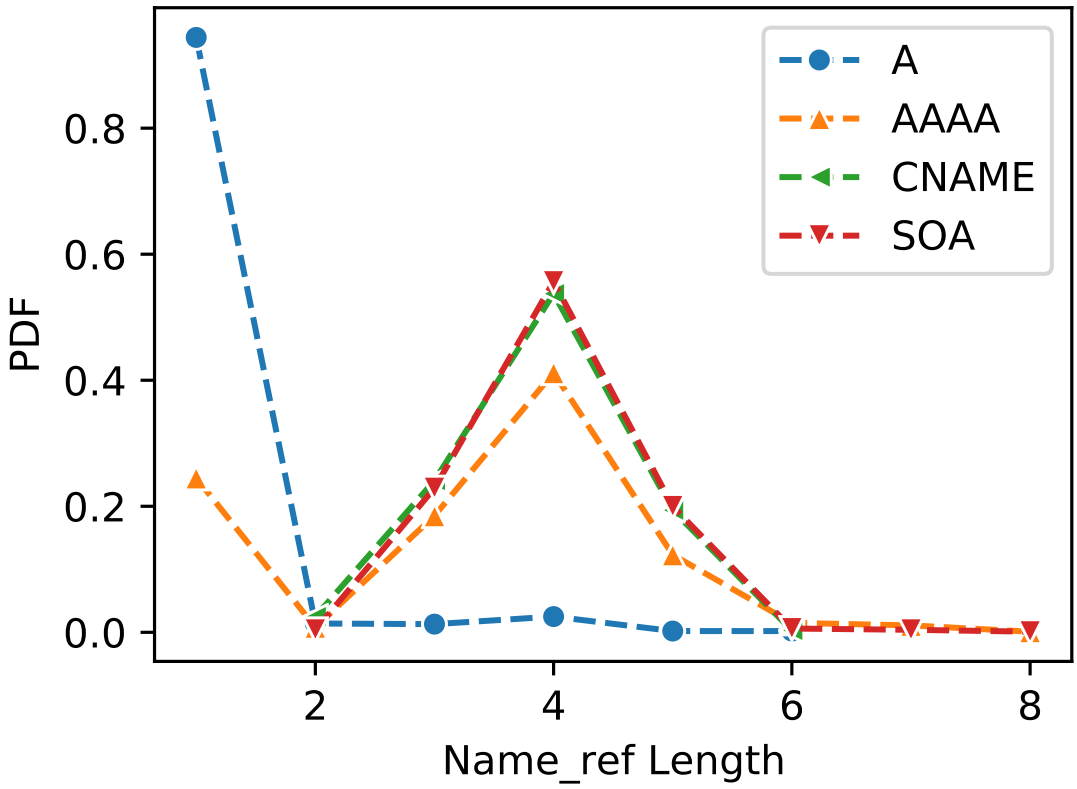


Figure 5.13: Pourcentage de réduction de eCBOR par rapport au DNS classique Réponses avec `name_ref(N)`

La Fig. 5.14 montre également que plus de 90% des réponses renvoyées par le serveur DNS à une résolution de type RR `A` ont un `name_ref` de longueur $L = 1$. La Fig. 5.13 montre que $L = 1$ est la longueur de `name_ref` qui offre le meilleur taux de réduction. L'approche présentée fonctionne donc très bien pour les résolutions de type `A`, qui sont les plus utilisées à ce jour. Pour les résolutions de type `AAAA`, nous avons environ 39% de réponses avec un `name_ref` de longueur $L = 4$. Pour les types `CNAME` et `SOA`, ce chiffre



é

Figure 5.14: Longueur du `nom _ref (L)` pour les différents RR avec e-CBOR

s'élève à environ 60%. Nous signalons également un nombre moins important de réponses de type `A` ayant un `name_ref` de longueur comprise entre $L = 2$ et $L = 6$. Cela peut conduire à une meilleure compression de la réponse DNS, car plus la longueur du `name_ref` est importante, plus la représentation e-CBOR sera grande.

5.5.4 DNSSEC utilisant e-CBOR

Dans cette section, nous montrons comment étendre notre approche à DNSSEC et présentons les résultats de notre mise en œuvre. DNSSEC[72] fournit une représentation différente des données par rapport au DNS classique. Il contient un plus grand nombre de champs et certains de ses enregistrements de ressources, tels que `DNSKEY`, `DS` et `RRSIG`, comportent entre 8 et 13 champs. Cela peut représenter un défi particulier pour e-CBOR, car la présence de nombreuses zones peut entraîner une augmentation de la taille de la représentation finale. Cependant, une grande quantité d'informations répétées peut être comprimée à l'aide de e-CBOR.
Dans la représentation des données DNSSEC avec e-CBOR, certains champs ont été ignorés comme dans le cas du DNS simple. La figure 5.15 montre un exemple de codage `RRSIG`. Le champ `class` n'est pas pris en compte dans la nouvelle représentation des données dans tous les différents RR DNSSEC.

```
[47347, 33184,     <-- Header
               [0, 28],     <-- Query
               {0: example.com}  <-- Name_ref
               [
                [28,0,IPv6 Address,TTL], <-- RR Signed
                [46,0,TTL,28,Algorithm, Labels,Orignal TTL ,
                  Signature Expiration, Signature Inception,
                  Key tag,0,Signature],        <-- DNSSEC RR
               ]
```

Figure 5.15: Exemple d'encodage de `RRSIG` avec e-CBOR

La figure 5.16 montre que la représentation des `RRs` DNSSEC dans le format proposé est plus compacte que dans le format classique : (i) Pour `DS`, nous avons une structure plus compacte avec un gain médian de 5%. ii) Pour `RRSIG`, le gain médian est de 6%. iii) `NSEC` est l'enregistrement de ressource pour lequel le gain le plus important est de 18%, ce qui est dû à la grande quantité d'informations redondantes. iv) La taille de la clé publique de `DNSKEY` a été réduite de 3% .

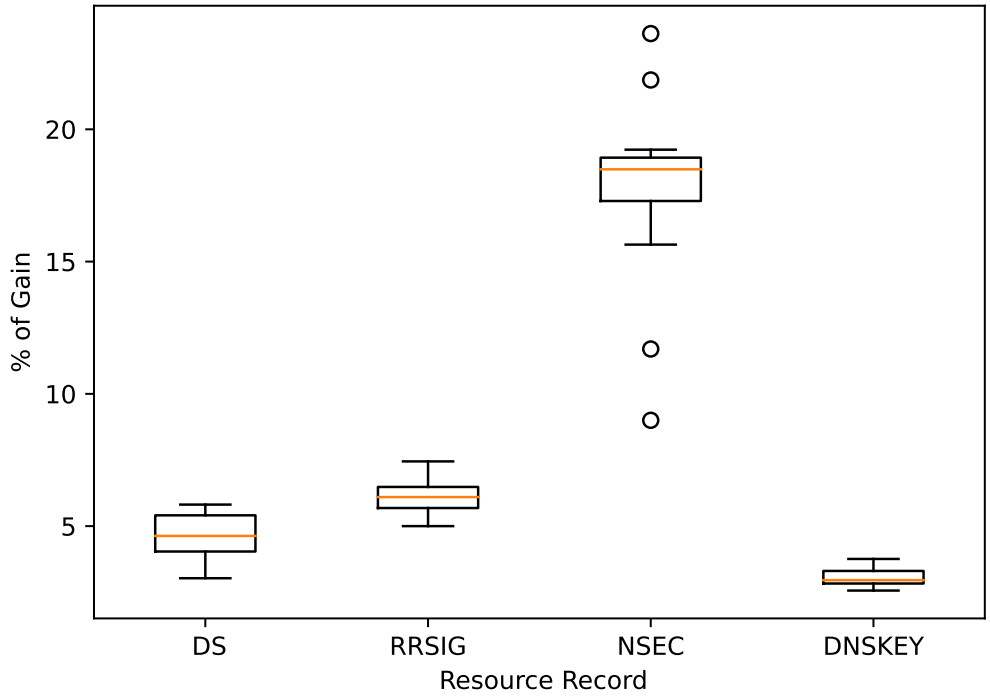


Figure 5.16: Pourcentage de réduction des requêtes DNSSEC avec e-CBOR

5.6 Conclusion

Nous avons proposé un nouveau format d'encodage pour représenter les données DNS et DNSSEC, entièrement compatible avec "wireformat" utilisé actuellement. Sa base CBOR lui confère la souplesse nécessaire pour s'adapter facilement aux évolutions futures. La mise en œuvre proposée montre une réduction significative de la taille des messages par rapport au format traditionnel.

Chapter 6

Segmentation sécurisée de la résolution de noms de domaine

It is always possible to add another level of indirection.

— **RFC 1925, Nr. 6a**
(The Twelve Networking Truths)

Sommaire

6.1 Introduction

Le DNS actuel est public. N'importe qui, à n'importe quel endroit sur l'Internet, peut demander la résolution d'un nom de domaine et obtenir un résultat. Avec l'introduction de nouveaux mécanismes tels que DoH, HTTPS encapsule les requêtes DNS dans des tunnels TLS sécurisés. L'objectif est de protéger les échanges des utilisateurs sur le web en assurant la confidentialité et l'intégrité ; le trafic DNS ne peut donc pas être spécifiquement filtré puisqu'il utilise HTTPS et que les requêtes DNS sont chiffrées.
DoH est devenu très populaire puisque tous les principaux navigateurs web incluent par défaut DoH pour la résolution de noms et que de grandes entreprises telles que Cloudfare et Google offrent des services DoH. Cependant, ce leadership peut affaiblir la neutralité du réseau puisque tout le trafic DNS est centralisé sur quelques serveurs. Cela peut être évité en décentralisant l'infrastructure DNS.
L'implémentation actuelle de DoH authentifie le résolveur DoH par le biais de son certificat public. L'identité de la requête n'est pas vérifiée car la philosophie est de maintenir la confidentialité. Si l'identité de l'utilisateur pouvait être vérifiée par le résolveur, l'accès à une partie de l'espace de nommage DNS serait limité à certains utilisateurs ou applications possédant les certificats appropriés.
Dans ce chapitre, nous proposons d'enrichir l'architecture classique du DNS par l'ajout d'une nouvelle entité appelée ***DNS-Broker***. Cette nouvelle architecture permet d'étendre les fonctionnalités de DoH par une ***segmentation*** de l'accès à l'espace de noms en s'appuyant sur une infrastructure de gestion de clés publiques/privées PKI [1]. Notre approche permet également de réduire le nombre de messages échangés entre le client et le serveur DNS lors d'une résolution qui aura été préalablement autorisée sur la foi des contrôles dérivés des échanges de certificats.

6.2 Résolution sur une architecture DNS utilisant DoH

6.2.1 Rôle du protocole HTTPS

Nous avons décrit le mode de fonctionnement de DoH à la section 3.3.1 du chapitre 3. Nous avons vu que cette architecture permet aujourd'hui de sécuriser le canal de communication entre le client et le résolveur grâce au protocole HTTPS qui permet d'envoyer toutes les données DNS sur le port TCP 443. Contrairement à sa première version HTTP, HTTPS (HTTP over TLS) assure le chiffrement des échanges par l'ajout de la couche de chiffrement TLS, rendant les données illisibles si elles sont interceptées par un attaquant qui écoute la communication. Afin de garantir un certain niveau de sécurité aux utilisateurs accèdant à un site, ce protocole repose sur trois principes fondamentaux :

1. Les utilisateurs peuvent utiliser des certificats d'authentification délivrés par des tiers pour vérifier l'identité des sites Internet qu'ils visitent.
2. Il garantit la confidentialité des données envoyées par les utilisateurs et des données reçues du serveur. C'est une garantie que les données reçues proviennent d'un serveur Web fiable.

[1] Une présentation de l'architecture PKI est proposée en annexe

3. L'intégrité des données échangées est aussi garantie : les données reçues n'ont pas été modifiées par un tiers lors de la communication.

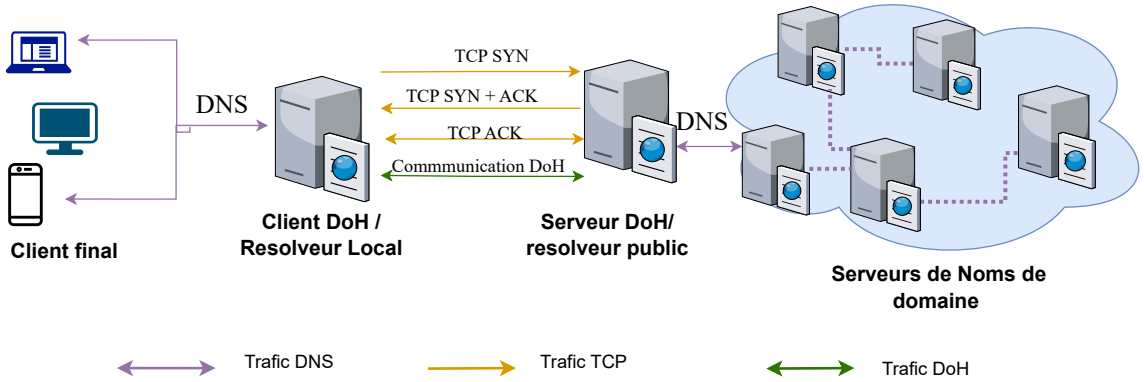


Figure 6.1: Résolution DNS utilisant DoH

La Figure 6.1 montre une résolution DNS utilisant le protocole DoH. Lorsqu'une requête DNS envoyée par le client final arrive au niveau du client DoH, celui-ci va premièrement établir une connexion TCP avec le serveur DoH (ou avec un résolveur DNS public implémentant DoH à l'instar de Google et Cloudflare) et cela se fait en trois phases : i) Le client DoH commence par ouvrir une connexion TCP en envoyant un paquet SYN (synchronisation) au serveur. Le paquet SYN contient des informations de séquence initiale. ii) ensuite, le serveur DoH reçoit le paquet SYN, crée une nouvelle session TCP, enregistre les informations de séquence, et répond avec un paquet SYN-ACK (synchronisation-accusé de réception). iii) Enfin, lorsque le client DoH reçoit le paquet SYN-ACK du serveur et il répond par un paquet ACK (accusé de réception) confirmant que la connexion est établie. La connexion TCP est maintenant active. Une fois la communication établie, le client DoH envoie un message ClientHello au serveur pour initier la négociation TLS. Le message ClientHello contient des informations sur les versions TLS prises en charge et les paramètres de chiffrement. Le serveur répond avec un message ServerHello, sélectionnant la version TLS et les paramètres de chiffrement à utiliser pour la communication sécurisée. Le client et le serveur DoH négocient par la suite les paramètres de sécurité, y compris le choix d'un algorithme de chiffrement et la génération de clés de session. Lorsque les clés pour établir la session sont échangées, les deux entités peuvent désormais échanger des données DNS en toute sécurité.

6.2.2 Modèle de gestion des certificats utilisés par DoH

Dans l'implémentation actuelle de DoH, le modèle de génération et de vérification des certificats est le **modèle hiérarchique** présenté en annexe. Toutes les grandes entreprises à l'instar de Google et Cloudfare qui fournissent les services DoH à leurs clients ont chacune leur infrastructure de gestion des certificats. Ces entreprises mettent en place une Autorité de Certification (AC) chargée de l'ensemble des filiales, appelée AC racine

(ou Root Certificate Authority (CA)). Cette AC racine autorise ensuite d'autres AC, appelées sous-AC, à signer les certificats des clients. Pour ce faire, l'AC racine génère d'abord son propre certificat auto-signé, puis signe les certificats des sous-AC. Cette étape, qui équivaut à l'établissement d'une relation de confiance entre deux AC, est appelée certification croisée. Ensuite, les sous-AC se chargent de signer les certificats des utilisateurs finaux. La schéma résultant est donc hiérarchique et ressemble à une structure en arborescence. Ainsi, la vérification des certificats d'un utilisateur passent nécessairement par la validation de la chaîne de certification. Mais actuellement la vérification des certificats sur DoH se fait à sens unique.

6.3 Architecture proposée

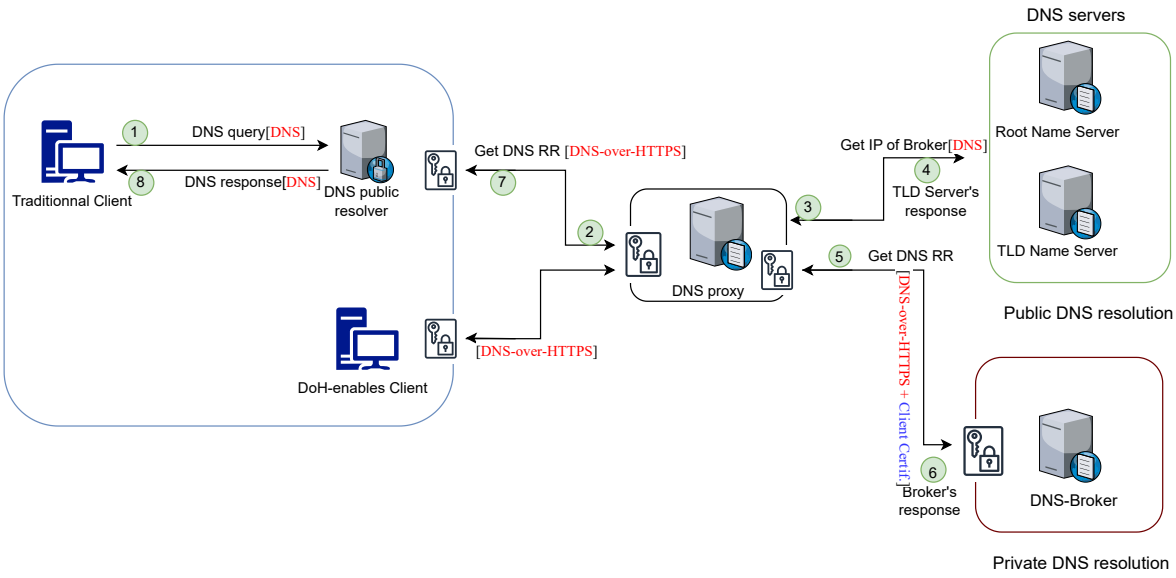


Figure 6.2: Architecture DNS-Broker

6.3.1 Description de DNS-Broker

La figure 6.2 présente l'architecture proposée. Le propriétaire des noms de domaines s'enregistre auprès du DNS-Broker et indique à l'autorité, la partie de son espace de noms dont il souhaite limiter l'accès au grand publique. Un accord est signé entre le propriétaire et l'autorité en charge du DNS-Broker. Nous utilisons ici des certificats pour matérialiser cet accord. On peut avoir plusieurs types de DNS-Broker en fonction des domaines. Par exemple, un pour gérer les identifications et l'accès aux ressources des laboratoires de recherche universitaires et un autre pour les industriels.
Lors de la résolution DNS, le nom de domaine utilisé par le client doit être divisé en deux parties différentes. Par exemple, pour contacter le serveur faisant autorité, le nom de

domaine est composé d'une partie publique et d'une partie privée (soulignée). Ainsi, nous avons : `DevEUI{EUI.64}.deveui.iot-roam.net`, et la résolution se fait comme suit :

1. Partie publique : `deveui.iot-roam.net` est résolue à l'aide du serveur DNS local en amont, la réponse DNS correspond à l'IP du Broker.
2. Partie privée: `DevEUI{EUI.64}` est résolue en utilisant le DNS-broker comme serveur en amont. Un certificat côté client est nécessaire pour obtenir une réponse DNS satisfaisante.

Les certificats des clients sont délivrés par le DNS-Broker, qui fait également office d'autorité de certification. Lorsqu'un client cherche ainsi à accéder à une ressource privée telle qu'une adresse IP identifiant d'un objet connecté associé à un nom de domaine, il passe par les étapes suivantes :

1. Le client final envoie une requête de résolution avec le nom de domaine par exemple `DevEUI{EUI.64}.deveui.iot-roam.net` au résolveur DNS public [étape 1].
2. Le résolveur ne possédant pas la réponse du client, va envoyer sa requête à un proxy DNS qui est chargé de vérifier la validité des certificats utilisés dans la requête du client et d'établir la communication entre le client et le DNS-Broker. Il vérifie aussi la partie publique et privée du nom de domaine [étape 2].
3. Le proxy DNS contacte les serveurs racines, ensuite les serveurs TLD avec la partie publique du nom de domaine afin d'avoir l'adresse IP du DNS-Boker qui detient les informations privées [étapes 3-4].
4. Le DNS proxy va vérifier si le client a souscrit un abonnement auprès du DNS-Broker qui l'autorise à faire une telle demande. Cette vérification se fait via les certificats. Si les certificats sont valides, le DNS-Broker répond au DNS proxy par un message contenant la ressource demandée par le client si elle existe [étapes 5-6].
5. le DNS proxy va ainsi envoyer le message au client contenant une réponse du DNS-Broker [étapes 7-8]

6.3.2 Gestion des certificats par le DNS-Broker

Pour cette nouvelle architecture, nous proposons une procédure décentralisée ou maillée d'approvisionnement des certificats. Notre méthode implique d'accorder une autonomie totale à chacune des sous-AC, éliminant ainsi la nécessité d'une AC racine, et transformant ainsi toutes les sous-AC en AC racines indépendantes. En d'autres termes, tous les utilisateurs finaux font confiance à l'ensemble des ACs précédemment définies. Dans ce scénario, les AC établissent des relations de certification croisée en pair-à-pair entre elles. La vérification du chemin de confiance se limite donc à la validation de la signature par l'une des AC en laquelle l'utilisateur a confiance (cf. figure A.5).
Comme le montre la figure 6.4, dans notre scénario, le responsable d'un nom de domaine qui souhaite rendre une partie de sa zone DNS privée et limiter ainsi tout accès non autorisé, contacte le gestionnaire du DNS-Broker qui fait office de CA. Celui-ci lui fournira les certificats dont il a besoin. Comme le montre la figure 6.3, le modèle d'approvisionnement des certificats étant complètement décentralisée, donne la possibilité à chaque type de DNS-Broker de générer et signer lui-même les certificats qui seront utilisés par les noms de domaines qui sont dans sa base de données.

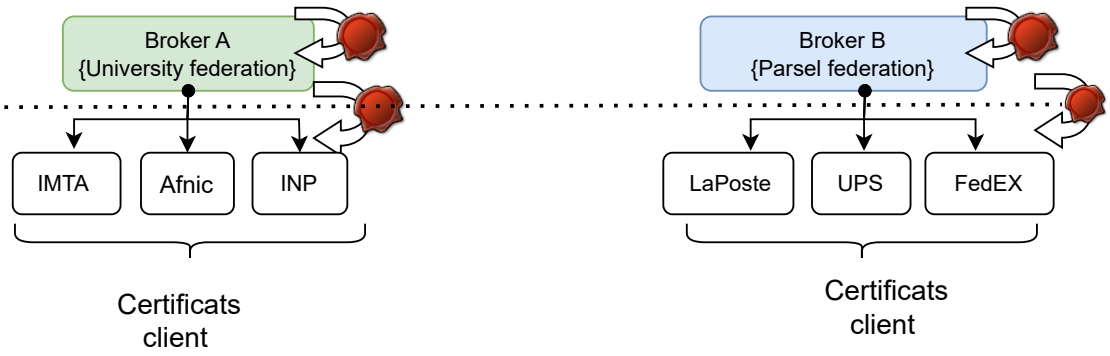


Figure 6.3: Approvisionnement en certificats pour les connexions HTTPS entre le client et le DNS-Broker

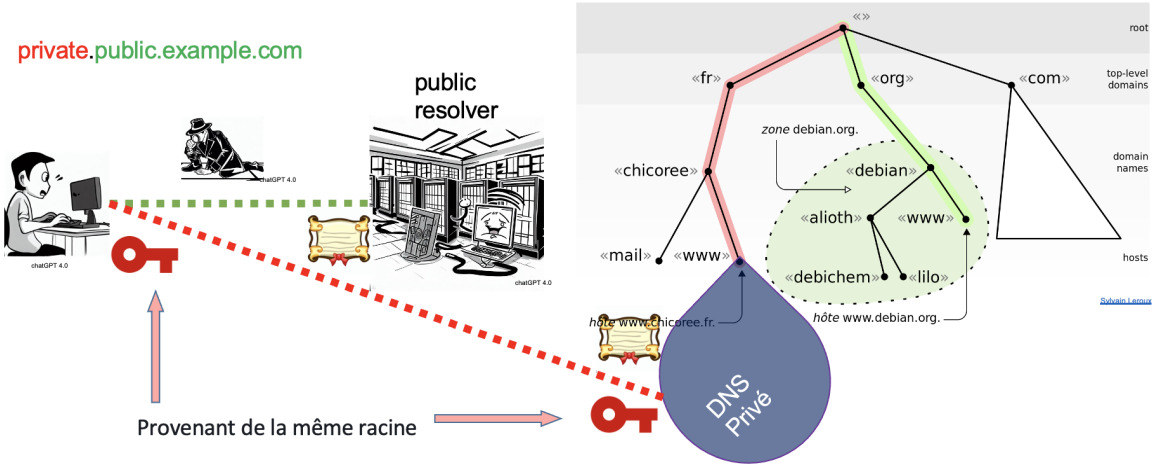


Figure 6.4: Cas d'usage du DNS privé

6.4 Mise en oeuvre de l'architecture

Nous décrivons dans cette partie les outils utilisés pour implémenter et tester de notre architecture.

6.4.1 Description de l'environnement expérimental

6.4.1.1 Raspberry PI

Au cours de ces dernières décennies, nous avons assisté à une évolution technologique significative, notamment avec l'apparition d'appareils compacts, performants et abordables, tels que le Raspberry Pi [89]. Il s'agit d'un ordinateur monocarte largement reconnu dans le domaine de l'informatique, permettant le développement d'ordinateurs polyvalents adaptés aux applications de l'Internet des objets. Le modèle Raspberry Pi 3 Modèle B que nous utilisons pour implémenter notre architecture représente la dernière avancée de cette série.

Nous avons constaté aussi que la sécurisation des systèmes distribués et des réseaux en utilisant la Raspberry Pi suscite un vif intérêt parmi de nombreux chercheurs. L'un de ces projets vise à exploiter la Raspberry Pi en tant que serveur VPN pour un réseau domestique, établissant ainsi une connexion VPN entre le réseau domestique et le réseau public.

Un autre projet exploite la Raspberry Pi comme un filtre, bloquant efficacement les noms de domaines malveillants, offrant ainsi une protection aux utilisateurs contre d'éventuelles attaques de logiciels malveillants en ligne [119]. Ce projet met en œuvre un DNS qui rejette toute requête venant des noms de domaines blacklistés par l'utilisateur et aussi aux messages non pertinents tentant de pénétrer le réseau.
Pour des raisons évoquées précédemment, nous avons choisi de déployer notre solution DNS-Broker sur la Raspberry Pi presenté à la figure 6.5.

Figure 6.5: Modèle de Raspberry utilisé pour notre architecture

6.4.1.2 Integration de Pi-hole dans la Rasperry PI

Pi-Hole est un projet open source qui a été créé en 2014 par Jacob Salmela, un administrateur système et développeur basé en Alaska, aux États-Unis. Le but initial de Pi-Hole était

de proposer une solution simple et efficace pour bloquer les noms de domaines indésirables sur un réseau local en utilisant un Raspberry Pi ou un autre dispositif similaire.

Il tire parti de la capacité de Raspberry Pi à servir de serveur DNS pour créer un système de filtrage de contenu. Ainsi, le serveur DNS Pi-Hole agit comme un intermédiaire entre les appareils de votre réseau et Internet. Lorsqu'un appareil effectue une requête DNS pour accéder à un site Web, Pi-Hole vérifie si le domaine demandé est répertorié dans sa liste de blocage. Si c'est le cas, Pi-Hole renvoie une adresse IP locale ou nulle, empêchant ainsi l'accès au contenu indésirable.

Pi-Hole est aujourd'hui très utilisé par la communauté scientifique, administrateurs systèmes et entreprises en raison de sa simplicité d'installation et d'utilisation, et surtout de sa capacité à économiser de la bande passante. De nouvelles fonctionnalités ont été ajoutées, et des améliorations ont été apportées pour optimiser son efficacité. Le projet est constamment mis à jour pour maintenir les listes de blocage des noms de domaines à jour et offrir une expérience de navigation plus propre et plus sécurisée.

Nous l'avons personnalisé et déployé sur notre Raspberry afin qu'il puisse répondre à nos besoins, i.e rendre une partie du nom de domaine privé et donc l'accès nécessite l'utilisation des certificats valides. Cette implémentation est décrite à la section suivante.

6.4.2 Implémentation du DNS-Broker

Pour l'implémentation du DNS-Broker, nous avons mis en place un serveur physique avec une IP publique et un domaine enregistré : `broker.iot-roam.net`. Trois services sont exécutés sur ce serveur :(i) un proxy inverse nginx, chargé de l'authentification du client et de la validation du certificat, (ii) une version modifiée de dnsproxy agissant en tant que serveur DoH et (iii) pihole chargé de la gestion de la zone DNS (cf. Figure 6.6).
Comme le montre la figure 6.6, la requête DoH arrive à `https://broker.iot-roam.net/dns-query`. La porte d'entrée du serveur est nginx, qui agit en tant que proxy inverse. Sa fonction est de diriger les requêtes des clients vers le serveur backend approprié. Il est également responsable de l'établissement de la poignée de main HTTPS, y compris la vérification des certificats des clients. Une fois cela fait, il décompresse le HTTPS et dirige le message DNS HTTP simple vers le serveur DoH (dnsproxy).
Ensuite, la requête arrive à une version modifiée de dnsproxy, qui est un résolveur de proxy DNS open-source. Ce logiciel permet de décapsuler la requête DNS et de la diriger vers pihole sur le port 53. Pihole est alors chargé de gérer la zone DNS. La zone DNS est simplement une entrée CNAME pour chaque `DevEUI` comme indiqué ci-dessous :

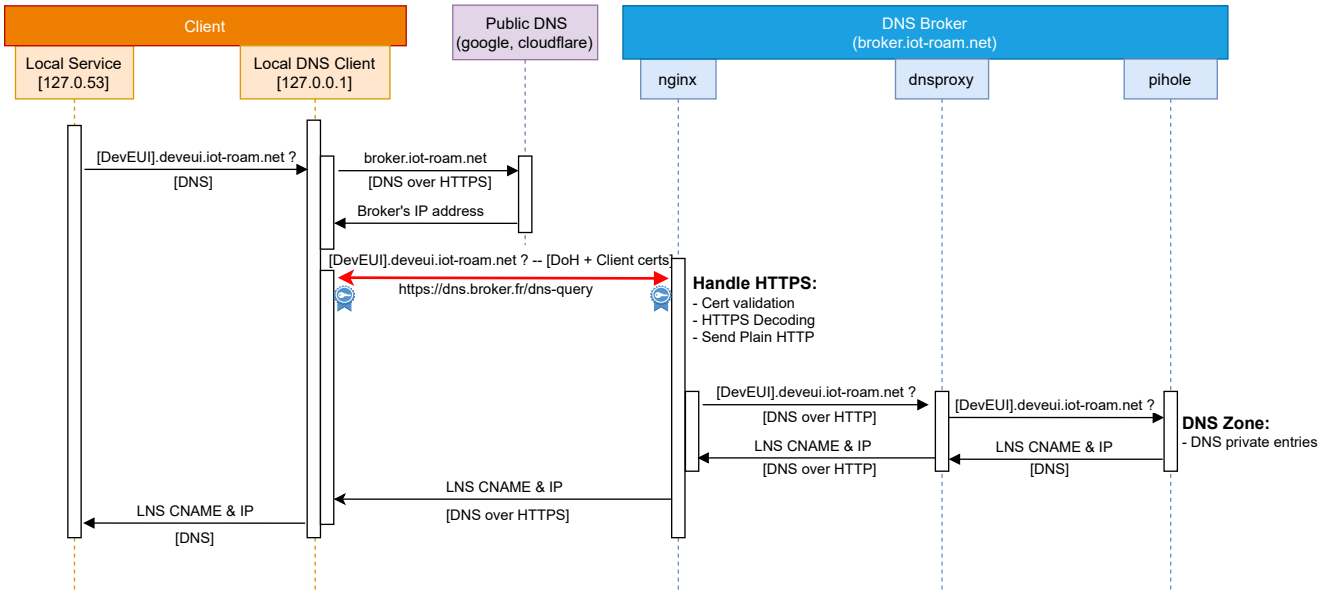


Figure 6.6: DNS Broker implementation

```
; <<>> DiG 9.16.1-Ubuntu <<>> @127.0.0.1 ae5823c4f5432d32.deveui.iot-roam.net
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 39335
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;ae5823c4f5432d32.deveui.iot-roam.net. IN A

;; ANSWER SECTION:
ae5823c4f5432d32.deveui.iot-roam.net. 0 IN CNAME 000016.netid.iot-roam.net.
000016.netid.iot-roam.net. 0    IN  A   51.68.127.56

;; Query time: 163 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: mar. janv. 18 11:24:15 CET 2022
;; MSG SIZE  rcvd: 108
```

6.4.3 Résultats

Les figures 6.7 et 6.8 présentent les résultats de nos implémentations décrites dans la section précédente.

6.4.4 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle architecture baptisée DNS-Broker permettant de segmenter la résolution DNS. Cette architecture contribue à limiter l'accès à un espace de noms. Un propriétaire d'une zone a désormais la possibilité de déterminer qui peut accéder à une ressource dans sa zone en fonction de ses besoins de sécurité.

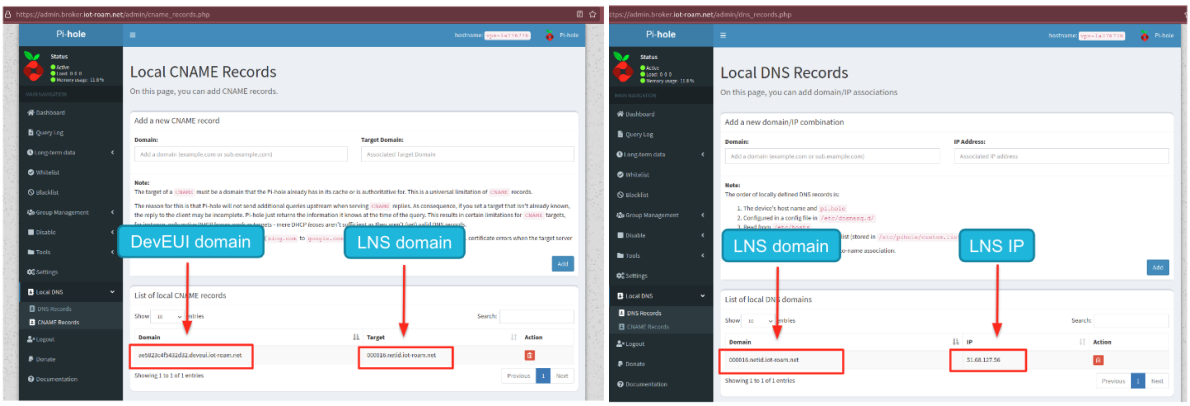


Figure 6.7: Implementation du DNS-Broker avec Pi-hole deployé sur une Raspberry P Modèle B+

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	54.37.155.27	8.8.8.8	DNS	79	Standard query 0x132c AAAA broker.iot-roam.net
2	0.000075315	54.37.155.27	8.8.8.8	DNS	79	Standard query 0xbf9c A broker.iot-roam.net
3	0.018529869	8.8.8.8	54.37.155.27	DNS	95	Standard query response 0xbf9c A broker.iot-roam.net A 51.38.184.112
4	0.019006029	8.8.8.8	54.37.155.27	DNS	136	Standard query response 0x132c AAAA broker.iot-roam.net SOA ns1.gandi.net
5	0.019458898	54.37.155.27	51.38.184.112	TCP	74	57664 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2483300776 TSecr=0 WS=128
6	0.029850408	51.38.184.112	54.37.155.27	TCP	74	443 → 57664 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=237982797 TSecr=2483300
7	0.029880703	54.37.155.27	51.38.184.112	TCP	66	57664 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2483300786 TSecr=237982797
8	0.030364205	54.37.155.27	51.38.184.112	TLSv1…	382	Client Hello
9	0.040658481	51.38.184.112	54.37.155.27	TCP	66	443 → 57664 [ACK] Seq=1 Ack=317 Win=64896 Len=0 TSval=237982808 TSecr=2483300787
10	0.041371874	51.38.184.112	54.37.155.27	TLSv1…	2962	Server Hello, Change Cipher Spec, Application Data, Application Data
11	0.041393512	54.37.155.27	51.38.184.112	TCP	66	57664 → 443 [ACK] Seq=317 Ack=2897 Win=63232 Len=0 TSval=2483300798 TSecr=237982808
12	0.041406758	51.38.184.112	54.37.155.27	TCP	1266	443 → 57664 [PSH, ACK] Seq=2897 Ack=317 Win=64896 Len=1200 TSval=237982808 TSecr=2483300787 [TCP se
13	0.041411634	54.37.155.27	51.38.184.112	TCP	66	57664 → 443 [ACK] Seq=317 Ack=4097 Win=62080 Len=0 TSval=2483300798 TSecr=237982808
14	0.043211514	51.38.184.112	54.37.155.27	TLSv1…	1463	Application Data, Application Data, Application Data
15	0.043220225	54.37.155.27	51.38.184.112	TCP	66	57664 → 443 [ACK] Seq=317 Ack=5494 Win=64128 Len=0 TSval=2483300800 TSecr=237982810
16	0.077456485	54.37.155.27	51.38.184.112	TLSv1…	1766	Change Cipher Spec, Application Data, Application Data, Application Data
17	0.077528338	54.37.155.27	51.38.184.112	TLSv1…	152	Application Data
18	0.077600035	54.37.155.27	51.38.184.112	TLSv1…	235	Application Data

Figure 6.8: Échanges entre un client et le DNS-Broker capturés sur wireshark

Les domaines d'applications de notre approche restent nombreux. Ainsi nous faisons face aujourd'hui au problème d'interopérabilité dans le domaine de l'Internet des Objets. Chaque propriétaire d'objets fabrique et met en service ses objets sans vraiment se préoccuper de leur interopérabilité avec des objets en provenance d'autres fabricants. Les études ont montré que si on souhaite fédérer ces objets venant de fabricants ou d'opérateurs différents, le DNS reste le meilleur candidat. Notre approche permet d'offrir une première solution à cette problématique.

Part III

Vers de nouveaux usages du DNS

Chapter 7

Itinérance d'objets connectés

Some things in life can never be fully appreciated nor understood unless experienced firsthand.

— **RFC 1925, Nr. 4**
(The Twelve Networking Truths)

Sommaire

7.1 Introduction

En connectant les objets du quotidien aux réseaux, l'Internet des Objets (IdO) est une évolution importante du monde numérique. Son développement exceptionnel se reflète dans le nombre d'objets connectés estimé à environ 75 milliards d'ici 2025 [22]. L'IdO a un impact significatif sur les services B2B et B2C dans tous les secteurs d'activité liés aux TIC (environnement, agriculture, santé, etc.).Certains usages impliquent des dispositifs en mouvement.

Les réseaux étendus à faible consommation d'énergie (LPWAN) sont un élément clé pour améliorer la mobilité des objets connectés [22]. En effet, la mobilité augmente considérablement le nombre d'utilisations et donc le nombre d'objets connectés. Cependant, la couverture d'un Opérateur Réseau ou Network Operator (NO) limite la mobilité de ces appareils. Ainsi, pour exploiter pleinement le potentiel de LPWAN, il faut envisager des scénarios d'itinérance à minima entre des opérateurs de réseaux déployant la même technologie.

L'itinérance (*roaming en anglais*) désigne le service fourni par un NO permettant à un appareil d'utiliser en toute sécurité un Réseau visité ou Visited Network (VN), qui peut se trouver dans un autre pays (itinérance internationale) ou dans le même pays (itinérance nationale) lorsqu'il se trouve en dehors de sa zone habituelle. Si l'itinérance est un pilier essentiel des réseaux cellulaires et Wi-Fi, elle n'en est encore qu'à ses débuts pour les réseaux LPWAN utilisant des bandes de fréquences sans licence.

De toutes les technologies LPWAN, les plus populaires sont les suivantes : (i) SigFox, un réseau unique où les appareils sont identifiés globalement, ce qui permet aux objets de se déplacer d'un pays à l'autre sans autre limitation que la bande régionale utilisée. La bande passante peut être utilisée en fonction de la région visitée. (ii) Long Term Evolution Cat M1 (LTE-M) et Message Queuing Telemetry Transport (NB-IoT) basés sur les technologies 3GPP (4G et 5G) qui intègrent certaines capacités d'itinérance permettant aux fournisseurs d'accepter des appareils d'autres pays dans leur réseau. iii) Enfin, la technologie Long Range Wide Aera Network (LoRaWAN) qui souffre d'une forte "balkanisation" du réseau. Les passerelles bon marché et l'ouverture du protocole facilitent le déploiement d'un réseau dans une zone spécifique. Les réseaux plus importants sont souvent limités à un pays, et les initiatives mondiales telles que The Things Network (TTN) ou Helium n'offrent pas une bonne couverture pour de nombreuses applications.

7.2 Architectures de Roaming IoT

Les premiers services de communication mobile, qui exigent de l'utilisateur qu'il déclare explicitement son inscription au réseau via n'importe quel accès physique, n'ont pas été couronnés de succès. Les opérateurs ont alors procédé à une évolution majeure du réseau d'accès pour permettre au réseau de gérer automatiquement la mobilité des utilisateurs. Le fait de décharger l'utilisateur grâce à cette nouvelle capacité du réseau a été la clé du succès sans précédent des réseaux mobiles. Afin d'exploiter toutes les possibilités, notamment en

permettant à un utilisateur d'avoir accès à tous ses services quelle que soit sa localisation, la mobilité intra-domaine a été rapidement étendue par l'itinérance inter-domaine. Cette dernière a nécessité la définition d'accords d'itinérance composés essentiellement d'un volet réglementaire, d'un volet économique et d'un volet technique. Dans ce qui suit, nous rappelons ce dernier point dans le contexte des réseaux cellulaires, 802.11 et LPWAN.

Dans les réseaux cellulaires, la gestion de la mobilité d'un terminal repose fondamentalement sur la mise à jour permanente de sa localisation par rapport à l'ensemble des cellules radio constituant le réseau d'accès de l'opérateur [138, 117, 6]. La version la plus récente de ces informations est conservée dans une base de données centrale du réseau et dans une autre base de données située à la périphérie du réseau, adjacente à la zone cellulaire où le terminal est actuellement connecté. Ces bases de données facilitent un accès fiable et en temps réel aux informations de localisation d'un terminal, qui sont essentielles pour acheminer avec précision un service de communication entrant. Outre la localisation du terminal, elles contiennent le profil de service de l'utilisateur associé. Ce profil est utilisé pour contrôler l'autorisation de l'utilisateur à initier un service de communication sortant à partir du terminal. La procédure de mise à jour de la localisation est l'outil essentiel pour soutenir la mobilité d'un utilisateur qui peut ainsi se déplacer sans problème d'une cellule à l'autre et accessoirement d'une zone d'accès à l'autre, éventuellement au cours de l'exécution d'un service de communication généralement défini comme un transfert (hand-over). Ce principe permettant une itinérance intra-opérateur s'étend sans difficulté à l'itinérance inter-opérateurs en fonction des accords d'itinérance comprenant les trois parties mentionnées ci-dessus, établis au préalable entre les opérateurs concernés.

Le Wi-Fi protège l'accès au réseau par le biais du WPA2, qui est un protocole de sécurité basé sur l'architecture 802.1X [28]. Une fois authentifié dans le WLAN, un terminal peut, au cours de ses déplacements à l'intérieur de son domaine, passer d'un point d'accès à un autre sans procédure supplémentaire. Eduroam représente une solution intéressante pour faire face à l'itinérance Wi-Fi interdomaines [155][14]. Il a été déployé avec succès entre des établissements d'enseignement européens organisés en fédération et couvrant plus de vingt mille sites [39]. Il repose sur un arbre hiérarchique de serveurs Radius dont les feuilles sont définies par les serveurs Radius locaux de chaque domaine impliqué dans la fédération. Lorsqu'un terminal souhaite accéder à un domaine visité, il envoie une requête à son point d'accès local qui la transmettra à travers l'arbre Radius jusqu'à son domaine d'origine qui authentifiera l'utilisateur itinérant et autorisera l'accès du terminal itinérant au réseau visité.

Dans [19], les auteurs proposent une nouvelle méthode de gestion de la mobilité pour l'IoT. Elle vise à garantir et à maintenir la continuité des échanges après un changement de technologie de couche de liaison entre les appareils et l'Application Server (AS). En utilisant cette méthode, les appareils peuvent passer de LoRaWAN à NB-IoT. Malgré son originalité, l'approche semble offrir une évolutivité limitée car elle repose sur une architecture centralisée.

Dans les travaux de [146], les auteurs ont proposé un mécanisme d'itinérance dans lequel

la 5G est utilisée pour l'authentification et la gestion des clés des appareils. Les capacités de mobilité et d'itinérance de LoRaWAN sont ainsi étendues à l'échelle mondiale. Les résultats des expériences réalisées sur un banc d'essai ont prouvé la faisabilité de ce mécanisme. Néanmoins, l'appareil doit mettre en œuvre deux technologies radio, ce qui augmente ses coûts de production et sa consommation d'énergie.

Pour gérer les communications dans les environnements maritimes tout en traitant les problèmes de congestion, les auteurs de [44] proposent une technique d'itinérance utilisant LoRaWAN. Un serveur de distribution (DS) est introduit pour gérer l'itinérance des véhicules de transport et le suivi des expéditions entre deux entreprises. Un module d'intelligence artificielle est alimenté par les informations collectées afin de prendre les bonnes décisions concernant le trafic. La solution d'itinérance utilisée n'est pas décrite, Les questions de sécurité ne sont pas abordées et les résultats de l'expérimentation ne sont pas fournis.

Dans [54], les auteurs ont étendu l'architecture de l'Alliance LoRa [8] en introduisant deux composants : (i) un agent maître (MA) qui gère la communication entre deux NO. Il fournit le `NetID`, qui est utilisé pour identifier un NO pendant l'itinérance, et (ii) un agent local (LA) par NO (vLA et hNA respectivement). Le MA utilise le DNS pour identifier les différents NS à partir de leur `NetID`. Ensuite, pour qu'un dispositif puisse utiliser le VN, celui-ci doit d'abord être doté d'un `JoinEUI`. Le MA utilise `JoinEUI` pour trouver le hNS. Bien que la solution soit évolutive grâce au protocole DNS, elle repose sur le `JoinEUI` qui, comme nous le verrons plus loin, induit plusieurs problèmes. Les auteurs se sont concentrés uniquement sur le handover roaming.

L'Alliance LoRa a défini l'approche la plus populaire et la plus déployée pour l'itinérance [8], qui est valable pour les appareils des versions 1.04 et 1.1. La section suivante la présente en détail.

7.3 Réseaux LoRaWAN

Nous allons maintenant approfondir notre compréhension du fonctionnement de LoRaWAN, ce qui sera essentiel pour la suite de cette partie applicative de notre travail, en particulier dans la section 7.5 où nous explorerons l'itinérance LoRa.

Definition 7.3.1. La LoRa Alliance, créée en 2015, est une organisation à but non lucratif qui a pour mission de promouvoir la technologie et l'écosystème LoRaWAN. Elle compte parmi ses membres des entreprises et des universités fortement engagées qui investissent dans sa croissance. Toute entité a la possibilité de solliciter son adhésion à la LoRa Alliance, ce qui lui permet de contribuer au développement continu de LoRaWAN. En France, on compte IMT Atlantique, Afnic et Acklio parmi ces membres aujourd'hui.

LoRaWAN [88] est une technologie centralisée où les messages sont diffusés par voie hertzienne par les appareils qui sont captés par des passerelles radio (RG) qui les relaient vers un point central, le serveur de réseau LoRa (LNS). Celui-ci gère les appareils, les

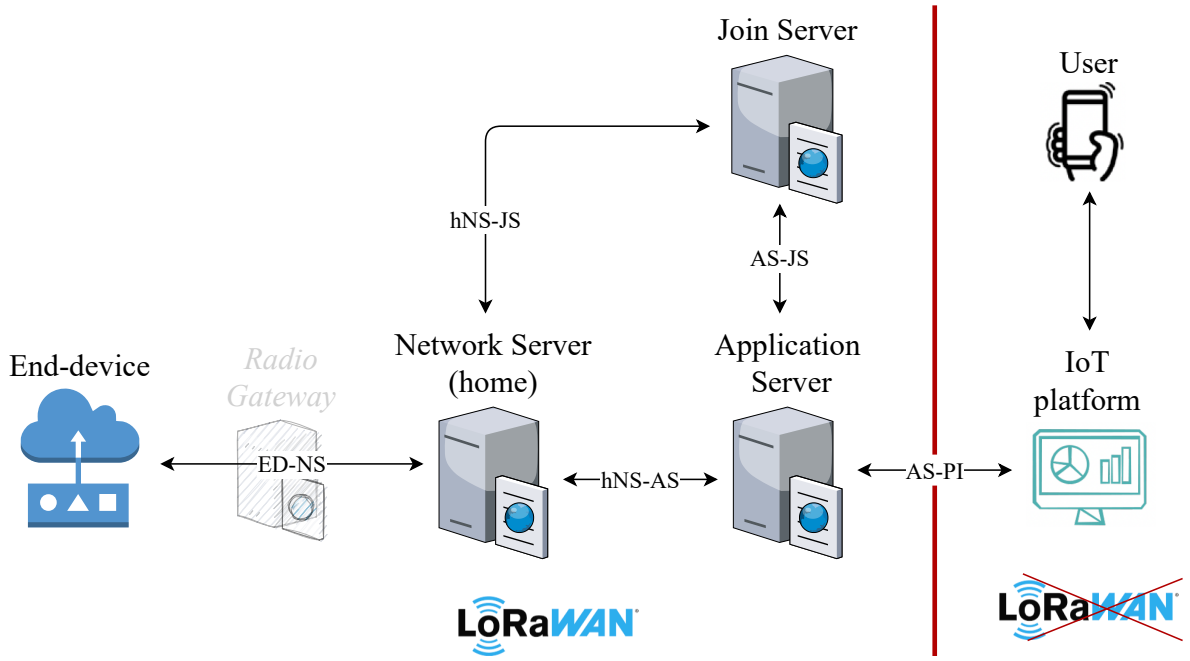


Figure 7.1: Architecture du réseau LoRaWAN[88]

authentifie et transmet leurs données à un AS où elles sont traitées.
Comme le montre la figure 7.1, l'architecture d'un réseau LoRaWAN adopte une configuration de type "étoile", dans laquelle les serveurs de réseau (*Network Servers ou NS*) occupent une position centrale, recevant l'ensemble des paquets émis par une multitude de passerelles (*Gateways*), qui, à leur tour, se trouvent au cœur d'un grand nombre de dispositifs finaux (End-*Devices*) capables d'envoyer des messages. Les passerelles sont connectées au NS via une connexion IP standard. Le rôle principal d'une passerelle consiste à agir comme un intermédiaire entre les End-Devices et le NS. Sa principale fonction est de traduire les trames Long Range (LoRa) en paquets IP et de les acheminer vers le NS, puis du NS à la passerelle pour les convertir en paquets UDP destinés aux End-Devices. Il est important de noter que les passerelles ne possèdent pas d'intelligence propre, elles n'effectuent aucun filtrage ni prise de décision. Elles n'ont même pas conscience si le paquet qu'elles transmettent est destiné au réseau ou s'il doit être acheminé vers un autre NS. Le NS peut jouer plusieurs rôles en fonction des différentes situations où il se trouve, à savoir :

- Le *Home Network Server (hNS)*, qui conserve le profil du End-Device et gère les communications avec le JS.
- Le *Serving Network Server (sNS)*, chargé de la gestion de la couche MAC, y compris les paramètres RF. Nous explorerons davantage ce rôle dans la section sur le Handover Roaming.
- Le *Forwarding Network Server (fNS)*, dont la principale fonction est de transmettre la trame LoRa PHY complète, avec très peu d'interprétations, au sNS (qui peut également être hNS). Ce rôle sera détaillé dans la partie consacrée au Roaming passif.

Ces différentes fonctions seront explorées dans la section dédiée au Roaming LoRa. Lorsque le roaming n'est pas en cours, un seul et unique serveur de réseau gère ces trois rôles.

En ce qui concerne le *Join Server (JS)*, il est responsable du processus d'activation "Over-the-Air" (OTAA) avec la génération des clés de session de communication entre le ED et NS, ensuite le NS et AS , et il peut y avoir plusieurs JS connectés à différents NS.

Quant à l'*Application Server (AS)*, il prend en charge l'ensemble de la couche applicative et fournit tous les services de cette couche à l'utilisateur final. Il est possible d'avoir plusieurs AS connectés à un seul NS, et un AS peut être connecté à plusieurs NS.

7.4 Différents types d'itinérance LoRaWAN

A notre connaissance, il existe deux types d'itinérance : (i) *passive roaming* et (ii) *handover roaming*. Nous les détaillons dans cette section.

7.4.1 Itinérance passive

Elle permet à un End-Device de bénéficier des services d'un NS tout en se trouvant dans une zone couverte exclusivement par les Gateways d'un autre opérateur. Dans ce cas, la session LoRa ainsi que la couche MAC (pour Media Access Control) sont toujours gérées par le premier serveur réseau, qui est alors désigné sous le nom de Serving NS (sNS) comme indiqué dans la section précédente. D'autre part, la gestion des trames LoRa et leur transmission par ondes radio est prise en charge par le deuxième serveur de réseau, celui qui contrôle les passerelles en communication avec le End-Device. Ce serveur de réseau est appelé le Forwarding NS (fNS). Il est important de noter qu'il ne peut y avoir qu'un seul sNS par session LoRa, tandis qu'il peut y avoir aucun (en l'absence de Roaming) ou plusieurs fNS pour une même session LoRa.

Il existe deux types de fNS : le *Stateful* et le *Stateless*. Un fNS Stateful commence par se comporter comme un fNS Stateless, mais il crée un contexte pour le End-Device concerné. Ce contexte sera ensuite utilisé pour le traitement des prochains paquets *uplink* ou *downlink* que le fNS devra gérer. En revanche, un fNS Stateless ne crée pas de tel contexte et doit traiter chaque paquet (uplink ou downlink) de manière indépendante. Il est important de noter que les spécifications officielles ne précisent pas la manière dont un fNS (qu'il soit Stateful ou Stateless) identifie ses partenaires de Roaming : cette gestion doit se faire hors bande, c'est-à-dire en dehors de l'utilisation de messages faisant partie du protocole LoRa.

La procédure d'activation du End-device lorsqu'il est en roaming passif sera détaillée dans le chapitre 8.

7.4.2 Roaming Handover

Le Roaming *Handover* est le type de roaming que nous pouvons expérimenter lorsque nous voyagons à l'étranger avec notre téléphone. Dans ce cas, notre carte SIM bascule totalement vers un opérateur différent et qui a établi un partenariat avec notre opérateur d'origine. Dans ce type de roaming, le contrôle de la couche MAC de l'appareil est transféré d'un NS à un autre (*le fNS acquiert le rôle de sNS*). Par conséquent, le NS visité (vNS) est appelé sNS.

Bien que ce type de Roaming reste intéressant et suscite aussi l'intérêt des chercheurs [54], il est hors du périmètre de cette thèse.
En effet, au début de cette passionnante aventure de thèse en 2020, lorsqu'on observait la carte de l'ensemble des pays (121 opérateurs dans 142 pays. En 2023, on est actuellement à 181 pays) qui implémentent la technologie LoRaWAN, seul le passif Roaming était le plus déployé et les choses n'ont pas beaucoup évolué depuis [59]. Le Handover roaming occupe un second rang même dans le document officiel de spécification LoRaWAN 1.1. Cela est dû probablement à plusieurs raisons (La liste n'est pas exhaustive):

- À l'heure actuelle, les serveurs LoRaWAN publics ne prennent en charge que le passif Roaming. Il se pourrait que Roaming Handover devienne disponible dans un futur proche etant donné qu'il intéresse de plus en plus la communauté scientifique.
- Une partie du Roaming Handover nécessite que les End-devices et les serveurs prennent en charge LoRaWAN 1.1, qui est une nouvelle version de LoRa disponible depuis 2018, mais qui n'est pas encore compatible avec tous les End-Device.
- La spécification 1.1, qui introduit le Roaming, présente très peu d'avantages en ce qui concerne le Handover Roaming par rapport au Passif Roaming qui est très bien détaillé. Elle mentionne simplement que le Handover Roaming "ouvre des possibilités de scénarios plus flexibles", sans fournir de détails supplémentaires.

Cependant, le Handover Roaming demeure important, car il permet à un End-device de déterminer s'il est engagé dans une procédure de Roaming ou non (ce qui n'est pas le cas en Roaming passif, où le End-device n'a pas conscience de son statut). Cela donne davantage de contrôle au End-device, notamment la possibilité de gérer sa propre stratégie de Roaming , la gestion de sa batterie ou les coûts associés (si les tarifs varient en Roaming).

7.5 Roaming dans d'autres type de réseau LPWAN

7.6 Roaming dans d'autres type de réseau LPWAN

7.6.1 LTE-M et NB-IoT

LTE-M et NB-IoT [58] sont des réseaux conçus spécifiquement pour l'IoT et sont proposés par les opérateurs de téléphonie mobile. Par conséquent, l'interopérabilité entre ces réseaux est prévue, et l'expérience des opérateurs de téléphonie mobile est un atout majeur dans ce domaine, que ce soit en termes de compétences techniques ou de mise en pratique. Il est envisageable qu'ils puissent tirer parti de leurs accords existants pour faciliter la mise en œuvre du Roaming, de sorte qu'il devienne un élément intégré au réseau LTE-M

Figure 7.2: Couverture LoRa de l'operateur français Orange [36]

et NB-IOT de manière naturelle.

7.6.2 Sigfox

Sigfox [11] est un réseau IoT privé déployé dans de nombreux pays aujourd'hui. Cependant, le défi réside dans le fait que SigFox ne dispose pas d'une couverture mondiale comme LoRaWAN. Bien qu'il offre une couverture relativement étendue en Europe de l'Ouest, en Amérique du Sud et en Australie, il présente des zones non couvertes dans plusieurs pays d'Europe de l'Est, une absence totale de couverture en Russie, dans la plupart des nations asiatiques, ainsi qu'en Afrique.

7.7 Projets implementant le réseau LoRaWAN

Aujourd'hui, nous avons la possibilité de créer notre propre réseau privé en déployant nos propres Gateways ainsi qu'une infrastructure de serveurs dédiée pour établir la communication avec les appareils LoRa. Il existe des solutions open source pour implémenter les serveurs de réseau et d'application LoRaWAN, telles que ChirpStack et The Things Stack, qui peuvent être utilisées à cet effet.

7.7.1 ChirpStack

Le projet Chirpstack LoRaWAN [35] est une solution open–source, très utilisée dans le monde académique. Il vise à fournir une solution complète et flexible pour la gestion des réseaux IoT LoRaWAN, permettant aux utilisateurs de déployer, gérer et exploiter efficacement leurs dispositifs IoT avec une grande facilité.
L'un des atouts de Chirpstack réside dans la capacité de ses différents éléments (GW, NS, JS et AS) à fonctionner de manière autonome, ce qui permet une intégration aisée au sein d'infrastructures préexistantes. Il offre une interface graphique pour simplifier les opérations de configuration, d'intégration et de gestion des appareils IoT. Les éléments

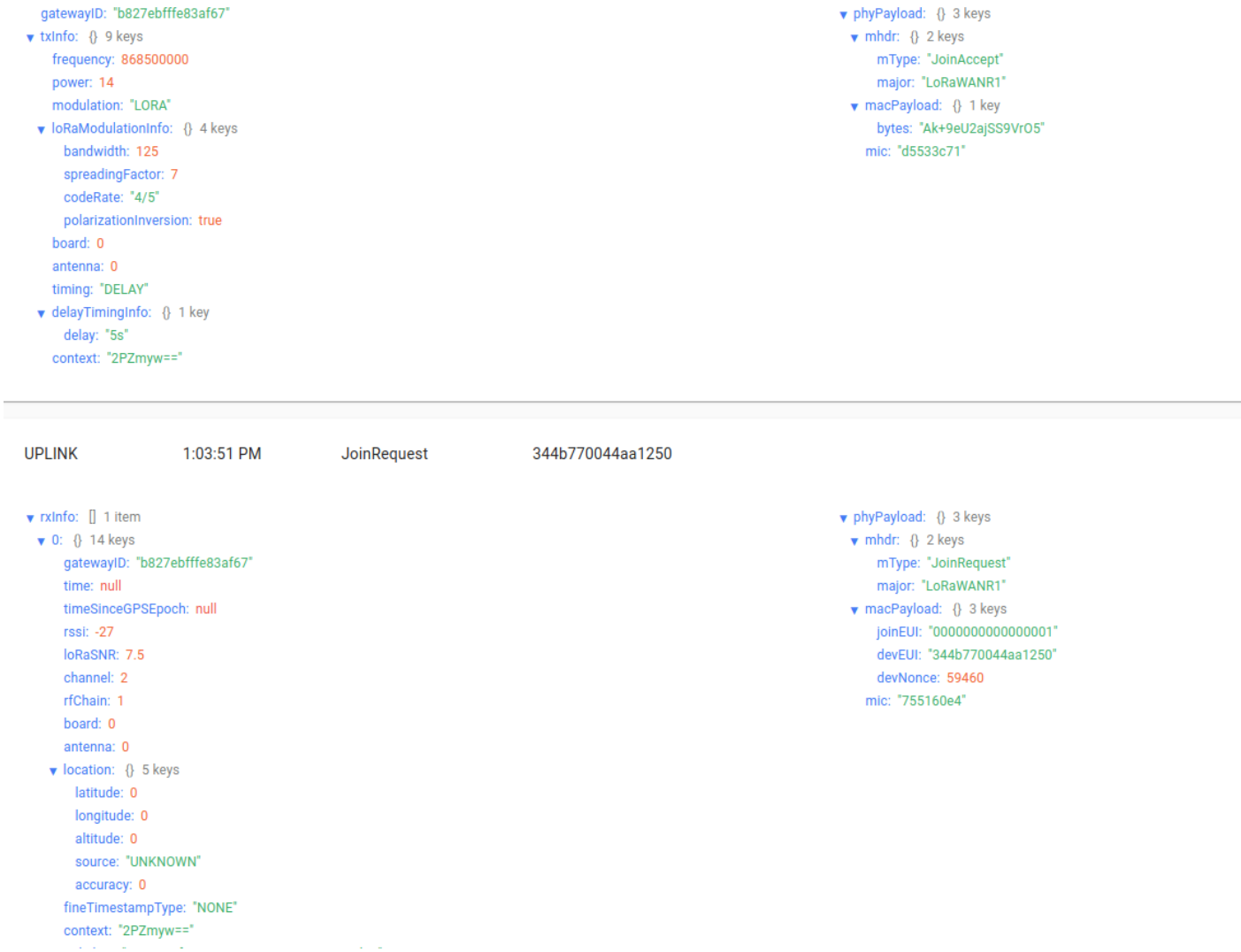


Figure 7.3: Exemple d'interface de l'un de nos Chirpstack Application Server

constituants sont les suivants : a) Chirpstack Gateway Bridge, b) Chirpstack Network Server et c) Chirpstack Application Server (voir figure 7.3).

7.7.2 The Thing Network (TTN)

The Things Network [107] est un réseau collaboratif mondial open-source et privé de l'Internet des objets qui développe des réseaux, des dispositifs et des solutions en utilisant la technologie LoRaWAN.
À l'heure actuelle, ce réseau compte plus de 40 000 contributeurs répartis au sein de plus de 400 communautés dans 90 pays, ayant déployé plus de 4 000 passerelles. Il est ouvert aux particuliers, aux universités et aux entreprises qui souhaitent contribuer à son expansion ou l'utiliser gratuitement.

7.8 Conclusion

Dans ce chapitre introductif de la dernière partie de cette thèse, nous avons mis en évidence l'importance du roaming dans la mobilité des dispositifs IoT. Dans cette optique, nous avons exposé les divers types de roaming existants et examiné les travaux pertinents qui ont contribué à cette question.
Au regard des travaux existants, il est clair que la problématique du roaming IoT, plus spécifiquement le roaming LoRaWAN, suscite un grand intérêt au sein de la communauté scientifique. Dans le chapitre suivant, nous nous pencherons sur le roaming passif au sein des réseaux LoRaWAN et proposerons une solution pour mieux l'optimiser.

Chapter 8

Optimisation de l'architecture LoRaWAN pour la mobilité des objets connectés

It Has To Work.

— **RFC 1925, Nr. 1**
(The Twelve Networking Truths)

Sommaire

8.1 Introduction

En 2017, pour améliorer la couverture du réseau, LoRa Alliance a normalisé une procédure de Roaming permettant à un appareil IoT mobile de rejoindre un réseau étranger. La spécification Roaming LoRaWAN s'appuie sur le DNS [103] pour localiser le serveur de réseau domestique (hNS) [8] de l'ED. Une première requête DNS permet de trouver le Join Server (JS) [88] identifié par le paramètre JoinEUI[1][88] transféré dans le message initial JoinRequest [88]. Le JS fournit le paramètre NetID [88] au sNS qui, par une seconde résolution DNS, détermine le hNS de l'ED, contacte celui-ci qui autorise l'activation de l'ED. Ce dernier point repose sur l'hypothèse sous-jacente que les opérateurs respectivement propriétaires du hNS et du sNS visité ont préalablement défini un accord d'itinérance entre eux. Cette approche met l'accent sur les avantages du DNS dans d'autres domaines que la résolution de noms.
Le DNS est un service robuste, conçu pour être évolutif et résister à la plupart des attaques de sécurité [103]. Dans [21], les auteurs montrent, par le biais d'une preuve de concept expérimentale, qu'une telle approche, tout en offrant une solution possible à l'itinérance LoRa, est toujours en adéquation avec les exigences de latence de LoRaWAN [88]. Néanmoins, nous pouvons souligner certaines limites du scénario initial : (i) les appareils doivent être configurés avec un JoinEUI appartenant au propriétaire de l'appareil. (ii) le propriétaire de l'appareil doit également avoir obtenu une JoinEUI unique (iii) le LNS doit effectuer un grand nombre de requêtes et de RTT avant d'accéder aux informations d'identification de l'appareil.

Dans ce chapitre, nous proposons une approche plus simple, compatible avec l'architecture de roaming LoRaWAN, tout en introduisant deux optimisations principales. La première propose d'utiliser l'identifiant Device Extended Unique Identifier (DevEUI) au lieu de AppEUI[2]/JoinEUI[3] pour réaliser le scénario de Roaming. La seconde introduit l'utilisation de DoH qui permet l'utilisation de certificats dans le DNS. Si un résolveur et un serveur DNS ne partagent pas la même autorité de certification, la requête est bloquée. Nous montrerons que cette méthode permet de réduire le nombre de requêtes dans la phase de Roaming.

8.2 Évolution de LoRaWAN dans un contexte de roaming

LoRaWAN est un réseau asymétrique, où la majeure partie du trafic provient des appareils. Ceux-ci transmettent généralement à un débit de données très réduit. Le réseau suit une topologie en étoile. Les données transmises par l'appareil sont d'abord reçues par un RG, dont la seule fonction est de transmettre les paquets LoRa au serveur de réseau (NS). Les RG ont des restrictions sur le traitement des paquets et une restriction importante sur

[1]Ce paramètre était appelé AppEUI dans la V1.0
[2]Application Extended Unique Identifier (AppEUI)
[3]Join Request Extended Unique Identifier (JoinEUI)

le cycle de travail. Grâce à cela, le coût de mise en œuvre d'une cellule LoRaWAN est très faible par rapport à d'autres réseaux cellulaires. Jusqu'à récemment, l'itinérance dans LoRaWAN était basée sur des développements indépendants pour des implémentations spécifiques. Pour améliorer la couverture du réseau, la LoRa Alliance a normalisé une procédure d'itinérance permettant à un appareil d'utiliser un Visited Nnetwork(VN). Cette procédure est contenue dans les interfaces LoRaWAN Back-end [8].

La figure 8.1 représente l'architecture générale de l'itinérance telle qu'elle figure dans la spécification [8]. Nous avons vu dans le chapitre précèdent qu'un NS peut jouer trois rôles différents en fonction du type d'itinérance concerné. Il s'agit du NS de service (sNS), du NS d'origine (hNS) et du NS de transmission (fNS). Le sNS contrôle la couche MAC de l'appareil, le hNS est l'endroit où le profil de l'appareil est stocké ; il a un lien direct avec le serveur JS qui sera utilisé pour la procédure d'activation des EDs et il est connecté à l'AS. Le fNS est le NS qui gère les RG. Il peut y avoir un ou plusieurs fNS. Lorsque le hNS et le fNS sont séparés, ils ont conclu un accord d'itinérance. Les paquets de liaison montante et descendante sont transmis entre le sNS et le hNS. La spécification décrit les

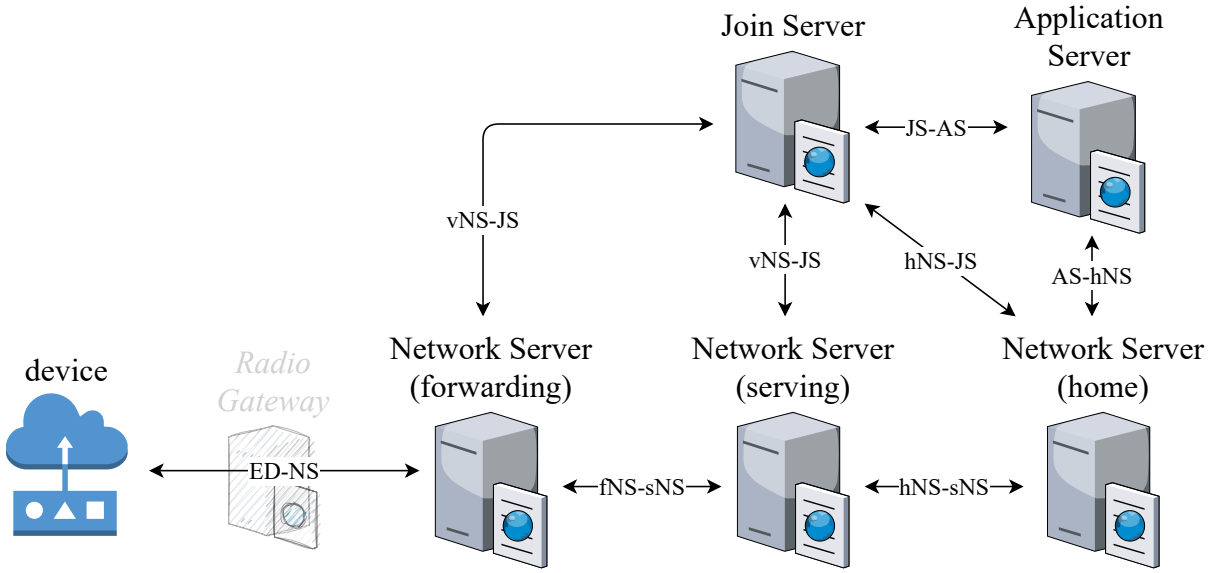


Figure 8.1: Architecture LoRaWAN lorsque l'appareil est en itinérance

procédures d'itinérance pour les sessions LoRa en cours et pour l'activation des appareils.

8.2.1 Procédure d'activation

Dans LoRaWAN, la procédure d'activation vise à délivrer un `DevAddr` à chaque appareil et à créer des clés de session à partir d'une clé partagée à l'aide d'AES. Il existe deux types d'activation des appareils : Activation By Personalization (ABP) et Over The Air Activation (OTAA). Dans le cas de l'ABP, tous les paramètres nécessaires ont été insérés au préalable dans l'appareil, tandis que dans le cas de l'OTAA, ils sont créés de manière dynamique. Pour des raisons pratiques et de sécurité, l'OTAA est préférable à l'ABP.

Pour effectuer une activation OTAA, chaque appareil est doté de trois ou quatre paramètres : deux ID IEEE EUI-64 : un ID d'appareil (`DevEUI`) et un ID JS (`AppEUI/JoinEUI`), et, selon la version de l'appareil, une ou deux clés AES-128 : `AppKey` et/ou `NwkKey`. À la fin de l'OTAA, les informations suivantes sont stockées dans l'appareil :

- `DevAddr` : une adresse de 32 bits utilisée pour identifier le dispositif. Sa valeur se compose de `NwkAddr` (adresse de l'appareil au sein d'un réseau) préfixée par un `NetID`, qui est une valeur de 24 bits fournie par l'Alliance LoRa pour identifier le NO.
- LoRaWAN 1.0 : deux clés de session dérivées du `AppKey` : `AppSKey` pour le chiffrement et `NetSKey` pour l'intégrité.
- LoRaWAN 1.1 : en plus de la `AppSKey`, trois clés de session sont dérivées de la `NwkKey` : `SNwkSIntKey` et `FNwkSIntKey` pour l'intégrité et `NwkSEncKey` pour le chiffrement.

Dans le présent chapitre, nous nous concentrons uniquement sur l'OTAA. Pour l'itinérance passive, comme le montre la figure 8.2, cette procédure fonctionne comme suit :

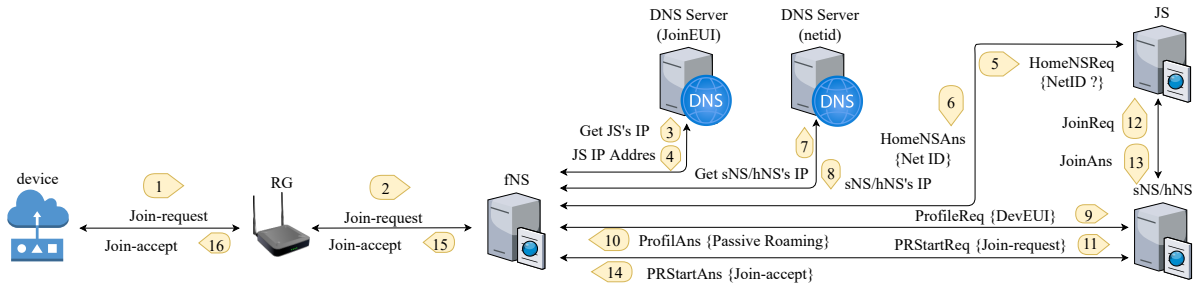


Figure 8.2: Procédure de jonction pour l'itinérance passive dans LoRaWAN.

- L'appareil envoie un `Join-request` (JR). Ensuite, RG transmet la JR au fNS [étapes 1–2].
- Le fNS détermine s'il est configuré pour travailler avec le JS identifié par le `JoinEUI` contenu dans le `Join-request`, en effectuant une requête DNS pour obtenir l'adresse IP du JS [étapes 3–4].
- fNS détermine s'il existe un accord d'itinérance avec le hNS/sNS concerné en envoyant un message `HomeNSReq` au JS. S'il existe, le JS répond par un `HomeNSAns` avec le `NetID` [étapes 5–6].
- fNS utilise le DNS pour rechercher l'adresse IP du hNS/sNS sur la base du `NetID`. Cette étape peut être évitée si les informations sont stockées dans le cache DNS. [étapes 7–8].
- fNS envoie un message `ProfileReq` au hNS/sNS contenant `DevEUI` [étape 9].
- hNS/sNS envoie un message `ProfileAns` indiquant le type de profil d'itinérance. Dans ce cas, il correspond à l'itinérance passive [étape 10].
- fNS envoie un message `PRStartReq` contenant le message `Join-request`. Ensuite, le hNS/sNS transmet le message `JoinReq` au JS [étapes 11-12].
- JS répond par un message `JoinAns` contenant le message `Join-accept` [étape 13].
- sNS/hNS envoie un message `PRStartAns` au SNF contenant le `Join-accept` [étape 14].
- Le fNS envoie finalement un `Join-accept` à l'appareil [étapes 15–16].

Comme indiqué plus loin dans la section 8.5, il convient de noter qu'une flèche peut représenter plusieurs échanges, soit dans les requêtes DNS pour localiser le serveur DNS effectif, soit dans les communications HTTPS pour ouvrir la connexion TCP et les clés TLS.

8.2.2 Résolutions DNS

Comme décrit dans la section 8.2.1, le service DNS est appelé deux fois. Une requête DNS trouve le JS associé au `JoinEUI` (figure 8.2 étapes 3-4), et l'autre, le sNS à partir du `NetID` renvoyé par le JS (étapes 7-8). Donc deux zones DNS : `NETIDS.lorawan.net` et `JOINEUIS.lorawan.net`. Les suffixes DNS sont obtenus comme suit :

- Le `JoinEUI` représenté au format hexadécimal est d'abord inversé. Ensuite, des points sont insérés entre chaque point et le nom de domaine.
- Le `NetID` (identificateur unique de 24 bits) du sNS.

Rappelons que LoRa Alliance est l'organisation qui gère les serveurs DNS faisant autorité et qui maintient les fichiers de zone pour les deux résolutions. Par conséquent, tout NO souhaitant bénéficier de l'itinérance doit obtenir un `NetID` de la LoRa Alliance, et chaque appareil itinérant doit être doté de `DevEUI` et de `JoinEUI` : `DevEUI`, `JoinEUI` et `AppKey/NwkKey`. L'architecture LoRaWAN actuelle repose sur la certification publique pour authentifier les JS et les fNS lors des échanges TLS (étapes 5-6 et 9-10 de la figure 8.2). Dans la configuration expérimentale initiale de IoTRoam[21], il a été proposé que l'Alliance LoRa agisse en tant qu'autorité de certification racine et génère des autorités de certification intermédiaires pour chaque réseau. Les AC intermédiaires généreront à leur tour des certificats feuilles pour les serveurs individuels (JS, NS, AS).

8.2.3 Problématique de roaming dans les réseaux LoRaWAN

Si la procédure de Roaming décrite par la LoRa Alliance fonctionne et permet une itinérance efficace des objets, certains problèmes d'évolutivité subsistent. Si l'attribution du `DevEUI` est effectuée par le fabricant de l'appareil, le Device Owner(DO) est responsable de l'attribution du `JoinEUI`. Le `DevEUI` est statique, alloué lors de la fabrication du dispositif et reste le même pendant toute la durée de vie du dispositif. Le `JoinEUI` doit être modifié lorsqu'un appareil est acheté ou vendu à un autre client.

- DO doit obtenir un `JoinEUI` valide et unique. Conformément à la spécification, ces identificateurs sont attribués par l'IEEE sur la base de l'EUI-64, en tant que premiers 24, 28 ou 36 bits. Les bits restants créent des valeurs uniques au niveau mondial. Cela n'est pas compatible avec le nombre de `JoinEUI` dont une entreprise aura besoin. Malgré les coûts, le processus d'attribution est défini par l'IEEE[1] et vise les fabricants d'objets, et non les DO. LoRa Alliance peut gérer les allocations de `JoinEUI`, mais cela engendre des frais administratifs généraux.
- Le DO doit insérer cette valeur `JoinEUI` dans son appareil. Cela implique l'installation d'un nouveau micrologiciel sur l'appareil ou une procédure d'approvisionnement. Cela peut augmenter les coûts de l'appareil avec un logiciel plus complexe.

[1]IEEE : Institute of Electrical and Electronics Engineers

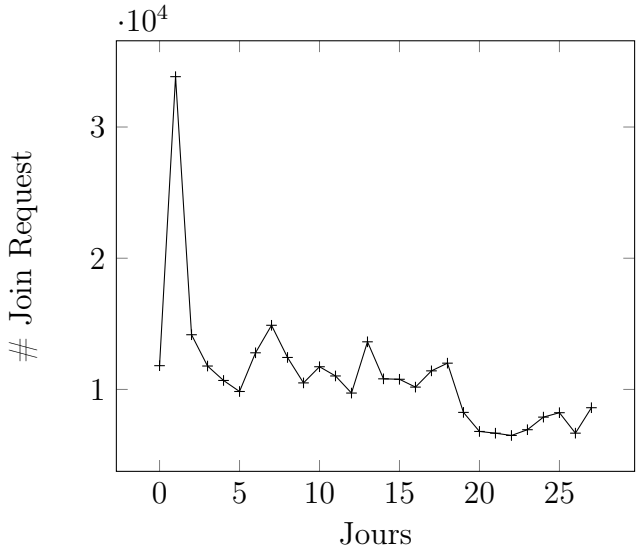


Figure 8.3: # Join request par jour dans la ville de Rennes.

- La zone DNS est gérée par la LoRa Alliance pour assurer la correspondance entre `JoinEUI` et JS ou `NetID` et NS. La LoRa Alliance doit contrôler qui met à jour l'enregistrement DNS. Cela implique une procédure d'authentification.
- Si le JS est interne au NO, seul le NS y accèdera pour authentifier les dispositifs. Avec la procédure d'itinérance, le fNS interroge le JS pour obtenir le NO `NetID` *(étapes 5-6] de la figure 8.2)*. Cela peut poser des problèmes de sécurité, car le JS peut ne pas être en mesure de prévoir quel fNS enverra une requête et exposera un élément sensible à l'internet.

Pour les DO disposant de grandes flottes d'objets susceptibles d'effectuer une itinérance passive régulière, cette approche, tout en restant peu sûre, coûteuse et difficile à mettre en œuvre, peut sérieusement limiter les avantages économiques qui pourraient être tirés des services exploitant les avantages de la mobilité des objets connectés.

Trafic Join Request: La figure 8.3 donne le nombre de trames Join Request, qu'un RG reçoit dans la ville de Rennes. Nous pouvons dire qu'un jour ordinaire, environ 10000 Join Requests sont envoyées par les appareils. Le deuxième jour de mesure, un nombre maximum de 33000 Join Requests a été reçu. Cela permet d'avoir une idée du trafic généré par les JoinRequests. Si nous prenons le nombre le plus élevé, nous obtenons une moyenne de 0,4 demande d'adhésion par seconde.

8.3 Architecture de roaming proposée

Le principal point technique concernant l'itinérance passive est de permettre à n'importe quel fNS de déterminer le sNS de n'importe quel appareil itinérant aussi simplement que possible. Nous proposons pour cela une extension de l'architecture IoTRoam en introduisant une nouvelle entité appelée ***DNS-Broker*** dont le rôle décrit dans le chapitre

6, est de fournir en toute sécurité ces informations aux entités qui en font la requête. La communication avec le DNS-Broker est protégée par l'utilisation de certificats attribués individuellement par celui-ci à DO et NO. Ces certificats peuvent être limités dans le temps et constituer la base d'un service commercial.

Le concept de courtier introduit une sorte de capacité de résolution DNS privée et sécurisée réservée principalement aux DO en tant qu'acteurs économiques impliqués dans les services IdO basés sur la mobilité des objets. Les DO concernés par ces questions peuvent simplement transmettre ces informations au DNS-Broker. Ils doivent enregistrer le `DevEUI` de leurs objets dans une zone DNS privée et la synchroniser avec le courtier en utilisant la capacité de transfert de zone. Le courtier utilise cette zone pour créer sa propre zone qui pointera vers le NS du DO. Le courtier disposera ainsi d'une liste complète de `DevEUI`, ce qui peut conduire à des milliards d'entrées, mais les DNS avec des zones telles que `.com` sont capables de gérer un tel volume d'entrées.

Le principe de l'itinérance passive peut être simplifié comme suit : lorsqu'un nouveau dispositif tente de rejoindre un NO par le biais de l'itinérance passive, le fNS concerné n'a qu'à interroger le courtier pour déterminer le bon sNS.
La procédure de jonction détaillée est présentée dans la figure 8.4, et se déroule comme suit :

- L'appareil envoie un `Join-request` (JR). RG transmet le JR à son NS (le fNS) [étapes 1–2].
- Le fNS détermine s'il existe un accord d'itinérance avec le NO du hNs en effectuant une requête DNS portant le `DevEUI` pour obtenir le hNS/sNS `NetID` et l'adresse IP (voir la section 8.3.1 pour plus de détails) [étapes $3' - 4'$ et $7' - 8'$]. Les étapes 5-6 de la Fig. 8.2 qui ne sont plus nécessaires, sont supprimées.
- Le fNS envoie un message `ProfileReq` au hNS/sNS contenant `DevEUI` [étape 9].
- Le hNS/sNS envoie un message `ProfileAns` indiquant le type de profil d'itinérance. Dans ce cas, il correspond à l'itinérance passive [étape 10].
- Le fNS envoie un message `PRStartReq` contenant le message `Join-request`. Ensuite, le hNS/sNS transmet le message `JoinReq` à son JS [étapes 11-12].
- Le JS répond par un message `JoinAns` contenant le message `Join-accept` [étape 13].
- Le hNS/sNS envoie un message `PRStartAns` au fNS contenant le `Join-accept`. Ensuite, le fNS envoie un `Join-accept` au dispositif [étapes 14–16].

8.3.1 Utilisation du DNS-Broker dans les reseaux LoRaWAN

Comme le montre la figure 8.4, l'architecture d'itinérance proposée utilise le DNS-Broker présenté au chapitre 6. Par conséquent, chaque fois qu'un message `Join-request` transportant un `DevEUI` inconnu arrive au fNS, ce dernier effectue une requête DNS contenant le `DevEUI`. Le fNS effectue une requête DNS contenant le `DevEUI` pour obtenir le `NetID` correspondant et l'adresse IP du hNS. Ces deux valeurs sont utilisées ultérieurement pour établir une connexion IP entre les deux NS.

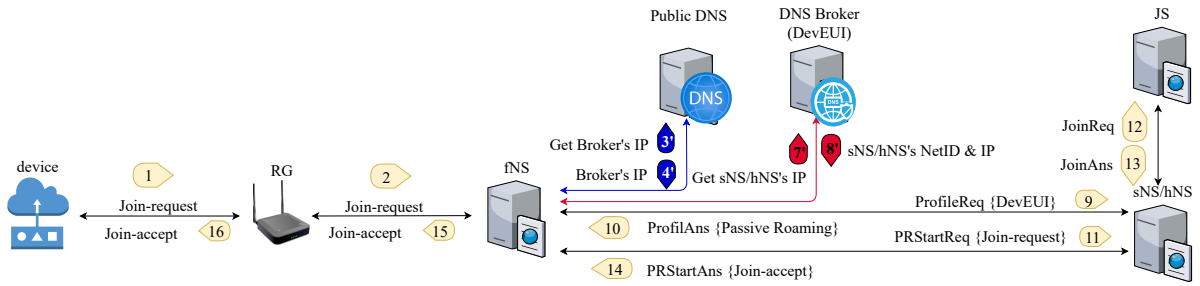


Figure 8.4: Architecture d'itinérance : Procédure d'adhésion dans l'architecture d'itinérance proposée

Cette résolution DNS est divisée en deux parties : une partie publique où nous résolvons l'IP d'un courtier DNS, et une partie privée où le `DevEUI` est résolu en un `NetID/adresse IP`. Ce faisant, nous contournons les résolutions `JoinEUI` afin que le NO réduise le coût des déploiements de réseaux et simplifie la phase de provisionnement des appareils.

La deuxième partie de la résolution est prise en charge par le **DNS Broker**. Son rôle principal consiste à gérer les zones DNS `DevEUI` et `NetID` et de délivrer des certificats côté client. Il est donc défini comme une entité située entre le NO et le DO. Le DO enregistre ses dispositifs auprès du Broker. Le NO interroge le Broker lorsqu'un nouveau dispositif rejoint le réseau pour l'autoriser ou non sur son réseau.

La figure 8.5 représente schématiquement l'architecture globale avec le Broker. DO ajoute dans une base de données les `DevEUI` des dispositifs qui doivent être pris en compte. Cette base de données est synchronisée avec celle du Broker. Lorsqu'un nouveau dispositif apparaît sur un NO et envoie un message `Join-request`, au lieu d'utiliser le `JoinEUI` pour localiser le sNS. Le fNS contacte le Broker pour obtenir l'adresse du sNS et poursuit le protocole d'adhésion comme d'habitude.

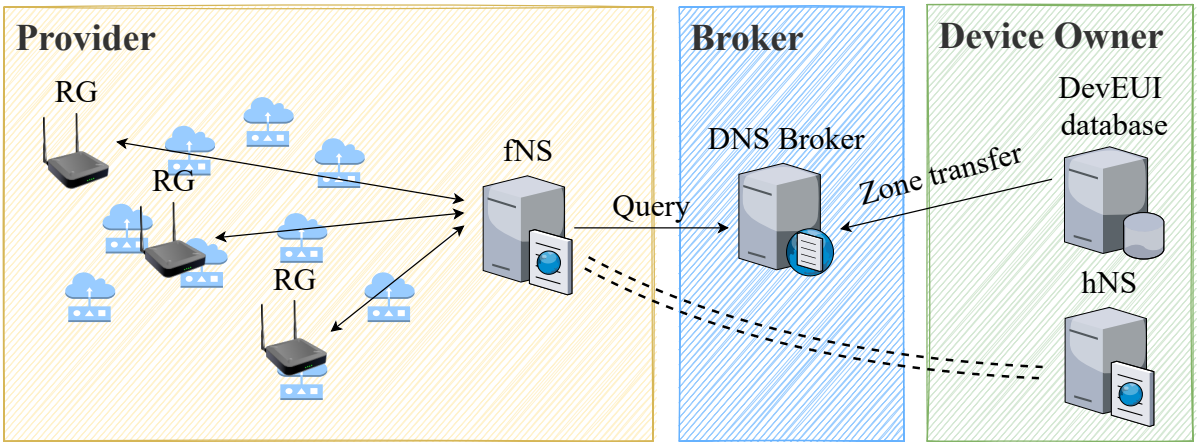


Figure 8.5: Architecture du DNS-Broker appliquée au roaming LoRaWAN

Les protocoles LoRaWAN ne sont pas fortement modifiés, seule la mise en œuvre du fNS doit être légèrement modifiée pour utiliser l'identifiant `DevEUI` au lieu de `JoinEUI`.

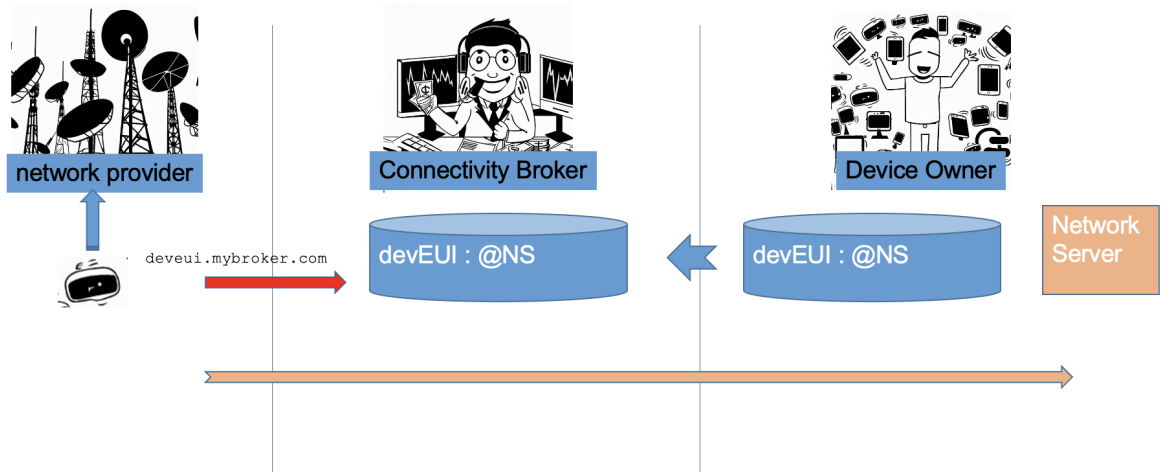


Figure 8.6: Scénario d'enregistrement des End-Devices ou d'abonnement auprès du DNS-Broker

Contrairement à l'approche originale de l'itinérance, qui impliquait de centraliser certaines informations dans la zone DNS, à savoir les registres `NetID` et `JoinEUI`, cette approche est plus souple.

Les besoins de notre PoC nous ont amenés à concevoir un seul DNS Broker centralisé. Toutefois, les exigences fonctionnelles générales du DNS Broker et leurs incidences sur son architecture méritent un traitement approfondi.

Nous pouvons en effet facilement imaginer des cas d'utilisation pertinents impliquant une multiplicité de catégories de DNS-Brokers obéissant à des organisations différentes (structures hiérarchiques comme DNS, fédérations comme EduRoam, approche décentralisée peer-to-peer ou îlots indépendants). Inversement, un DO peut enregistrer ses dispositifs auprès de plusieurs DNS-Brokers et un NO peut envoyer simultanément sa requête à plusieurs DNS-Brokers.

8.3.2 Gestion des certifications pour l'itinérance

Comme nous l'avons présenté au chapitre 6, dans cette nouvelle architecture, nous proposons une fourniture décentralisée ou maillé de certificats clients, où il y aura deux types de certificats : (i) des certificats d'itinérance, utilisés par les LNS pour établir des connexions HTTPS avec le DNS Broker et (ii) des certificats LNS, utilisés par chaque LNS pour établir des connexions HTTPS entre eux.

En ce qui concerne la fourniture de certificats d'itinérance, le schéma général est présenté dans la figure 8.7. Comme nous pouvons le voir, le DNS-Broker agit en tant que CA et délivre des certificats clients à chaque LNS. Le DNS-Broker est ensuite chargé d'authentifier les différents LNS, ce qui permet d'avoir des requêtes DNS privées.

Il convient de noter qu'une entreprise peut faire partie de plusieurs fédérations et qu'il est

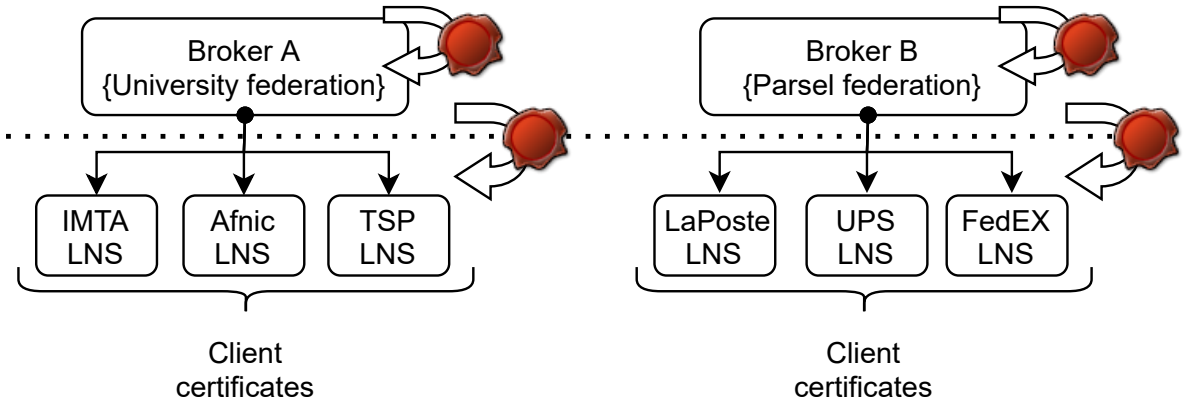


Figure 8.7: Approvisionnement en certificats pour les connexions HTTPS entre le client et le DNS-Broker

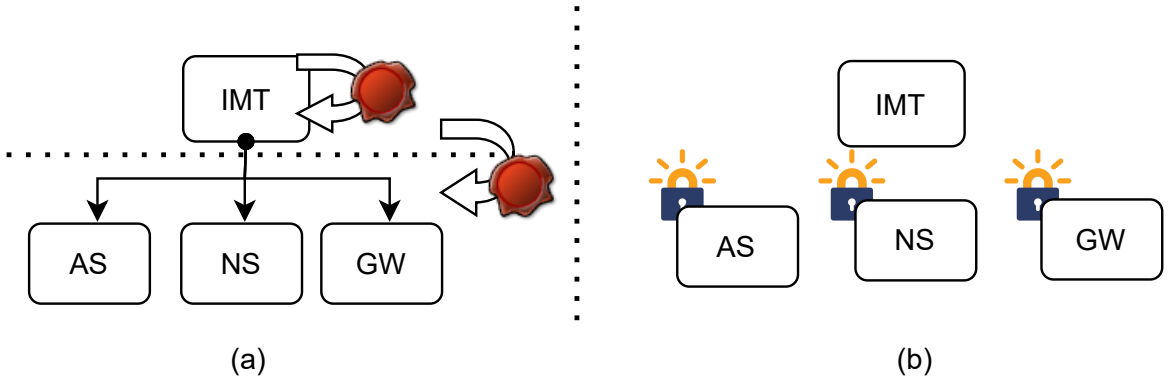


Figure 8.8: Fourniture de certificats pour les communications HTTPS entre LNS en itinérance

même possible de prévoir le transfert de bases de données entre différents DNS-Broker utilisant des PTR(Pointer records ou reverse DNS).

Par contre, pour les certificats nécessaires pour les connexions HTTPS entre LNS, nous proposons que cela soit fait indépendamment et selon l'accord de roaming qui est fait entre les opérateurs, de cette façon, contrairement à l'architecture actuelle, cela sera complètement indépendant d'une association comme la LoRa Alliance.

Une proposition pour cela pourrait être, comme présenté dans la Figure 8.8, l'utilisation de let's encrypt ou même de certificats auto-signés ou, pourquoi pas, une combinaison des deux, afin de protéger, le nom de domaine et de faire de l'authentification mutuelle en utilisant des certificats de serveur et de client.

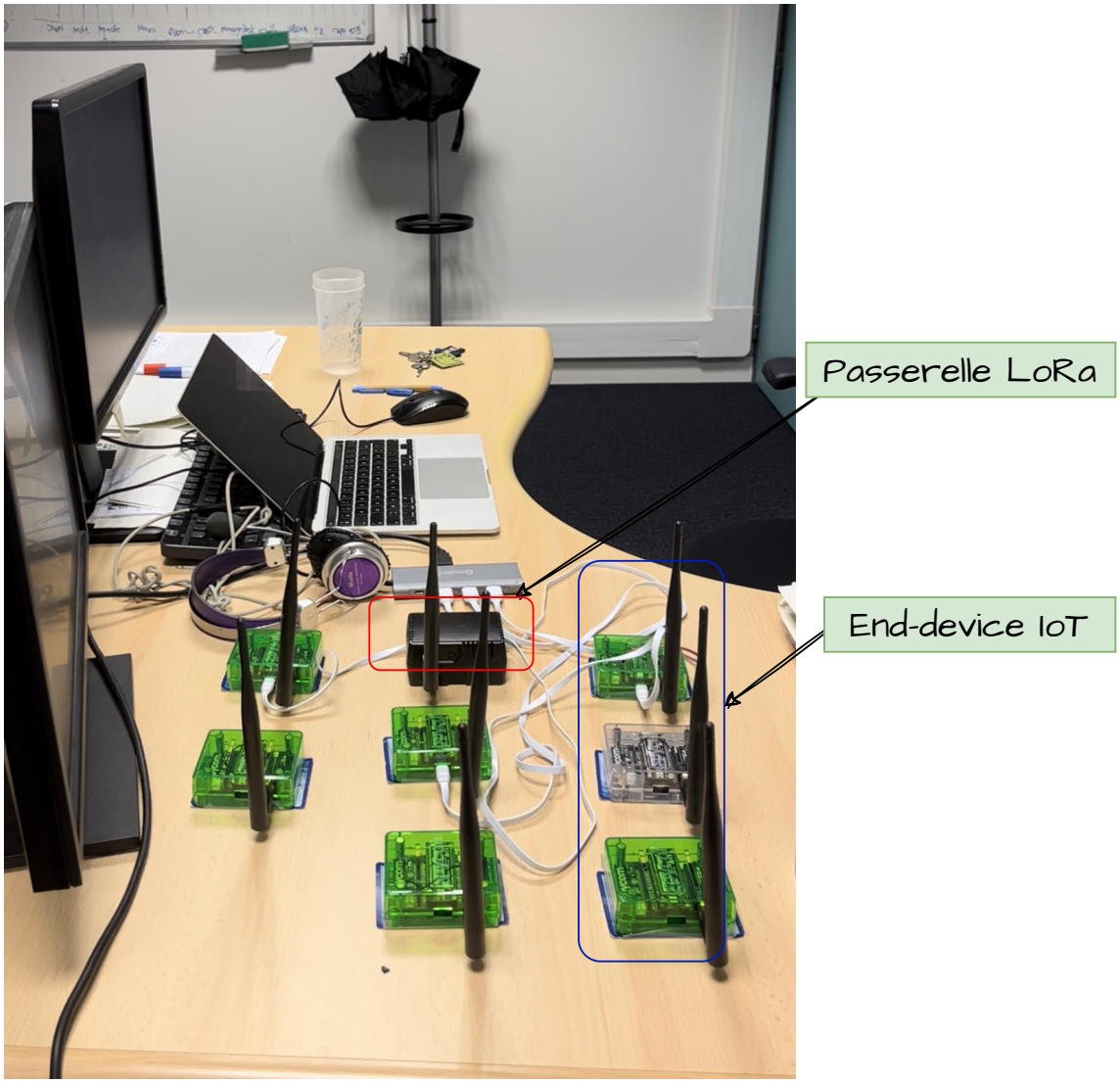


Figure 8.9: Mise en œuvre de la plate-forme d'itinérance LoRaWAN

8.4 Mise en œuvre à travers le PoC

L'architecture d'itinérance décrite ci-dessus a été testée et mise en œuvre comme suit :

1. **NS** une version modifiée de Chirpstack avec un client DoH dnsproxy supportant l'authentification du client.
2. **DNS Broker** un serveur physique avec une IP publique et un domaine enregistré : `broker.iot-roam.net`.

Le diagramme de sequence de la solution d'itinérance passive proposée, détaillant le rôle du DNS-Broker, est représenté dans la figure 8.10. Comme nous pouvons le voir, une fois que le JR arrive au fNS, il fait une requête DNS régulière au résolveur DNS local. La réponse qu'il reçoit correspond à l'IP du courtier DNS.

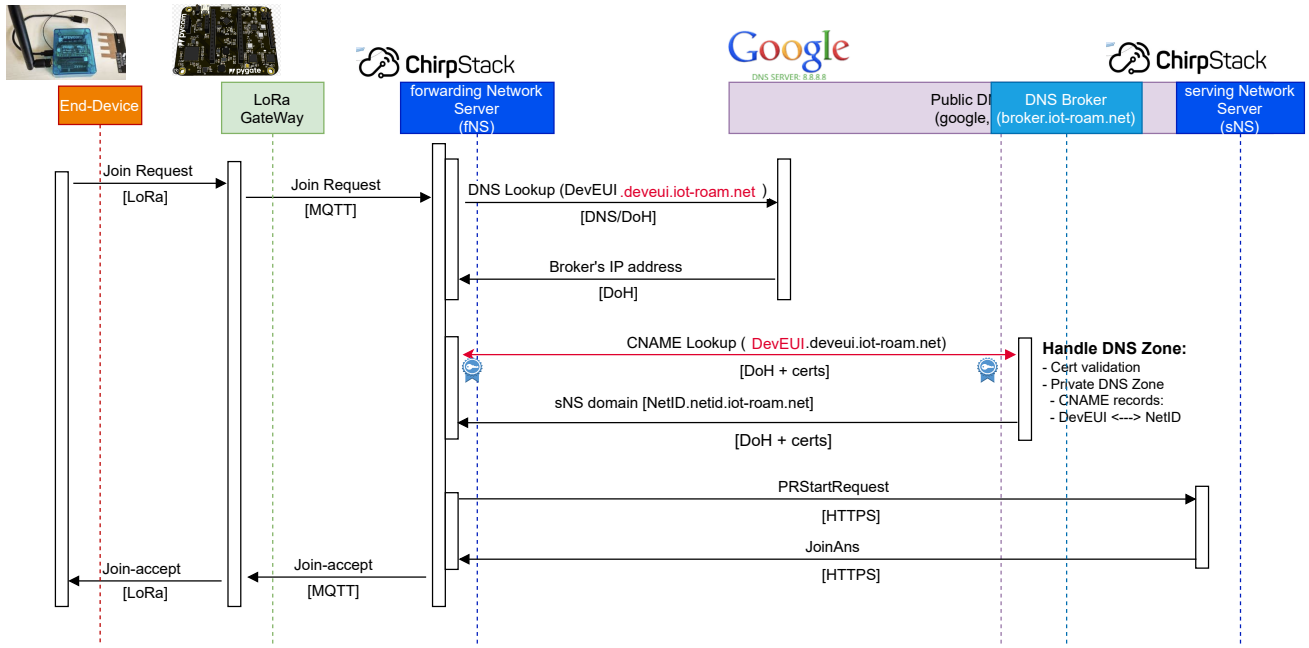


Figure 8.10: Diagramme de l'architecture de l'itinérance : Procédure de jointure avec le DNS-Broker

En interne, le fNS identifie que cette IP correspond au DNS-Broker qui a été précédemment configuré comme résolveur DoH pour ce domaine particulier. Il effectue donc une requête DoH privée auprès du DevEUI afin d'obtenir les informations nécessaires à l'établissement d'une connexion IP avec le hNS. Pour cela, il démarre une connexion HTTPS avec le DNS Broker en utilisant le certificat côté client qui a déjà été fourni dans le fNS.

Du côté du DNS-Broker, la première chose qu'il fait est de valider le certificat du fNS, puis de vérifier si le DevEUI demandé a été enregistré dans sa base de données par le propriétaire de l'appareil. Si c'est le cas, il répond avec l'enregistrement CNAME correspondant. Cet enregistrement est composé du nom de domaine du hNS et de son adresse IP

Le code source, ainsi qu'un tutoriel sur la mise en œuvre de notre plateforme d'itinérance telle que présentée partiellement à la figure 8.6 et la figure 8.9 sont disponibles en Annexe.

8.5 Évaluation des performances

Pour évaluer les performances de notre architecture, nous avons analysé le trafic via le fNS pendant toute la durée du processus d'itinérance pour la procédure de jonction.

Ce faisant, nous : (i) nous déterminons le volume de trafic IP, DNS et UDP dans un paquet DNS lorsque le fNS veut obtenir l'adresse IP du JS/hNS, et (ii) nous déterminons également le volume de trafic IP, TCP et TLS dans les échanges HTTPS.

Les résultats présentés dans le Tableau 8.1 concernent le trafic requis dans IoTRoam par rapport à notre proposition pour chaque protocole dans les trois catégories suivantes : (i) DNS, correspondant au trafic requis pour obtenir l'IP des différents serveurs (JS, NS, Broker), (ii) HTTPS, correspondant au trafic généré par l'échange de messages entre les serveurs, et (iii) la granularité TLS, qui est le volume de trafic requis pour la poignée de main et les données. Nous comptons également le nombre total de paquets nécessaires pour rejoindre un dispositif ainsi que le nombre de RTT (round-trip delay time).

Table 8.1: Évaluation des Performances

Types	Parameters	**IoTRoam** JR ($n=1$)		JR ($n>1$)		**proposed architecture** JR ($n=1$)		JR ($n>1$)	
		#	Len. [B]	#	Len. [B]	#	Len. [B]	#	Len. [B]
DNS	IP	8	160	0	0	4	80	0	0
	UDP	8	56	0	0	4	32	0	0
	DNS	8	606	0	0	4	332	0	0
HTTPS	TCP	21	1140	3	156	27	1452	3	152
	Sig'								
	TLS	15	18167	6	2811	24	25788	6	2793
TLS Gran.	IP	15	300	6	120	24	480	6	120
	TCP	15	480	6	192	24	768	6	192
	C. Hello	3	858	0	0	3	870	0	0
	S. Hello	3	8198	0	0	5	10862	0	0
	C. Cipher	3	5775	0	0	3	6166	0	0
	App. Data	6	2556	6	2499	13	6642	6	2481
# of packets		44		9		55		9	
RTTs		21		6		24		6	

'(SYN – SYN/ACK – ACK)

Pour l'évaluation des performances, nous avons effectué plusieurs JR (Join Requests). Le premier JR ($n = 1$) indique la première tentative d'un appareil de se joindre au VN. Ensuite, le JR ($n > 1$) désigne un JR ultérieur émanant du même appareil. Cela nous permet de voir comment les deux plateformes réagissent lorsque le cache DNS est plein.

JR($n = 1$)
En ce qui concerne le trafic DNS, nous notons une diminution du nombre d'échanges. En effet, il passe de 8 à 4. Cela est dû à la suppression de la deuxième requête DNS où le fNS obtient l'adresse IP du sNS à partir du `NetID`. En revanche, en ce qui concerne HTTPS, nous constatons une augmentation due à l'introduction de certificats serveur/client pour l'authentification mutuelle avec le Broker. En effet, le nombre de paquets passe de 15 à 24. Cela se reflète dans la charge totale de trafic TLS, qui passe de 18167 octets à 25788 octets.

JR ($n > 1$)
Comme prévu, il n'y a pas de changement significatif d'une architecture à l'autre pour les JR ($n > 1$). Les échanges DNS restent identiques pour les deux architectures (0). En ce qui concerne HTTPS, la charge TLS est presque identique, et le nombre de paquets TLS est de 6.

8.6 Conclusion

Nous avons défini une architecture flexible, évolutive et sécurisée pour l'itinérance dans les réseaux LoRaWAN nécessitant des modifications mineures du protocole LoRaWAN ou DNS. Le processus de localisation des sNS peut être simplifié en utilisant, en plus du DNS, une nouvelle entité appelée DNS Broker permettant une résolution privée reposant sur le DoH. Ce sont les clés de l'évolutivité et de la sécurité apportées par notre approche. Cela confirme également le nouveau rôle que le DNS pourrait jouer pour tirer parti de sa robustesse et de sa fiabilité.

Le DNS Broker mérite cependant une étude future dédiée. Les mesures de performance sont légèrement pénalisées par l'utilisation de HTTPS. Cependant, d'autres approches plus légères telles que DNS over CoAP [15] seront considérées comme une tentative pertinente d'alléger l'empreinte du protocole tout en conservant la même fonctionnalité. Le protocole CoAP permettrait non seulement de réduire l'impact de HTTPS sur notre architecture mais aussi nous permettra d'utiliser le format e-CBOR proposé au chapitre 5. Une expérience publique de cette nouvelle architecture de roaming passif LoRaWAN avec plusieurs universités partenaires sera ouverte.

Chapter 9

Conclusion et Perspectives

For all resources, whatever it is, you need more.

— **RFC 1925, Nr. 9**
(The Twelve Networking Truths)

Sommaire

9.1 Problème

Le système de noms de domaine peut être conceptualisé comme un système distribué qui améliore l'interopérabilité des protocoles sur Internet en utilisant une technique d'identification des ressources. L'extension de son utilisation à d'autres domaines d'application, tels que la sécurité, le edge computing et la mobilité des objets connectés, qui sont des domaines de recherche émergent, s'avère très complexe. Cela s'explique par le fait que ce système a été initialement conçu uniquement pour traduire les noms de domaine en adresses IP à des fins d'identification.

L'objectif principal de notre thèse était de minimiser l'influence négative du paradigme REST sur les performances du DNS, afin de rendre possible son utilisation dans d'autres contextes que la simple résolution de noms. Pour atteindre cet objectif, nous avons défini deux questions de recherche distinctes : la première concernant l'optimisation du format de représentation des données du DNS, qui manquait de flexibilité pour d'éventuelles évolutions (QR1), et la seconde se focalisant sur la résolution privée de noms de domaine appliquée à la mobilité des objets connectés (QR2).

9.2 Résultats

Au cours de notre travail, notre objectif premier a été de comprendre en profondeur le fonctionnement de l'infrastructure DNS et l'impact du paradigme REST sur ses performances car nous avons estimé que cette compréhension était essentielle pour aborder de manière efficace le défi de son expansion vers de nouveaux domaines d'application. Par conséquent, nous avons entamé notre recherche par une analyse globale détaillée, en identifiant les obstacles à surmonter et en explorant les technologies de l'Internet des objets susceptibles de contribuer à lever ces verrous.

Notre travail sur la QR1 nous a conduit à concevoir un format de représentation des données DNS plus compact et flexible, que nous avons nommé e-CBOR. Cette flexibilité offre la possibilité d'utiliser le DNS pour stocker divers types d'informations en fonction des besoins des utilisateurs, tels que les clés de sécurité dans le contexte du edge computing et les identifiants des objets connectés au sein des réseaux LPWAN. Les données dans ce nouveau format sont encodées en utilisant CBOR, une technologie adaptée à la représentation de données dans des environnements contraints. Nous avons fait ce choix afin de maintenir la compatibilité avec le format DNS existant.

La protection de la vie privée des utilisateurs sur le web est devenue une préoccupation de recherche majeure en matière de sécurité, en particulier depuis les révélations d'Edward Snowden. Aujourd'hui, un domaine en pleine expansion est celui de l'Internet des objets (IoT). Pour répondre à la QR2, nous avons développé une architecture appelée DNS-Broker, qui vise à restreindre l'accès à l'espace de noms de domaine des utilisateurs. Pour démontrer la pertinence de notre architecture, nous l'avons appliquée au contexte de la mobilité des objets connectés.

9.3 Discussions

Les concepts e-CBOR et DNS-Broker développés dans le cadre de cette thèse apportent des solutions significatives au problème de l'extension des fonctionnalités du DNS à de nouveaux domaines émergents, en particulier ceux qui utilisent le modèle architectural REST pour représenter et transférer leurs données. Dans cette section, nous souhaitons engager une discussion ouverte et une réflexion sur les limites de notre travail.

Tout d'abord, il est important d'expliquer ce qui nous a incités à mener des recherches sur le DNS et par ricochet sur la mobilité des appareils IoT. Lors de l'examen de l'état de l'art de cette thèse, nous avons constaté que les références scientifiques concernant l'utilisation du DNS dans d'autres domaines étaient très limitées, notamment pour le stockage et la représentation des informations dans des domaines émergents nécessitant une faible latence tels que l'IoT et le edge computing . Il nous a donc semblé nécessaire de consacrer une partie de notre travail à cet axe de recherche.

Ensuite, les performances du DNS repose principalement sur le format d'encodage de ses messages. À l'heure actuelle, deux approches existent : l'encodage binaire (Wireformat) et l'encodage textuel (JSON). Nous avons montré dans l'état de l'art que ces approches présentent chacune leurs avantages et leurs limitations. Notre objectif dans cette étude était donc de trouver un format capable de combiner les performances des deux approches, ce qui nous a conduit au format CBOR.

Le format CBOR que nous avons choisi pour développer e-CBOR utilise un encodage binaire, ce qui offre une compacité effective dans la représentation des messages sur notre nouveau format. De plus, il est compatible avec le format JSON, ce qui permet à l'utilisateur d'ajouter des champs en fonction de ses besoins, offrant ainsi une certaine flexibilité.
Il est important de noter que les critères d'évaluation des performances de e-CBOR et les mesures obtenues sont spécifiques à notre environnement de travail et ne prétendent pas être exhaustifs ni strictement représentatifs de la réalité. Une expérimentation sur des serveurs de noms réels reste notamment à réaliser. Néanmoins, ce nouveau format nous a permis de démontrer que l'ajout ou la suppression de certains champs ou sous-champs dans le format DNS actuel en fonction des besoins des utilisateurs reste possible.

Enfin, nous avons également abordé la question de la sécurité de l'infrastructure DNS. Nous savons que le DNS a été développé sans tenir compte des problèmes de sécurité. Dans l'état de l'art, plusieurs travaux ont abordé ces questions, proposant différentes méthodes d'encapsulation des messages DNS dans des protocoles plus sécurisés, tels que TLS, DTLS et HTTPS. Cependant, ces solutions ne protègent pas l'espace de noms ni le serveur de noms contenant les ressources demandées par les utilisateurs. C'est pourquoi nous avons proposé DNS-Broker, qui impose aux utilisateurs de disposer des clés de sécurité pour accéder à certaines zones contenant des ressources critiques. Notre approche a été utilisée pour résoudre le problème de la mobilité des appareils IoT.

La mobilité des objets connectés suscite un intérêt croissant auprès des chercheurs, car

les objets connectés que nous concevons aujourd'hui sont destinés à se déplacer, et la possibilité de rester en communication est cruciale. C'est pourquoi nous avons choisi d'appliquer l'approche DNS-Broker à ce domaine spécifique. Nos expérimentations ont été menées dans un environnement réel.

Cette thèse s'inscrit dans le cadre du projet DiNS, un consortium réunissant des acteurs industriels et des instituts académiques tels que l'IMT Atlantique - IRISA, Afnic, Bouygues Telecom, ACKLIO et INP Grenoble. Ce consortium a été formé en réponse à l'appel à projets de l'ANR 2019, dans le but de briser les cloisonnements entre les réseaux IoT et de les rendre compatibles avec Internet en exploitant le DNS et ses extensions avancées de sécurité. Le problème de la mobilité des appareils IoT que nous avons étudié dans cette thèse s'inscrit donc dans la continuité des travaux réalisés par l'équipe de l'Afnic dans le cadre du projet DiNS.

9.4 Perspectives

Les travaux présentés dans cette thèse apportent une contribution au domaine de la sécurité des protocoles réseaux et de l'Internet des objets. Nos contributions ont ouvert la voie à de nouvelles pistes de recherche, ouvrant ainsi la porte à plusieurs perspectives futures.

DNS over CoAP (DoC) [83] est une proposition récente visant à utiliser CoAP et OSCORE pour la résolution de noms. Si CoAP est utilisé principalement pour permettre le chiffrement des données avec OSCORE, la charge utile reste la requête DNS d'origine. Le premier défi consistait à transformer les structures de données DNS en structures de données CBOR afin de bénéficier de leur compacité. ce que nous avons fait en proposant la méthode e-CBOR. Le deuxième défi consistera à évaluer la possibilité d'utilisation de C509 [93] pour permettre des extensions DNS privées comme nous l'avons proposé dans ce projet.

Nous envisageons également de procéder à une évaluation pratique de la méthode e-CBOR dans un environnement réel. Cette approche nous permettra d'obtenir une évaluation approfondie de la complexité de notre solution, tout en identifiant ses limites et en apportant les corrections nécessaires. Une piste de recherche qui suscite un intérêt scientifique est l'utilisation de e-CBOR pour réduire la latence dans les domaines exigeant en terme de performance comme l'edge computing.

En ce qui concerne notre contribution sur la mobilité des appareils IoT, actuellement, notre solution repose sur l'utilisation d'une connexion TLS pour établir la communication entre les divers opérateurs LoRaWAN. Cependant, cette approche génère un trafic important. Nous envisageons la possibilité, à l'avenir, d'utiliser OSCORE ou EDHOC pour sécuriser nos échanges une fois que HTTPS aura été remplacé par CoAP.

Appendix A

Annexes

It is always something. Good, Fast, Cheap: Pick any two (you can't have all three).

— **RFC 1925, Nr. 7**
(The Twelve Networking Truths)

Sommaire

A.1 Détails sur les formats de sérialisation

A.1.1 JSON

JSON est largement utilisé dans le cloud computing pour la configuration des ressources, les échanges de données, l'orchestration des services, le stockage de données et la gestion des configurations. Son format lisible et facilement interprétable par les machines en fait un choix populaire pour la représentation et la manipulation des données dans les environnements cloud.

```
{
    "gateway_conf": {
        "gateway_ID": "32e35818f2fb812b",
        "server_address": "outils.plido .net",
        "serv_port_up": 1700,
        "serv_port_down": 1700,
        "keepalive_interval": 10,
        "stat_interval": 30,
        "push_timeout_ms": 100,
        "forward_crc_valid": true,
        "forward_crc_error": false,
        "forward_crc_disabled": false,
        "servers": [ {
            "server_address": "outils.plido.net",
            "serv_port_up": 1700,
            "serv_port_down": 1700,
            "serv_enabled": true
        } ]
    }
}
```

Listing A.1: Exemple d'un document JSON portant sur la configuration d'une passerelle LoRaWAN(Pygate)

En 2019, [149] a constaté que les documents JSON représentent une majorité croissante des réponses HTTP fournies par Akamai Technologies, un réseau de diffusion de contenu (CDN) de premier plan qui traite environ 3 000 milliards de requêtes HTTP par jour. Gartner, une société américaine d'étude et de conseil, prévoit que le marché du cloud public augmentera de 20,7 % en 2023 par rapport à 2022 pour atteindre 591,8 milliards de dollars et que le SaaS restera le segment de marché le plus important. Les systèmes de l'entreprise (SAAS) offrent souvent des interfaces pour programmer des applications. Il est courant que les API utilisent JSON pour demander et répondre aux questions. Selon leur étude, JSON est plus utilisé que XML [110]. La popularité de JSON par rapport à XML peut être attribuée au fait que, par rapport à XML, JSON se traduit par des chaînes de bits plus petites et par des implémentations d'exécution et de désérialisation efficaces en termes de mémoire [110].

La communauté scientifique porte également un intérêt à JSON. Dans [30], les chercheurs établissent le premier cadre formel pour les documents JSON et présentent ains JSON

Navigational Logic (JNL), un langage d'interrogation spécifique à JSON. De plus en plus de publications utilisent JSON dans le contexte des API, telles que les références [157], [38] et [40], ainsi que les technologies liées à l'écosystème JSON, comme la spécification JSON mentionnée dans les références[29],[95],[53] et [45]. En dehors du cloud computing, JSON joue aussi un rôle important dans divers domaines tels que les bases de données[71], le big data [160], l'analytique [113], [85], les applications mobiles [48], la modélisation 3D [78], l'Internet des objets[109], la recherche biomédicale [64] et les fichiers de configuration, par exemple, l'article [20] propose une vue d'ensemble de l'écosystème JSON, incluant une enquête sur les langages de schémas populaires, leurs implémentations, les technologies d'extraction de schémas ainsi que de nouveaux outils d'analyse syntaxique.

A.1.2 Apache Avro

Apache Avro [144] est un format de sérialisation de données développé par la fondation Apache Software. Il est conçu pour fournir une méthode compacte, rapide et efficace de sérialisation et de désérialisation des données, ce qui en fait un outil adapté pour les systèmes de traitement de données volumineuses. Cette spécification propose une approche souple et basée sur un schéma pour la sérialisation des données. Un schéma définit la structure des données, y compris les types de données et les noms des champs. Les schémas Avro sont définis en utilisant le format JSON, ce qui les rend faciles à lire, écrire et comprendre. Cette approche basée sur le schéma permet une évolution des données dans le temps, car de nouveaux champs peuvent être ajoutés ou des champs existants modifiés sans rompre la compatibilité descendante.
Une des principales caractéristiques d'Avro est son format binaire compact. Les données Avro peuvent être sérialisées dans un format binaire qui est très efficace en termes d'espace de stockage et de bande passante réseau. Le format binaire permet également une sérialisation et une désérialisation rapides, ce qui rend Avro adapté aux systèmes de traitement de données à haute performance.
Comme d'autres formats de données présentés, Avro prend en charge une large gamme de types de données primitifs tels que les entiers, les nombres à virgule flottante, les booléens, les chaînes de caractères et les octets, ainsi que des types de données complexes tels que les tableaux, les cartes, les enregistrements et les énumérations. Il prend également en charge les champs optionnels, permettant des valeurs de données facultatives. Les schémas Avro peuvent être imbriqués, ce qui permet de définir des structures de données complexes.

```
1
2 {
3     "type": "record",
4     "name": "TestObject",
```

```
5     "namespace": "schema.avro",
6     "fields": [{
7         "name": "gateway_conf",
8         "type": ["null", {
9             "type": "record",
10            "name": "Gateway_conf",
11            "fields": [{
12                "name": "forward_crc_disabled",
13                "type": ["null", "boolean"],
14                "default": null
15            }, {
16                "name": "forward_crc_error",
17                "type": ["null", "boolean"],
18                "default": null
19            }, {
20                "name": "forward_crc_valid",
21                "type": ["null", "boolean"],
22                "default": null
23            }, {
24                "name": "gateway_ID",
25                "type": ["null", "string"],
26                "default": null
27            }, {
28                "name": "keepalive_interval",
29                "type": ["null", "int"],
30                "default": null
31            }, {
32                "name": "push_timeout_ms",
33                "type": ["null", "int"],
34                "default": null
35            }, {
36                "name": "serv_port_down",
37                "type": ["null", "int"],
38                "default": null
39            }, {
40                "name": "serv_port_up",
41                "type": ["null", "int"],
42                "default": null
43            }, {
44                "name": "server_address",
45                "type": ["null", "string"],
46                "default": null
47            }, {
48                "name": "servers",
49                "type": ["null", {
50                    "type": "array",
51                    "items": ["null", {
52                        "type": "record",
53                        "name": "Server",
54                        "fields": [{
55                            "name": "serv_enabled",
56                            "type": ["null", "boolean"],
57                            "default": null
58                        }, {
59                            "name": "serv_port_down",
60                            "type": ["null", "int"],
61                            "default": null
62                        }, {
63                            "name": "serv_port_up",
64                            "type": ["null", "int"],
65                            "default": null
66                        }, {
67                            "name": "server_address",
68                            "type": ["null", "string"],
69                            "default": null
```

```
                        }]
                    }]
                }],
                "default": null
            }, {
                "name": "stat_interval",
                "type": ["null", "int"],
                "default": null
            }]
        }],
        "default": null
    }]
}
```

Listing A.2: Utilisation du format Avro Apache pour sérialiser les données d'entrée JSON de la listing A.1

En plus d'être un format binaire compact, Avro prend également en charge un format JSON pour l'échange de données et un format en mémoire compact appelé "in-memory data". Cette flexibilité permet à Avro d'être utilisé dans divers scénarios, tels que le stockage de données, les systèmes de messagerie et les appels de procédure à distance. Apache Avro s'intègre bien avec d'autres frameworks et outils de traitement de données volumineuses. Il est nativement pris en charge dans Apache Hadoop, Apache Spark, Apache Kafka et d'autres systèmes populaires de traitement de données, ce qui en fait un bon choix pour la sérialisation des données dans l'écosystème du Big Data.

A.1.3 ASN.1

ASN.1 (Abstract Syntax Notation One) [153] est une notation standard et flexible utilisée pour décrire des structures de données. Elle est définie par l'Union internationale des télécommunications (UIT) dans le cadre de la série de normes X.680. Ce format est principalement utilisé dans les télécommunications et les réseaux informatiques pour définir la syntaxe des messages échangés entre différents systèmes. Il permet de spécifier précisément les structures de données, les types de données et les règles d'encodage. ASN.1 prend en charge une large gamme de types de données, y compris les types primitifs (tels que les entiers, les booléens et les chaînes de caractères) et les types structurés (tels que les séquences, les ensembles et les types de choix).
La structure d'une spécification ASN.1 repose sur des modules, qui définissent les types de données et les règles d'encodage utilisés dans un contexte particulier. Un module ASN.1 est composé de définitions de types, de définitions de valeurs et d'instructions d'assignation. Les types définis dans les modules ASN.1 peuvent être utilisés pour encoder et décoder des données dans différents formats d'encodage, tels que BER (Basic Encoding Rules), DER (Distinguished Encoding Rules) et PER (Packed Encoding Rules).

ASN.1 fournit une manière normalisée et interopérable de décrire des structures de données et des protocoles. Il permet à différents systèmes implémentés dans différents langages de programmation d'échanger des données en utilisant une spécification commune. Cela le rend particulièrement utile dans les scénarios où les données doivent être échangées entre des systèmes hétérogènes ou à travers différents protocoles réseau.
Plusieurs langages de programmation proposent des bibliothèques et des outils pour travailler avec ASN.1, permettant aux développeurs de parser, d'encoder et de décoder des données basées sur des spécifications ASN.1. Ces bibliothèques génèrent souvent du code représentant les types ASN.1, ce qui facilite le travail avec les structures de données spécifiées dans le langage de programmation choisi.

```
1
2  GeneratedSchema DEFINITIONS AUTOMATIC TAGS ::= BEGIN
3  GeneratedType ::= SEQUENCE {
4      gateway_conf SEQUENCE {
5          gateway_ID UTF8String,
6          server_address UTF8String,
7          serv_port_up INTEGER,
8          serv_port_down INTEGER,
9          keepalive_interval INTEGER,
10         stat_interval INTEGER,
11         push_timeout_ms INTEGER,
12         forward_crc_valid BOOLEAN,
13         forward_crc_error BOOLEAN,
14         forward_crc_disabled BOOLEAN,
15         servers SEQUENCE OF SEQUENCE {
16             server_address UTF8String,
17             serv_port_up INTEGER,
18             serv_port_down INTEGER,
19             serv_enabled BOOLEAN
20         }
21     }
22 }
23 END
```

Listing A.3: Utilisation du format ASN.1 pour sérialiser les données d'entrée JSON de la listing A.1

A.1.4 Protocol Buffers

Protocol Buffers [5] , également connu sous le nom de Protobuf est un format de sérialisation développé par Google. Il est conçu pour être un moyen efficace et flexible de représenter des données structurées de manière binaire. Les données sérialisées en utilisant Protocol Buffers sont compactes, rapides à sérialiser et désérialiser, et peuvent être utilisées dans un large éventail d'applications.
Le principal avantage de Protocol Buffers réside dans sa structure de schéma. Les données sont définies dans un fichier de description de protocole (.proto), où les structures de données sont déclarées en utilisant un langage simple et lisible par l'homme. Le fichier

de description de protocole agit comme un contrat entre les systèmes qui partagent les données, spécifiant les types de données, les champs et les règles de sérialisation.
Une fois le fichier de description de protocole défini, un compilateur Protobuf est utilisé pour générer du code source dans différents langages de programmation. Ce code généré fournit des classes et des méthodes pour sérialiser, désérialiser et manipuler les données selon la structure définie dans le fichier de description de protocole.
L'un des avantages de Protocol Buffers est sa compacité. Les données sérialisées sont binaires et ont une taille réduite par rapport à d'autres formats, tels que le JSON ou le XML. Cela permet un transfert plus rapide des données sur le réseau et une utilisation plus efficace de l'espace de stockage.

```
1
2  syntax = "proto3";
3
4  message SomeMessage {
5
6      message Servers {
7          string server_address = 1;
8          uint32 serv_port_up = 2;
9          uint32 serv_port_down = 3;
10         bool serv_enabled = 4;
11     }
12
13     message Gateway_conf {
14         string gateway_ID = 1;
15         string server_address = 2;
16         uint32 serv_port_up = 3;
17         uint32 serv_port_down = 4;
18         uint32 keepalive_interval = 5;
19         uint32 stat_interval = 6;
20         uint32 push_timeout_ms = 7;
21         bool forward_crc_valid = 8;
22         bool forward_crc_error = 9;
23         bool forward_crc_disabled = 10;
24         repeated Servers servers = 11;
25     }
26
27     Gateway_conf gateway_conf = 1;
28 }
```

Listing A.4: Utilisation du format Protocol Buffers pour sérialiser les données d'entrée JSON de la listing A.1

En plus de sa compacité, Protocol Buffers offre également une évolutivité. Les schémas peuvent être mis à jour en ajoutant, en supprimant ou en modifiant des champs, tout en garantissant la compatibilité ascendante et descendante. Cela signifie que les anciennes versions des données peuvent toujours être lues par les nouvelles versions du logiciel et vice versa. Protocol Buffers est compatible avec plusieurs langages de programmation couramment utilisés, tels que C++, Java, Python et C#. Cela permet aux développeurs de travailler avec les données sérialisées dans leur langage de prédilection, simplifiant ainsi l'intégration dans les systèmes existants.

A.1.5 BSON

BSON (Binary JSON) [96] est un format binaire qui vise à offrir une alternative plus efficace à JSON pour le stockage et la transmission de données. Cependant, il est important de noter qu'il n'existe pas de spécification largement reconnue ou normalisée pour BJSON, et différentes implémentations peuvent avoir des interprétations légèrement différentes et des variations du format.Ce format prend en charge divers types de données similaires à JSON, tels que les chaînes de caractères, les nombres, les booléens, les tableaux et les objets. Les types de données sont encodés en binaire selon des conventions spécifiques. Les valeurs sont converties en leur équivalent binaire, ce qui permet de réduire la taille des données par rapport à un format basé sur du texte comme JSON. BSON utilise un encodage de longueur variable pour représenter les entiers et les chaines de caractères. Cela permet d'optimiser l'espace en utilisant moins d'octets pour représenter des valeurs plus petites. Les objets et les tableaux sont représentés sous forme de séquences d'éléments binaires. Les objets sont généralement encodés en tant que paires clé-valeur, tandis que les tableaux sont encodés en tant que séquences d'éléments.

```
bson.M{
  "gateway_conf": bson.M{
    "gateway_ID": "32e35818f2fb812b",
    "server_address": "outils.plido .net",
    "serv_port_up": 1700,
    "serv_port_down": 1700,
    "keepalive_interval": 10,
    "stat_interval": 30,
    "push_timeout_ms": 100,
    "forward_crc_valid": true,
    "forward_crc_error": false,
    "forward_crc_disabled": false,
    "servers": bson.A{
      bson.M{
        "server_address": "outils.plido.net",
        "serv_port_up": 1700,
        "serv_port_down": 1700,
        "serv_enabled": true,
      },
    },
  },
}
```

Listing A.5: Utilisation du format BSON pour sérialiser les données d'entrée JSON de la listing A.1

Etant donné n'est pas encore normalisé, il est important de noter que les détails précis du format BJSON peuvent varier entre les différentes implémentations et bibliothèques. Si vous utilisez une implémentation spécifique de BSON, il est recommandé de consulter la documentation fournie avec cette implémentation pour obtenir des informations détaillées sur le format et les fonctionnalités spécifiques.

A.1.6 UBJSON

L'objectif principal d'UBJSON [3] était de réduire la taille des données sérialisées tout en maintenant la simplicité et la lisibilité du JSON. Le format UBJSON utilise une représentation binaire pour différents types de données tels que les entiers, les flottants, les chaînes de caractères, les tableaux, les objets et les valeurs null ou booléennes. Le format UBJSON utilise des marqueurs spéciaux pour représenter chaque type de données, ce qui permet de réduire la taille globale des messages. Par exemple, un nombre entier peut être représenté en utilisant un marqueur spécifique plutôt qu'en convertissant le nombre en texte comme dans le JSON.
Ce format prend également en charge la compression des données sérialisées à l'aide d'algorithmes de compression tels que zlib ou Snappy. Cela permet de réduire davantage la taille des données lorsqu'une compression supplémentaire est nécessaire.

A.1.7 Microsoft Bond

Le format de sérialisation binaire Microsoft Bond [98], également connu sous le nom de simplement "Bond", est un format de sérialisation développé par Microsoft. Il a été créé dans le but de fournir une sérialisation binaire efficace et interopérable pour les communications entre différents systèmes.
Bond utilise un schéma de données déclaratif pour décrire la structure des données à sérialiser. Ce schéma est écrit dans un langage spécifique à Bond, qui ressemble à une syntaxe de type structurée. Le schéma spécifie les types de données, les champs et leurs attributs, ainsi que les relations entre les structures de données.Une fois le schéma défini, Bond génère du code dans différents langages de programmation, tels que C++, C#, Java et Python, pour représenter les données conformément au schéma. Ce code généré fournit des classes ou des structures qui facilitent la sérialisation et la désérialisation des données.
Bond utilise un format de sérialisation binaire compact et optimisé pour réduire la taille des données sérialisées. Il utilise également des mécanismes de compression pour réduire davantage la taille des données lorsque cela est nécessaire.En plus de la sérialisation binaire, Bond fournit également des fonctionnalités avancées telles que la sérialisation incrémentielle, la sérialisation en streaming et la gestion de la compatibilité avec les versions précédentes des schémas de données.
L'interopérabilité est une caractéristique clé de Bond. Les données sérialisées avec Bond peuvent être échangées entre des systèmes développés dans différents langages de programmation, permettant ainsi une communication transparente entre les plateformes.
Bond est utilisé dans de nombreux produits et services de Microsoft, tels que Azure Service

Fabric, Bing Ads, Xbox, Office 365, et bien d'autres.

A.1.8 Cap'n Proto

Le format binaire Cap'n Proto [2] est un format de sérialisation binaire efficace et performant développé par Kenton Varda chez Sandstorm.io. Il est conçu pour permettre une manipulation directe et rapide des données sans nécessiter de désérialisation coûteuse. Le format binaire Cap'n Proto est un format de sérialisation binaire efficace et performant développé par Kenton Varda chez Sandstorm.io. Il est conçu pour permettre une manipulation directe et rapide des données sans nécessiter de désérialisation coûteuse. Contrairement à d'autres formats de sérialisation, Cap'n Proto permet une sérialisation directe des données sans conversion en une représentation intermédiaire. Les données sont écrites directement en mémoire, ce qui évite les coûts de copie et de désérialisation. Cela rend le processus de sérialisation très rapide et efficace. Cap'n Proto utilise également une fonctionnalité appelée "sharing" (partage) de mémoire, ce qui signifie que plusieurs vues peuvent accéder aux mêmes données en mémoire sans nécessiter de copie. Cela permet d'économiser de l'espace mémoire et améliore les performances lors de la manipulation de grandes quantités de données.
Ce format prend également en charge la sérialisation incrémentielle, ce qui signifie que les modifications apportées aux données peuvent être sérialisées de manière incrémentale sans avoir à re-sérialiser l'ensemble des données. Cela permet des mises à jour plus rapides et plus efficaces des données. Il est conçu pour être interopérable entre différents langages de programmation. Le schéma déclaratif est utilisé pour générer du code dans le langage cible, permettant aux applications écrites dans différents langages de partager et de manipuler les mêmes données.

A.1.9 FlatBuffers

Le format FlatBuffers [111] est un format de sérialisation binaire conçu pour faciliter la manipulation efficace des données structurées. Il a été développé par Google en 2014 dans le but de répondre aux besoins de performance des applications utilisant de grandes quantités de données en temps réel. FlatBuffers a été conçu pour répondre à ces exigences spécifiques. Contrairement à d'autres formats de sérialisation, tels que JSON ou XML, qui nécessitent une étape de désérialisation pour accéder aux données, FlatBuffers permet un accès direct aux données sérialisées. Cela signifie qu'aucune copie ou désérialisation coûteuse n'est nécessaire pour accéder aux champs spécifiques des données.
Le principe central de FlatBuffers est la notion de "buffers plats" (flat buffers) c'est-a-dire,

les données sont organisées en mémoire de manière à pouvoir être lues directement, sans nécessiter de désérialisation préalable. Cela permet d'économiser du temps et de l'espace en évitant les opérations de désérialisation coûteuses. Une autre caractéristique clé de FlatBuffers est sa capacité à prendre en charge la gestion de la rétrocompatibilité des données. Les schémas de données peuvent évoluer avec le temps, et FlatBuffers offre des mécanismes intégrés pour gérer les versions antérieures des schémas de données et garantir la compatibilité avec les versions précédentes.
FlatBuffers est devenu populaire dans l'industrie du jeu vidéo, où les performances et l'efficacité sont essentielles. Il est également utilisé dans d'autres domaines, tels que les applications mobiles, les systèmes distribués, les bases de données, etc. Il est open source et est disponible dans plusieurs langages de programmation, notamment C++, Java, C#, Python et bien d'autres. Cela a facilité son adoption par la communauté des développeurs.

A.1.10 FlexBuffers

développé par l'équipe de FlatBuffers [112] chez Google en tant qu'extension de la bibliothèque de sérialisation FlatBuffers. FlexBuffers a été introduit en 2019 comme un format alternatif pour pallier certaines limitations de FlatBuffers, notamment dans les scénarios où des structures de données dynamiques et une évolution du schéma sont nécessaires.
La motivation derrière FlexBuffers était de proposer un format binaire plus flexible et extensible tout en conservant les avantages de performance de FlatBuffers. L'objectif de conception était de créer un format qui permette un stockage et une récupération efficaces des données structurées sans nécessiter d'analyse ou de déballage. Le développement de ce format a été influencé par divers facteurs, notamment le besoin de transfert et de stockage de données plus efficaces dans des environnements limités en ressources tels que les systèmes embarqués et les appareils mobiles. De plus, l'émergence de langages à typage dynamique et la demande croissante de flexibilité dans les structures de données ont nécessité un format de sérialisation capable de gérer des modifications de schéma dynamiques.
FlexBuffers utilise une approche hybride qui combine les meilleures caractéristiques des formats JSON et binaires. Il offre une représentation compacte des données en utilisant un encodage à longueur variable pour les valeurs numériques, ainsi qu'une représentation efficace des chaînes de caractères et des tableaux d'octets. Comme d'autres formats présentés, il prend en charge une large gamme de types de données, notamment les entiers, les nombres à virgule flottante, les booléens, les chaînes de caractères, les cartes et les vecteurs.
L'une des fonctionnalités remarquables de FlexBuffers est sa prise en charge de l'évolution du schéma qui lui permet d'ajouter, de supprimer ou de modifier des champs dans un

schéma sans rompre la compatibilité avec les données existantes. Cette flexibilité le rend adapté aux scénarios où les structures de données peuvent évoluer avec le temps ou lorsque différentes versions des mêmes données sont échangées entre les systèmes. Il est aujourd'hui dans le domaine des systèmes embarqués comme son prédécesseur.

A.1.11 Apache Thrift

Le format Apache Thrift [91] est un protocole de sérialisation binaire développé par Facebook en 2007. Il a été conçu pour faciliter la communication entre des systèmes distribués hétérogènes, en permettant l'échange de données structurées de manière efficace et interopérable.
Son développement remonte aux besoins de Facebook à l'époque, qui devait gérer des milliards de requêtes par jour provenant de différents types de clients et de serveurs. Ils avaient besoin d'un moyen de communication flexible et performant entre ces systèmes distribué, qui incluaient des langages de programmation différents, des plateformes variées et des besoins de performances élevées. En réponse à ces défis, Facebook a développé le projet Thrift (anciennement connu sous le nom de "Apache Software Foundation's Thrift") en tant que framework pour la communication entre systèmes distribués. Le format binaire Thrift est l'un des composants clés de ce framework, utilisé pour sérialiser les données structurées lors des échanges entre les clients et les serveurs.

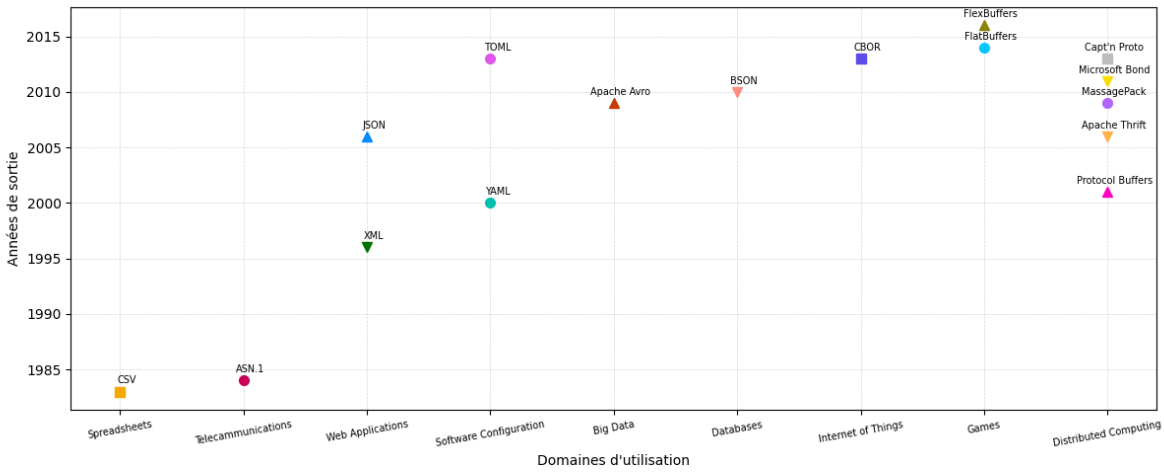


Figure A.1: Chronologie des formats de données les plus populaires depuis 1983

Table A.1: Les travaux portant sur les formats de sérialisation binaire plus compact que JSON [108, 101, 151, 154, 142, 49, 148, 75]

Domaine	Année	Travaux	Format binaire sans schéma						
			JSON	BSON	CBOR	FlBuff	MPack	Smile	UBJSON
Réseaux	1992	*Neufeld et al.*	-	-	-	-	-	-	-
	1994	*Mitra et al.*	-	-	-	-	-	-	-
	2022	*Juan Cruz et al.*	✓	✓	✓	✓	✓	✓	✓
Microblogging	2011	*Wibowo et al.*	✓	-	-	-	-	-	-
Mobile	2011	*Gil et al.*	✓	-	-	-	-	-	-
	2012	*Sumaray et al.*	✓	-	-	-	-	-	-
Services Web	2012	*Vanura et al.*	✓	-	-	-	✓	-	-
	2015	*Kyurkchiev. et al.*	✓	-	-	-	✓	-	-
Jeux vidéos	2013	*Espinoza-Arias et al.*	-	-	-	-	-	-	-
Internet des objets	2016	*Thomas Rix et al.*	-	-	✓	-	-	-	-
	2017	*Petersen et al.*	-	✓	✓	-	✓	✓	-
	2018	*Sahlmann et al.*	✓	-	✓	-	-	-	-
	2018	*Hamerski et al.*	-	-	-	-	✓	-	-
	2019	*Biswal et al.*	-	✓	-	✓	✓	-	-
	2019	*Pradana et al.*	✓	-	-	✓	-	-	-
	2019	*Jahed et al.*	✓	-	-	✓	-	-	-
	2019	*Kristina et al.*	-	-	✓	-	-	-	-
	2019	*Iglesias-Urkia et al.*	-	-	✓	-	-	-	-
	2020	*Sanchez et al.*	-	-	✓	-	-	-	-
	2020	*Brendan Moran et al.*	-	-	✓	-	-	-	-
	2022	*Molina Araque et al.*	-	-	✓	-	-	-	-
	2023	***Lemogue et al.***	-	-	✓	-	-	-	-
Automobile	2020	*Proos et al.*	-	-	-	-	-	-	-
Intelligence Artificielle	2020	*Parimi et al.*	-	-	-	✓	-	-	-

Table A.2: La suite du tableau 4.2 [66, 7, 41, 118, 127, 129, 42, 10, 62, 63, 13]

Domaine	Année	Travaux	Format binaire avec schéma						
			ASN.1	Apache Avro	MBond	Capt'n Proto	Flatbuff	ProtoBuff	Apache Trift
Réseaux	1992	*Neufeld et al.*	✓	-	-	-	-	-	-
	1994	*Mitra et al.*	✓	-	-	-	-	-	-
	2022	*Juan Cruz et al.*	✓	✓	✓	✓	✓	✓	✓
Microblogging	2011	*Wibowo et al.*	-	-	-	-	-	✓	-
Mobile	2011	*Gil et al.*	-	-	-	-	-	✓	-
	2012	*Sumaray et al.*	-	✓	-	-	-	✓	✓
Services Web	2012	*Vanura et al.*	-	✓	-	-	-	✓	-
	2015	*Kyurkchiev. et al.*	-	-	-	-	-	-	-
Jeux vidéos	2013	*Espinoza-Arias et al.*	-	-	-	-	-	✓	-
Internet des objets	2016	*Thomas Rix et al.*	-	-	-	-	-	-	-
	2017	*Petersen et al.*	-	✓	-	-	-	✓	-
	2018	*Sahlmann et al.*	-	-	-	-	-	-	-
	2018	*Hamerski et al.*	-	-	-	-	✓	✓	-
	2019	*Biswal et al.*	-	-	-	-	✓	✓	-
	2019	*Pradana et al.*	-	-	-	-	✓	-	-
	2019	*Jahed et al.*	-	✓	-	-	✓	✓	✓
	2019	*Kristina et al.*	-	-	-	-	-	-	-
	2019	*Iglesias-Urkia et al.*	-	-	-	-	-	-	-
	2020	*Sanchez et al.*	-	-	-	-	-	-	-
	2020	*Brendan Moran et al.*	-	-	-	-	-	-	-
	2022	*Molina Araque et al.*	-	-	-	-	-	-	-
	2023	***Lemogue et al.***	-	-	-	-	-	-	-
Automobile	2020	*Proos et al.*	-	✓	-	✓	✓	✓	-
Intelligence Artificielle	2020	*Parimi et al.*	-	-	-	-	✓	✓	-

A.1.12 Travaux antérieurs les plus pertinents

Il existe de nombreux travaux portant sur l'évaluation et de l'analyse comparative des différents formats de sérialisation de données. Cependant, nous allons nous focaliser sur ceux qui traitent les formats binaires compatibles avec JSON. Comme dans la section xx, ces travaux sont classés en fonction de leur domaine d'application.
Les tableaux A.1 et A.2 résument les articles sélectionnés, classés en fonction de la catégorie des différents formats.

A.1.12.1 Réseaux et systèmes distribués

Dans des systèmes distribués, pour avoir une bonne communication, il est important de spécifier lors des échanges le type et le format des données qu'on manipule. Cette spécification est surtout nécessaire pour des protocoles qui manipulent des informations très complexes, car chaque instance ou valeur d'un type doit être convertie dans ce format avant transmission. C'est dans ce sens que Neufeld et al [108] proposent dans leurs travaux un ensemble d'outils de programmation pour le traitement automatique basé sur le format de sérialisation binaire ASN.1. Ils analysent aussi la possibilité d'extension de ce format. Dans les années 1990, ANS.1 etait la première représentation abstraite normalisée des données structurées, ainsi presque toutes les normes de la couche application du modele OSI spécifiait leurs messages avec ANS.1, tout comme plusieurs normes de signalisation de réseau numérique à intégration de services (RNIS) basées sur des protocoles OSI. BER (Basic Encoding Rules) était utilisé lors de cette représentation, malheureusement cette encodage n'était pas efficace , surtout pour des applications en temps. Pour résoudre ce problème, Mitra et al [101] ont proposé des schémas de codages alternatifs tels DER (Distinguished Encoding Rules) CER (Canonical Encoding Rules), PER (Packed Encoding Rules et LWER (Lightweight Encoding Rules)

Juan Cruz et al. [151] font une évaluation comparative de l'ensemble des formats de serialisation binaire sans schéma (BSON, CBOR, FlexBuffers, MessagePack, Smileet UBJSON) et avec schemas ((ASN.1, Apache Avro, Microsoft Bond, Cap'n Proto, FlatBuffers, Protocol Buffers et Apache Thrift). Dans leurs travaux, ils présentent l'histoire, les caractéristiques et le processus de sérialisation de chaque format. Grâce à ces travaux, on peut facilement prendre des décisions éclairées sur le choix entre des formats de sérialisation binaire compatibles JSON avec ou sans schéma.

A.1.12.2 Microblogging

Avec l'arrivée des médias sociaux à l'instar de twitter et Instagram, le microblogging est devenu très populaire, car il facilite interaction directe et rapide avec le public. Les données échangées dans ce domaine sont généralement encodées en JSON. Les Wibowo et al. [154] évaluent le Protocol Buffers comme alternative à JSON. Pour cela, il est utilisé pour sérialiser les données échangées lors de la communication de microblogging en prenant Twitter comme exemple d'application. Leurs résultats montrent qu'en termes de performance, Protocol Buffers est meilleur que JSON et permet de gagner plus d'espace lors de la sérialisation, par conséquent les échanges de données sont plus rapides par rapport à JSON.

A.1.12.3 Mobile

Aujourd'hui, en raison de l'augmentation du nombre d'appareils mobiles connectés à Internet facilement disponibles et de leurs caractéristiques uniques, le choix du format de sérialisation des données approprié est devenu de plus en plus difficile. Ces appareils disposent des ressources rares et une bande passante très limitée.
Sumaray et al. [142] comparent, dans leurs travaux, quatre formats de sérialisation de données différents en mettant l'accent sur la vitesse de sérialisation, la taille des données et la convivialité. Les formats de sérialisation sélectionnés incluent XML, JSON, Thrift et ProtoBuf. XML et JSON sont les formats de données textuels les plus connus, tandis que ProtoBuf et Thrift sont des formats de sérialisation binaires relativement nouveaux.

Dans [49], les auteurs Analysent l'impact des formats XML, JSON et Protocol Buffers en termes de consommation énergétique et de coût de traitement des données. Leurs résultats contribuent au développement d'applications mobiles qui nécessitent de transférer de manière indépendante ou non des données pour un serveur Web, comme des applications de sauvegarde de données, de surveillance d'informations ou toute application mobile qui synchronise des données avec le Web, ayant comme objectifs principaux les impacts de l'énergie et du traitement dans les équipements mobiles.

A.1.12.4 Services Web

Juan Vanura et al. [148] comparent les formats et les bibliothèques utilisés pour la sérialisation et la désérialisation des données, généralement avec les services Web RESTful, en termes de temps de traitement et de taille des données de sortie. Les formats testés incluent XML, JSON, MessagePack, Avro, Protocol Buffers et la sérialisation native de chacun des langages de programmation testés. La sérialisation et la désérialisation

sont testées en PHP, Java et JavaScript à l'aide de 49 bibliothèques officielles et tierces différentes. Leurs résultats montrent que les formats Avro et Protobuf obtiennent le meilleur résultat lors du premier test. Ces formats binaires nécessitent une définition de schéma. Par conséquent, une partie des informations n'a pas besoin d'être sérialisée, car elle fait partie du schéma. Cela implique une taille plus petite des données de sortie. Les pires résultats sont généralement obtenus par les bibliothèques XML.

Afin de faciliter l'accès à distance à une ressource Web, le choix d'un moyen simple et rapide de représentation et de transfert des données est très important. C'est pourquoi dans [75], les auteurs comparent MessagePack et JSON en utilisant une grande quantité de données. Les critères de comparaison dans ces travaux sont basés sur la quantité de données transférées sur le web et la facilité d'utilisation du format. Leurs résultats montrent que les données sérialisées en JSON sont faciles à lire et à traiter, mais en retard par rapport à MessagePack.

A.1.12.5 Jeux vidéos

Le développement des jeux vidéos [66] [7]a largement augmenté ces dernières années. Les exigences de communication de données de ces jeux en ligne sont de plus en plus importantes. Comme une grande quantité de données doit interagir, la vitesse d'interaction entre les données détermine la vitesse de réponse et l'efficacité des jeux en ligne, et détermine ainsi l'expérience de jeu des joueurs. L'un des protocoles les plus utilisés dans ces jeux est XMPP, c'est un protocole qui met beaucoup l'accent sur l'instantanéité et la sécurité. Parce qu'il peut être utilisé dans différents domaines, il a besoin d'une coopération entre différents domaines. JSON a longtemps été le format le plus utilisé par ce protocole parce qu'il est tres proche de XML.

Jianhua Feng et al. [41] proposent de remplacer dans ces jeux vidéos, le format JSON par protobuf qui est un format de transmission binaire afin de réduire le volume de données transmises et améliorer encore l'efficacité et la vitesse de réponse dans le processus de transmission de données des jeux et d'offrir une meilleure expérience de jeu. Leurs résultats montrent que l'adoption de protobuf dans la transmission de données entre le client et le serveur peut réduire considérablement la quantité de données dans la transmission de données.

A.1.12.6 Internet des objets

Dans le contexte de l'Internet des objets, la transmission des données se fait sur une faible bande passante et avec des appareils contraints. Ainsi, pour avoir une communication efficace dans ce domaine, il faut avoir une bonne sérialisation. Perterson et al [118]. présentent une des meilleures alternatives que XML et JAXB et donnent des conseils pour choisir le format de sérialisation et la bibliothèque les plus appropriés en fonction du contexte. Dans [127], Thomas et al. proposent une approche pour convertir le format XML au format CBOR. En effet, XML est adopté aux systèmes qui produisent de grandes quantités d'entités de données dans un temps limité. Afin de garder les avantages du XML dans de tels systèmes, il est important de réduire la bande passante. Comme Tous les formats textuels, XML nécessite trop de bande passante en raison de sa représentation en texte brut et des compromis que cela implique. Par conséquent, un format binaire comme CBOR est nécessaire, car il permet une conversion sans perte depuis et vers des documents XML.

Sahlmann et al [129] proposent une approche permettant de donner une description des dispositifs contraints basée sur l'ontologie. Pour cela, ils évaluent CBOR, comme une norme relativement nouvelle pour la représentation des données de ces dispositifs contraints, et RDF HDT [42], qui prend en charge la compression des données RDF et peut donc être bien adapté à la représentation binaire des descriptions de dispositifs.

Dans les travaux de Biswal et al [10], il est question de chercher comment améliorer l'efficacité des ressources dans les systèmes IoT en raison de la nature très limitée de leurs ressources, de leurs dispositifs et des limites posées par la bande passante lors de la transmission des données. Pour cela, dans leurs études, ils ont évalué l'efficacité de plusieurs formats de sérialisation binaire tels que Protocol Buffers, FlatBuffers et MessagePacket, dans le traitement des informations au niveau de chaque capteur afin de trouver quel format est plus adapté au sein d'infrastructure IoT pour faciliter leur interopérabilité.

Pour Glesias-Urkia et al [62], les sous-stations électriques sont des éléments cruciaux des réseaux électriques intelligents (Smart Grids). Leur déploiement nécessite l'utilisation des protocoles de communication non interopérables. Bien que la communication inter-station soit basée sur la norme 61850 de la Commission électrotechnique internationale (CEI), cela représente toujours un très grand défi. Dans leurs travaux, pour faire face à ces défis, les auteurs démontrent que la CEI 61850 peut être entièrement mappée au protocole d'application contrainte (CoAP) en combinaison avec le format de représentation d'objet binaire concis (CBOR) tout en améliorant les performances du système par rapport aux alternatives existantes [par exemple, WS-SOAP et le protocole HTTP)]. Dans leurs tests, ils comparent les systèmes qui utilisent CoAP+CBOR et ceux basés sur les services Web HTTP et WS-SOAP. Leurs résultats montrent qu'en combinant CoAP et CBOR, on gagne

entre 18% à 44% sur la taille des messages et entre 71% à 85% sur le temps de transmission.

Selon Sanchez et al. [63], lorsqu'un dispositif IoT est déployé et mis sous tension, il doit effectuer un processus d'amorçage pour faire partie du réseau et, par conséquent, faire partie du domaine de sécurité. Le processus d'amorçage implique l'exécution de procédures de sécurité telles que l'authentification, l'autorisation et la gestion des certificats/clés. Ce processus est d'une importance considérable pour les opérateurs qui doivent gérer et déployer un nombre important d'appareils IoT. Un amorçage efficace doit être simple, léger, basé sur des protocoles standard et interopérable pour faciliter une solution autonome à toute technologie IoT. Pour faire face à ce défi, les auteurs évaluent, dans leurs travaux, la possibilité de fournir des niveaux de sécurité adéquats avec des technologies telles que EAP-NOOB (EAP-Nimble out-of-band) [18], CoAP-EAP (Constrained Application Protocol EAP) [46] et LO-CoAP-EAP (Low-Overhead CoAP-EAP) [47]. Ils optimisent le format JSON qui est utilisé dans ces différents protocoles en le remplaçant par CBOR. Leurs tests sont réalisés dans un réseau LoRaWAN et les résultats montrent que l'utilisation de CBOR permet de réduire la taille des messages envoyés par les dispositifs IoT par voie hertzienne sans perdre d'informations.

Molina Araque et al. [13] pensent qu'aujourd'hui, l'interopérabilité est un défi majeur pour accélérer le déploiement des dispositifs ou applications IoT. La plupart du temps, les informations manipulées par les dispositifs et/ou applications IoT sont présentées sous forme de séries chronologiques (TS) [12]. Une solution très peu prometteuse, car elle utilise généralement des formats non standardisés. C'est pourquoi, les auteurs introduisent dans leurs travaux, une nouvelle architecture avec un middleware dont le but est de regrouper de manière compacte les informations provenant d'une série chronologique produite par les dispositifs IoT. Ensuite, ils proposent deux autres approches pour améliorer le modèle proposé dans un projet actuel de l'IETF [27], ainsi qu'une représentation arborescente basée sur CBOR pour indiquer la position des variables et des deltas. Leurs travaux permettent ainsi de rompre la dépendance importante qui existe entre les dispositifs IoT et les appareils cloud.

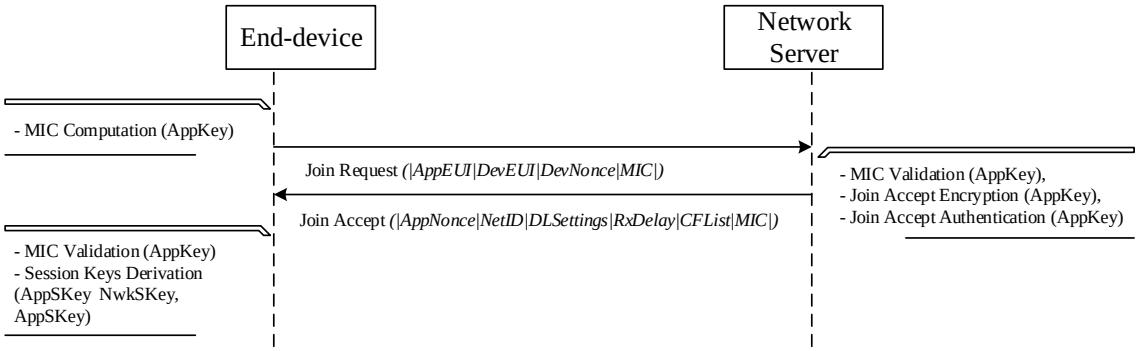


Figure A.2: Procédure de jonction dans LoRaWAN 1.0

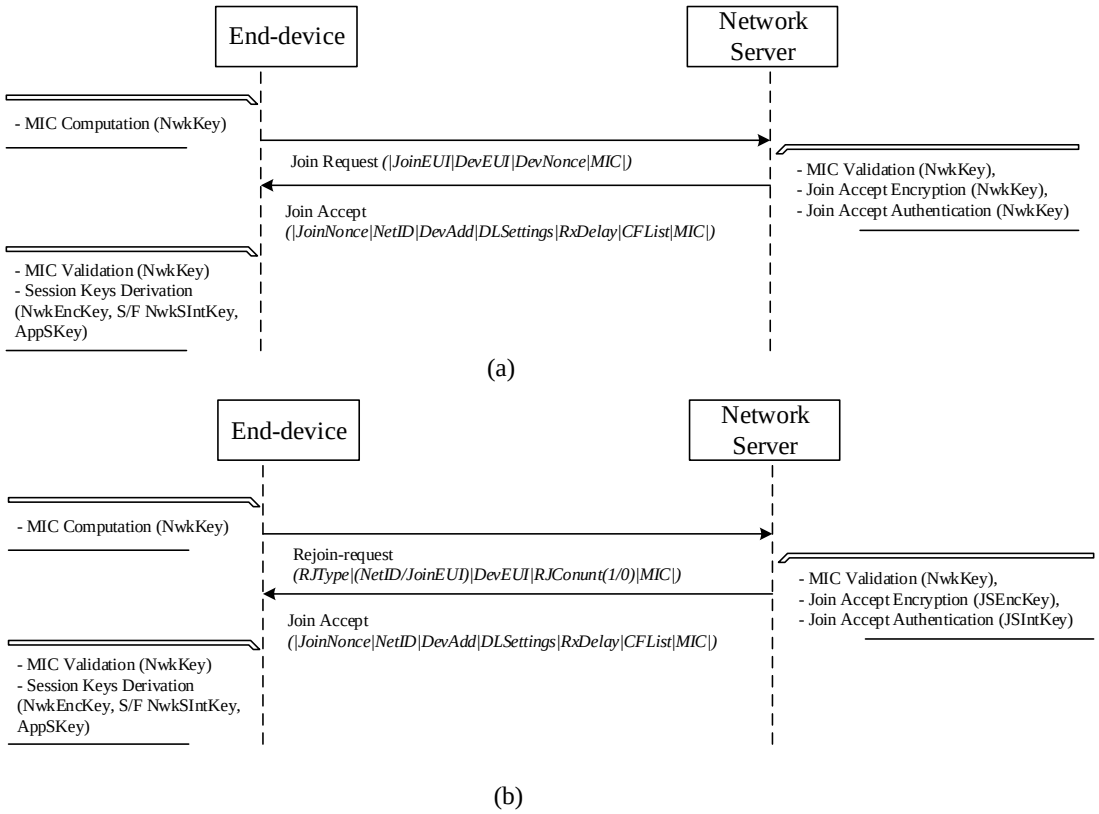


Figure A.3: Procédure de jonction dans LoRaWAN 1.1, (a) procédure de jonction à partir de zéro, (b) procédure de jonction déclenchée par un join request envoyé par un ED.

A.2 Infrastructure de gestion de clés

Dans cette section, nous décrivons le mode de fonctionnement d'une Public Key Infrastructure (PKI), la notion de confiance et certificats. Ces notions sont essentielles à la compréhension de notre approche.

A.2.1 Cryptographie asymétrique

La cryptologie, de manière étymologique, est la discipline consacrée à l'art du secret. Elle se focalise sur l'ensemble des procédés visant à garantir certaines caractéristiques liées à la confidentialité de l'information. Parmi ces caractéristiques, les plus reconnues sont les suivantes [84] :

- La préservation de la **confidentialité** des données, qui assure que seules les personnes autorisées peuvent accéder aux informations ;
- La vérification de l'**authenticité** des interlocuteurs, qui confirme que l'identité déclarée par un participant est exacte ;
- La garantie de l'**intégrité** des données, qui certifie que celles-ci n'ont pas été altérées pendant leur transmission par une tierce partie.

Il existe aujourd'hui plusieurs méthodes permettant de vérifier ces caractéristiques en se basant sur les deux types de cryptographie. Il s'agit de la cryptographie symétrique ou asymétrique.

Nous nous intéressons dans ce chapitre à la cryptographie asymétrique. Elle fonctionne de la manière suivante : Le paradigme du chiffrement asymétrique repose sur un principe simple. C'est-à-dire que chaque participant dispose d'une paire de clés (P_k,S_k), dont l'une est la clé publique (P_k) et l'autre est la clé privée (ou clé privée) (S_k). Comme son nom l'indique, le clé publique peut être librement partagée et est principalement utilisée pour chiffrer les messages. La clé privée (associée à la clé publique) est utilisée par son unique propriétaire pour retrouver le message d'origine. Selon les travaux de [87], un cryptosystème à clés publique est défini comme suit :

Definition A.2.1. Un système de cryptographie ou cryptosystème à clé publique est caractérisé par trois algorithmes qui s'exécutent en temps polynomial (Génération, Chiffrement, Déchiffrement) notés (G, E, D) tels que :

- Le premier algorithme G qui génère la clé, prend en entrée un paramètre de sécurité de taille n en bits, noté 1_n, et produit en sortie une paire de clés, une clé publique P_k et une clé privée S_k.
- L'algorithme de chiffrement E utilise la clé publique P_k et un message m appartenant

à l'espace des textes en clair comme paramètres. Il renvoie un texte chiffré c tel que $c = E_{Pk}(m)$.

- Enfin, l'algorithme de déchiffrement D prend en entrée la clé privée S_k et un texte chiffré c. Il génère un texte en clair m tel que $m = D_{Sk}(c)$.

Le but principal du système cryptographique est d'assurer la confidentialité des données pour que le propriétaire de la clé privée soit la seule personne à déchiffrer et accéder aux contenus du message. Il permet aussi d'assurer l'intégrité des données grâce aux signatures numériques.

Definition A.2.2 (D'après [87])**.** Un procédé de signature est un ensemble de trois algorithmes probabilistes à complexité temporelle polynomiale(Génération, Signature, Vérification), noté (G, S, V), qui doivent satisfaire les conditions suivantes :

- L'algorithme de génération de clé G prend en entrée un paramètre de sécurité de taille n bits, noté 1_n, et produit en sortie une paire de clés publique et privée (P_k, S_k).
- L'algorithme de signature S prend en entrée la clé privée S_k et un message m. Il renvoie la signature s telle que $s = S_{Sk}(m)$.
- Enfin, l'algorithme de vérification V prend en entrée la clé publique P_k, un message m et une signature s. Sa sortie est un b bit, où $b = V_{Pk}(m, s)$, qui vaut 1 si la signature est valide et 0 sinon.

Dans la pratique, il est fréquent que l'algorithme de signature corresponde à l'algorithme de déchiffrement, et que l'algorithme de vérification de la signature corresponde à l'algorithme de chiffrement. Cependant, il est essentiel de prendre des précautions lors de la sélection du message à signer, car il peut parfois être nécessaire de le transformer avant de le signer. Par exemple, dans le cas de l'algorithme RSA bien connu [76], la signature doit être appliquée à l'empreinte du message, sous peine d'exposer des vulnérabilités à des attaques.

La signature des messages est basée sur un modèle de confiance établi par une infrastructure PKI. On peut avoir un modèle de confiance hiérarchique ou maillé.

A.2.2 Modèle de confiance de l'infrastructure PKI

Definition A.2.3. La PKI ou infrastructure à clés public peut être définie comme « un ensemble d'entités qui communiquent à l'aide de protocoles et fournissent des services de gestion des clés publiques et de leurs certificats ». De manière générale, le rôle de la PKI

est de définir les mécanismes nécessaires pour établir et maintenir les relations de confiance requises et obtenir des services de sécurité. La plupart des services Internet qui offrent une connexion sécurisée à leurs serveurs, par le biais du protocole HTTPS par exemple, se fondent sur PKI ou PKIX (lorsqu'ils utilisent les certificats certificats X.509).

Definition A.2.4. Un certificat est un objet qui contient une clé publique et tous les paramètres de son propriétaire, et est émis et signé par (au moins) un tiers de confiance, communément appelé CA.

A.2.2.1 Modèle hiérarchique

Le modèle hiérarchique se caractérise par la présence de plusieurs autorités de certification, généralement désignées sous le nom de CA. Ces CA sont liées par une relation de supérieur à subordonné. La plus haute autorité dans cette hiérarchie est la CA racine (Root CA), qui délègue les responsabilités de certification à des entités subordonnées, et ainsi de suite, en fonction de la taille de l'organisation, jusqu'à atteindre l'utilisateur final. Lorsqu'une nouvelle autorité de certification est établie, elle doit faire certifier son propre certificat par l'autorité qui est directement au-dessus d'elle dans la hiérarchie. La CA racine est la seule autorité à posséder un certificat auto-signé. La confiance en la CA racine est universelle, et chaque membre de la hiérarchie a confiance en ses supérieurs immédiats ainsi qu'en ceux qui sont au-dessus d'eux dans la structure.
La figure A.4 montre le processus de validation des certificats dans un modèle de confiance hiérarchique. Dans ce modèle, le propriétaire de $A1$ fait confiance à l'autorité de certification $CA1$ et qui fait son tour confiance à CA racine.

A.2.2.2 Modèle maillé

Contrairement au modèle hiérarchique, dans le modèle maillé , les autorités de certification établissent des relations de confiance mutuelle. Comme illustré dans la figure A.5, chaque CA assume la responsabilité des certificats de ses pairs (les autres CAs) ainsi que des certificats de ses propres utilisateurs.

Peu importe le modèle de confiance utilisé, lorsque l'utilisateur $B1$ souhaite établir la confiance envers $A1$, il doit suivre la chaîne de confiance en remontant depuis $A1$ jusqu'à une autorité de confiance partagée. Par conséquent, pour vérifier le certificat de $B1$, $A1$ doit entreprendre les étapes suivantes :

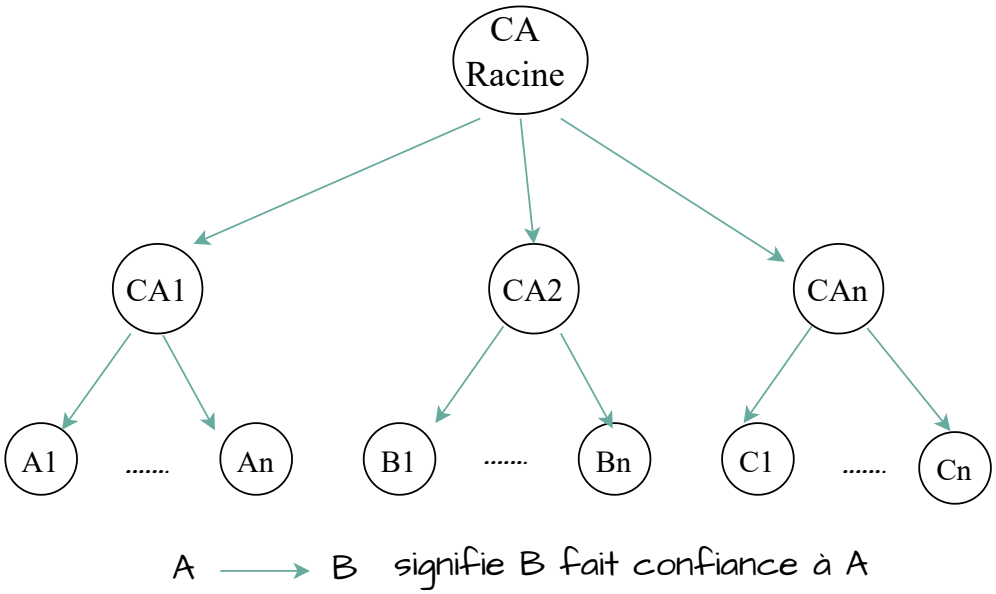


Figure A.4: Modèle hiérarchique de certification

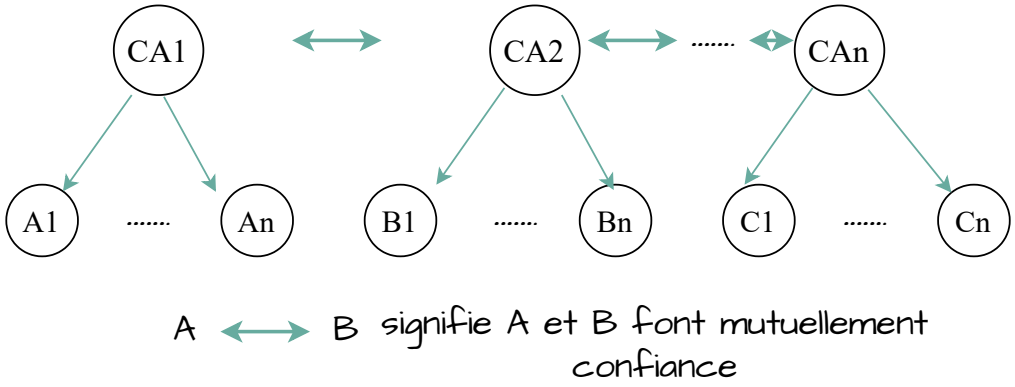


Figure A.5: Modèle maillé de certification

- Valider la signature de B1 en utilisant sa clé publique.
- Authentifier la signature de l'autorité CAn sur le certificat de $C1$ en utilisant la clé publique de CAn.
- Vérifier la signature de l'autorité racine (Root CA) sur le certificat de CAn en utilisant la clé publique de la racine CA.

A.3 Liste des publications et communications

A. Lemogue, I. Martinez, L. Toutain, and A. Bouabdallah, "Federated IoT roaming using private DNS resolutions," in NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, April 2022, Budapest, Hungary, pp. 1–6.(DOI:10.1109/NOMS54207.2022)

A. Lemogue, I. Martinez, L. Toutain, and A. Bouabdallah, "Towards Flexible and Compact encoded DNS Messages using CBOR Structures," in 2023 42nd IEEE International Performance Computing and Communications Conference(IPCCC), November 2023, Anaheim, California, USA DOI:10.1109/IPCCC59175.2023.10253849

A. Lemogue, I. Martinez, L. Toutain, and A. Bouabdallah, "Optimisation de l'architecture LoRaWAN pour la Mobilité des Appareils IoT" in AlgoTel 2023 - 25èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, May 2023, Cargese, France. (https://hal.science/hal-04092150)

A. Lemogue, I. Martinez, L. Toutain, and A. Bouabdallah, "Utilisation du DNS pour le roaming entre les serveurs réseau LoRa (LNS)" in Journées LPWAN 2021, 8-9 juillet, Clermont-Ferrand, France. (https://journees-lpwan-2021.limos.fr/)

Appendix B

Bibliographie

No matter how hard you push and no matter what the priority, you can't increase the speed of light.

— **RFC 1925, Nr. 2**
(The Twelve Networking Truths)

Sommaire

Références

[1] In: *ITU Telecommunication Standardization Sector. 2015. Abstract Syntax Notation One (ASN.1):Specification of basic notation. Standard. ITU Telecommunication Standardization Sector*, p. 166.

[2] Kenton Varda. 2013. "Sandstorm". In: *Cap'n Proto Schema Language*. URL: https://capnproto.org/language.html.

[3] Steven Braeger. 2016. In: *Universal Binary JSON Specification. UBJSON*. URL: https://ubjson.org.

[4] Tatu Saloranta. 2017. "FasterXML". In: *Efficient JSON-compatible binary format: "Smile"*. URL: https://github.com/FasterXML/smile-format-specification/blob/master/smile-specification.md.

[5] Google. 2020. "Google". In: *Protocol Buffers Version 3 Language Specification*. URL: https://developers.google.com/protocol-buffers/docs/reference/proto3-spec.

[6] *5GS Roaming Guidelines*. Tech. rep. Version 4.0. GSMA, 2021. URL: https://www.gsma.com/newsroom/wp-content/uploads//NG.113-v4.0.pdf.

[7] Patrick Abellard and Alexandre Abellard. "Serious games adapted to children with profound intellectual and multiple disabilities". In: *2017 9th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*. IEEE, Sept. 2017. DOI: 10.1109/vs-games.2017.8056597. URL: https://doi.org/10.1109/vs-games.2017.8056597.

[8] LoRa Alliance. *LoRaWAN 1.1 and Backend Interfaces 1.0 Specification*. Tech. rep. 2021.

[9] E. Allman et al. *RFC 4871: DomainKeys Identified Mail (DKIM) Signatures — rfc-editor.org*. https://www.rfc-editor.org/rfc/rfc4871. [Accessed 28-07-2023]. 2007.

[10] Marco Brambilla Amrit Kumar Biswal Obada Almallah. "Analytical Assessment of Binary Data Serialization Techniques in IoT Context". In: 19 (Dec. 2019), pp. 22–41.

URL: https://www.politesi.polimi.it/bitstream/10589/150617/1/Thesis_ObadaAlmallah.pdf.

[11] Lionel Anciaux. *Le Roaming LoRaWAN. État des déploiements, Besoins et alternatives ... - IOT Factory — iotfactory.eu*. https://iotfactory.eu/fr/le-roaming-lorawan-etat-des-deploiements-besoins-et-alternatives/. [Accessed 20-09-2023].

[12] *Apache IoTDB: time-series database for internet of things: Proceedings of the VLDB Endowment: Vol 13, No 12 — doi.org*. https://doi.org/10.14778/3415478.3415504. [Accessed 25-07-2023].

[13] Sebastian Molina Araque et al. "Toward a Standard Time Series Representation for IoT based on CBOR Templates". In: *2022 Global Information Infrastructure and Networking Symposium (GIIS)*. IEEE, Sept. 2022. DOI: 10.1109/giis56506.2022.9936910. URL: https://doi.org/10.1109/giis56506.2022.9936910.

[14] R. Arends et al. "RFC4033: DNS Security Introduction and Requirements". In: (March 2005). URL: https://www.rfc-editor.org/rfc/rfc4033.

[15] Roy Arends. *RFC 4033: DNS Security Introduction and Requirements — datatracker.ietf.org*. https://datatracker.ietf.org/doc/html/rfc4033. [Accessed 14-09-2023].

[16] Roy Arends. *RFC 4034: Resource Records for the DNS Security Extensions — datatracker.ietf.org*. https://datatracker.ietf.org/doc/html/rfc4034. [Accessed 14-09-2023].

[17] Roy Arends. *RFC 4035: Protocol Modifications for the DNS Security Extensions — datatracker.ietf.org*. https://datatracker.ietf.org/doc/html/rfc4035. [Accessed 14-09-2023].

[18] Tuomas Aura, Mohit Sethi, and Aleksi Peltonen. *Nimble Out-of-Band Authentication for EAP (EAP-NOOB)*. English. Other. United States, Dec. 2021. DOI: 10.17487/RFC9140.

[19] Wael Ayoub et al. "Media independent solution for mobility management in heterogeneous LPWAN technologies". In: *Computer Networks* 182 (Dec. 2020), p. 107423. DOI: 10.1016/j.comnet.2020.107423.

[20] Mohamed-Amine Baazizi et al. "Schemas and Types for JSON Data". In: *Proceedings of the 2019 International Conference on Management of Data*. ACM, June 2019. DOI: 10.1145/3299869.3314032. URL: https://doi.org/10.1145/3299869.3314032.

[21] Sandoche Balakrichenan et al. "IoTRoam: design and implementation of a federated IoT roaming infrastructure using LoRaWAN". 2021.

[22] A. Bera. *80 Insightful 'Internet of Things' Statistics (Infographic)*. 2021. URL: https://safeatlast.co/blog/iot-statistics/.

[23] Sebastian Bittl, Arturo Gonzalez, and Wolf Heidrich. "Performance comparison of encoding schemes for ETSI ITS C2X communication systems". In: (Aug. 2016).

[24] *Blog Stephane Bortzmeyer: Accueil — bortzmeyer.org*. https://www.bortzmeyer.org/. [Accessed 27-07-2023].

[25] *Blog stephane Bortzmeyer: RFC 8427: Representing DNS Messages in JSON — bortzmeyer.org*. https://www.bortzmeyer.org/8427.html. [Accessed 28-07-2023].

[26] Daniele Bonetta and Matthias Brantner. "FAD.js". In: *Proceedings of the VLDB Endowment* 10.12 (Aug. 2017), pp. 1778–1789. DOI: 10.14778/3137765.3137782. URL: https://doi.org/10.14778/3137765.3137782.

[27] C Bormann and P Hoffman. "Concise Binary Object Representation (CBOR) Tag for CBOR Templates". In: *IETF, DRAFT* 2 (2018).

[28] N. Boudriga. *Security of mobile communications*. CRC Press, 2010.

[29] Pierre Bourhis et al. In: *Foundations of JSON schema*. 2016, pp. 1263–273.

[30] Pierre Bourhis et al. "JSON: Data Model, Query Languages and Schema Specification". In: *Proceedings of the 36th ACM SIGMODSIGACT- SIGAI Symposium on Principles of Database Systems (Chicago, Illinois, USA) (PODS '17). Association for Computing Machinery, New York, NY, USA*. 2017, pp. 123–135. URL: https://doi.org/10.1145/3034786.3056120.

[31] A. Mankin C. Huitema S. Dickinson. "RFC9250: DNS over Dedicated QUIC Connections". In: (May 2022). URL: https://www.rfc-editor.org/rfc/rfc9250/.

[32] Francesco Cesarini, Viviana Pappalardo, and Corrado Santoro. "A comparative evaluation of imperative and functional implementations of the imap protocol". In: Sept. 2008, pp. 29–40. DOI: 10.1145/1411273.1411279.

[33] Feng-Cheng Chang and Duen-Kai Chen. "The Design of an XMPP-based Service Integration Scheme". In: *2011 Seventh International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. 2011, pp. 33–36. DOI: 10.1109/IIHMSP.2011.45.

[34] Rishabh Chhabra et al. "Measuring DNS-over-HTTPS performance around the world". In: *Proceedings of the 21st ACM Internet Measurement Conference*. ACM, Nov. 2021. DOI: 10.1145/3487552.3487849. URL: https://doi.org/10.1145/3487552.3487849.

[35] *ChirpStack open-source LoRaWAN Network Server chirpstack.io*. https://www.chirpstack.io/. [Accessed 10-10-2023].

[36] *Couverture LoRa® Orange.business.com*. https://www.orange-business.com/fr/reseau-iot. [Accessed 20-09-2023]. September 2023.

[37] J. Damas, M. Graff, and P. Vixie. "RFC6891: Extension Mechanisms for DNS". In: (April 2013). URL: https://www.rfc-editor.org/rfc/rfc6891.

[38] Hamza Ed-douibi, Javier Luis Cánovas Izquierdo, and Jordi Cabot. "Example-DrivenWeb API Specification Discovery". In: *Modelling Foundations and Applications, Anthony Anjorin and Huáscar Espinoza (Eds.)* 2017, pp. 267–284.

[39] *Eduroam Supporting Services*. 2021. URL: https://monitor.eduroam.org.

[40] Paola Espinoza-Arias, Daniel Garijo, and Oscar Corcho. "Mapping the Web Ontology Language to the OpenAPI Specification." In: *International Conference on Conceptual Modeling Springer. Springer, Springer, Vienna, Austria,* 2020, pp. 117–127.

[41] Jianhua Feng and Jinhong Li. "Google protocol buffers research and application in online game". In: *IEEE Conference Anthology*. IEEE, Jan. 2013. DOI: 10.1109/anthology.2013.6784954. URL: https://doi.org/10.1109/anthology.2013.6784954.

[42] Javier Fernández et al. "Binary RDF Representation for Publication and Exchange (HDT)". In: *Journal of Web Semantics* 19 (Mar. 2013), pp. 22–41. DOI: 10.1016/j.websem.2013.01.002.

[43] Roy T. Fielding et al. "Reflections on the REST architectural style and "principled design of the modern web architecture" (impact paper award)". In: *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, Aug. 2017. DOI: 10.1145/3106237.3121282. URL: https://doi.org/10.1145/3106237.3121282.

[44] Francesco Flammini et al. "LoRa WAN Roaming for Intelligent Shipment Tracking". In: *2020 IEEE Global Conference on Artificial Intelligence and Internet of Things (GCAIoT)*. IEEE, Dec. 2020. DOI: 10.1109/gcaiot51063.2020.9345843.

[45] Michael Fruth et al. "Challenges in Checking JSON Schema Containment over Evolving Real-World Schemas". In: *International Conference on Conceptual Modeling. Springer, Springer, Vienna, Austria,* 2020, pp. 220–230. URL: https://doi.org/10.1145/3034786.3056120.

[46] Dan Garcia-Carrillo and Rafael Marin-Lopez. "Lightweight CoAP-Based Bootstrapping Service for the Internet of Things". In: *Sensors* 16.3 (2016). ISSN: 1424-8220. DOI: 10.3390/s16030358. URL: https://www.mdpi.com/1424-8220/16/3/358.

[47] Dan Garcia-Carrillo et al. "A CoAP-Based Network Access Authentication Service for Low-Power Wide Area Networks: LO-CoAP-EAP". In: *Sensors* 17.11 (2017).

ISSN: 1424-8220. DOI: 10.3390/s17112646. URL: https://www.mdpi.com/1424-8220/17/11/2646.

[48] Bruno Gil and Paulo Trezentos. "Impacts of data interchange formats on energy consumption and performance in smartphones". In: *Proceedings of the 2011 Workshop on Open Source and Design of Communication*. ACM, July 2011. DOI: 10.1145/2016716.2016718. URL: https://doi.org/10.1145/2016716.2016718.

[49] Bruno Gil and Paulo Trezentos. "Impacts of data interchange formats on energy consumption and performance in smartphones". In: (July 2011). DOI: 10.1145/2016716.2016718.

[50] Google. *Cloudflare DNS over HTTP*. https://developers.cloudflare.com/1.1.1.1/encryption/dns-over-https/make-api-requests/dns-json/. Accessed: [25/01/2023]. 2018.

[51] Google. *Google Public DNS over HTTPS (DoH) supports*. https://developers.google.com/speed/public-dns/docs/doh/json?hl=frgoogle-public-dns-over-https-doh.htm. Accessed: [25/01/2023]. 2018.

[52] A. Gulbrandsen, P. Vixie, and L. Esibov. *RFC2782: A DNS RR for specifying the location of services (DNS SRV)*. Tech. rep. 2000.

[53] Andrew Habib et al. "Type Safety with JSON Subschema". In: 2019, pp. 123–135. URL: https://doi.org/10.1145/3034786.3056120.

[54] Mohamed Hamnache, Rahim Kacimi, and Andre-Luc Beylot. "Unifying LoRaWAN Networks by Enabling the Roaming Capability". In: *2021 IEEE 46th Conference on Local Computer Networks (LCN)*. IEEE, Oct. 2021. DOI: 10.1109/lcn52139.2021.9524996.

[55] Wes Hardaker. *RFC 7671: The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance — datatracker.ietf.org*. https://datatracker.ietf.org/doc/html/rfc7671. [Accessed 28-07-2023]. 2015.

[56] Michael Hauben. "History of ARPANET: Behind the Net - The untold history of the ARPANET". In: (2007). URL: https://www.bibliotecapleyades.net/sociopolitica/sociopol_DARPA10.htm.

[57] P. Hoffman. "RFC8484 : DNS Queries over HTTPS (DoH)". In: *IETF* (2018).

[58] *Home.tele2iot.com*. https://tele2iot.com/. [Accessed 21-09-2023].

[59] *Homepage - LoRa Alliance®lora-alliance.org*. https://lora-alliance.org/. [Accessed 21-09-2023].

[60] Z. Hu et al. "RFC7858: Specification for DNS over Transport Layer Security (TLS)". In: (May 2016). URL: https://www.rfc-editor.org/rfc/rfc7858.

[61] C. Huitema, S. Dickinson, and A. Mankin. "RFC8094: DNS over Datagram Transport Layer Security (DTLS)". In: (Mar 2017). URL: https://www.rfc-editor.org/rfc/rfc8094/.

[62] Markel Iglesias-Urkia et al. "Integrating Electrical Substations Within the IoT Using IEC 61850, CoAP, and CBOR". In: *IEEE Internet of Things Journal* 6.5 (Oct. 2019), pp. 7437–7449. DOI: 10.1109/jiot.2019.2903344. URL: https://doi.org/10.1109/jiot.2019.2903344.

[63] Eduardo Ingles-Sanchez et al. "Adaptation of EAP-NOOB Method for LoRaWAN with LO-CoAP-EAP and CBOR". In: *2020 Global Internet of Things Summit (GIoTS)*. IEEE, June 2020. DOI: 10.1109/giots49054.2020.9119629. URL: https://doi.org/10.1109/giots49054.2020.9119629.

[64] Massimiliano Izzo. "The JSON-Based Data Model". In: *Springer Theses*. Springer International Publishing, 2016, pp. 39–48. DOI: 10.1007/978-3-319-31241-5_3. URL: https://doi.org/10.1007/978-3-319-31241-5_3.

[65] Sukriti Jalali. "M2M solutions — Design challenges and considerations". In: *2013 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*. 2013, pp. 210–214. DOI: 10.1109/RAICS.2013.6745475.

[66] YeiBeech Jang and SeoungHo Ryu. "Exploring Game Leadership and Online Game Community". In: *2009 Conference in Games and Virtual Worlds for Serious Applications*. IEEE, Mar. 2009. DOI: 10.1109/vs-games.2009.29. URL: https://doi.org/10.1109/vs-games.2009.29.

[67] Jaeyeon Jung et al. "DNS performance and the effectiveness of caching". In: *Proceedings of the First ACM SIGCOMM Workshop on Internet Measurement - IMW '01*. ACM Press, 2001. DOI: 10.1145/505202.505223. URL: https://doi.org/10.1145/505202.505223.

[68] Bill Karakostas. "A DNS Architecture for the Internet of Things: A Case Study in Transport Logistics". In: *Procedia Computer Science* 19 (2013). The 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013), the 3rd International Conference on Sustainable Energy Information Technology (SEIT-2013), pp. 594–601. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2013.06.079. URL: https://www.sciencedirect.com/science/article/pii/S187705091300687X.

[69] kelleyvice-msft. *Réseaux de distribution de contenu - Microsoft 365 Enterprise — learn.microsoft.com*. https://learn.microsoft.com/fr-fr/microsoft-365/enterprise/content-delivery-networks?view=o365-worldwide. [Accessed 28-07-2023].

[70] Scott Kitterman. *RFC 7208: Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1 — datatracker.ietf.org*. https://datatracker.ietf.org/doc/html/rfc7208. [Accessed 28-07-2023]. 2014.

[71] Meike Klettke, Uta Störl, and Stefanie Scherzinger. "Schema extraction and structural outlier detection for JSON-based nosql data stores". In: *Datenbanksysteme für Business, Technologie undWeb (BTW 2015), Thomas Seidl, Norbert Ritter, Harald Schöning, Kai-Uwe Sattler, Theo Härder, Steffen Friedrich, and Wolfram Wingerath (Eds.). Gesellschaft für Informatik e.V., Bonn,* 2015, pp. 425–444.

[72] Suresh Krishnaswamy, Wes Hardaker, and Russ Mundy. "DNSSEC in Practice: Using DNSSEC-Tools to Deploy DNSSEC". In: *2009 Cybersecurity Applications Technology Conference for Homeland Security*. 2009, pp. 3–15. DOI: 10.1109/CATCH.2009.21.

[73] Jonathan Kua, Grenville Armitage, and Philip Branch. "A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming Over HTTP". In: *IEEE Communications Surveys Tutorials* 19.3 (2017), pp. 1842–1866. DOI: 10.1109/COMST.2017.2685630.

[74] M. Kucherawy and E. Zwicky. *RFC 7489: Domain-based Message Authentication, Reporting, and Conformance (DMARC) — rfc-editor.org*. https://www.rfc-editor.org/rfc/rfc7489.html. [Accessed 28-07-2023]. 2015.

[75] Pavel Kyurkchiev. "Integrating a System for Symbol Programming of Real Processes with a Cloud Service". In: 2015, pp. 166–175.

[76] RSA Laboratories. "PKCS11 cryptographic token interface standard, version 2.40". CRC press. 2014.

[77] Geoff Langdale and Daniel Lemire. "Parsing gigabytes of JSON per second". In: 10.12 (Aug. 2019), pp. 941–960.

[78] H. Ledoux et al. "CityJSON: a compact and easy-to-use encoding of the CityGML data model". In: 2019, pp. 2363–7501.

[79] Keuntae Lee et al. "A framework for DNS naming services for Internet-of-Things devices". In: *Future Generation Computer Systems* 92 (2019), pp. 617–627. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2018.01.023. URL: https://www.sciencedirect.com/science/article/pii/S0167739X17317429.

[80] Arnol Lemogue et al. "Federated IoT Roaming using Private DNS Resolutions". In: *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. 2022, pp. 1–6. DOI: 10.1109/NOMS54207.2022.9789852.

[81] M.S. Lenders et al. "DNS Queries over CoAP (DoC) - draft-lenders-dns-over-coap-02". In: *IETF* (oct. 2021).

[82] Martine S. Lenders et al. *Securing name resolution in the IoT: DNS over CoAP*. 2022. DOI: 10.48550/ARXIV.2207.07486. URL: https://arxiv.org/abs/2207.07486.

[83] Martine Sophie Lenders et al. *DNS Queries over CoAP (DoC)*. Internet-Draft draft-lenders-dns-over-coap-02. Work in Progress. Internet Engineering Task Force, Oct. 2021. 13 pp. URL: https://datatracker.ietf.org/doc/html/draft-lenders-dns-over-coap-02.

[84] Chi-Yu Li et al. "Transparent AAA Security Design for Low-Latency MEC-Integrated Cellular Networks". In: *IEEE Transactions on Vehicular Technology* 69.3 (2020), pp. 3231–3243. DOI: 10.1109/TVT.2020.2964596.

[85] Yinan Li et al. "Mison". In: *Proceedings of the VLDB Endowment* 10.10 (June 2017), pp. 1118–1129. DOI: 10.14778/3115404.3115416. URL: https://doi.org/10.14778/3115404.3115416.

[86] *Lightweight Authenticated Key Exchange (lake)-datatracker.ietf.org*. https://datatracker.ietf.org/wg/lake/about/. [Accessed 28-07-2023].

[87] Jonathan Katz; Yehuda Lindell. "Introduction to modern cryptography". CRC press. 2014.

[88] *LoRaWAN 1.1 Specification. Technical standard*. Tech. rep. LoRa Alliance, 2017.

[89] Raspberry Pi Ltd. *Buy a Raspberry Pi 3 Model B – Raspberry Piraspberrypi.com*. https://www.raspberrypi.com/products/raspberry-pi-3-model-b/. [Accessed 18-10-2023].

[90] Kevin J. Ma and Radim Bartos. "HTTP Live Streaming Bandwidth Management Using Intelligent Segment Selection". In: *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*. 2011, pp. 1–5. DOI: 10.1109/GLOCOM.2011.6133856.

[91] Aditya Agarwal Mark Slee and Marc Kwiatkowski. "Facebook". In: *Thrift: Scalable cross-language services implementation*, p. 8.

[92] Claudio Marques. *Benign and malicious domains based on DNS logs*. 2021. DOI: 10.17632/623SSHKDRZ.5. URL: https://data.mendeley.com/datasets/623sshkdrz/5.

[93] John Preuß Mattsson et al. *CBOR Encoded X.509 Certificates (C509 Certificates)*. Internet-Draft draft-ietf-cose-cbor-encoded-cert-03. Work in Progress. Internet Engineering Task Force, Jan. 2022. 57 pp. URL: https://datatracker.ietf.org/doc/html/draft-ietf-cose-cbor-encoded-cert-03.

[94] M. Mealling and R. Daniel. *RFC2915: The Naming Authority Pointer (NAPTR) DNS Resource Record*. Tech. rep. 2000.

[95] Dwight Merriman et al. “BSON Specification Version 1.1. MongoDB”. In: 2020, pp. 123–135. URL: http://bsonspec.org/spec.html.

[96] Dwight Merriman et al. “MongoDB”. In: *BSON Specification Version 1.1*. URL: http://bsonspec.org/spec.html.

[97] *MessagePack: It's like JSON. but fast and small. — msgpack.org*. https://msgpack.org/index.html. [Accessed 04-Jan-2023].

[98] “Microsoft”. In: *Microsoft. 2018. Bond Compiler 0.12.0.1*. URL: https://microsoft.github.io/bond/manual/compiler.html.

[99] A. Minaburo, L. Toutain, and R. Andreasen. “RFC8824: Static Context Header Compression (SCHC) for the Constrained Application Protocol (CoAP)”. In: (June 2021). URL: https://www.rfc-editor.org/rfc/rfc8824.

[100] A. Minaburo et al. “RFC8724: SCHC : Generic Framework for Static Context Header Compression and Fragmentation”. In: (April 2020). URL: https://www.rfc-editor.org/rfc/rfc8724.

[101] Nilotpal Mitra. “Efficient Encoding Rules for ASN.1-Based Protocols”. In: *AT&T Technical Journal* 73.3 (May 1994), pp. 80–93. DOI: 10.1002/j.1538-7305.1994.tb00590.x. URL: https://doi.org/10.1002/j.1538-7305.1994.tb00590.x.

[102] P. Mockapetris. “RFC1034: Concept and Facilities”. In: (1984). URL: https://www.rfc-editor.org/rfc/rfc1034.

[103] P. Mockapetris. *RFC1035: DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION*. Tech. rep. 1984. URL: https://datatracker.ietf.org/doc/html/rfc1035.

[104] P. Mockapetris. “RFC882: DOMAIN NAMES - CONCEPTS and FACILITIES”. In: (November 1983). URL: https://www.rfc-editor.org/rfc/rfc882.

[105] P. Mockapetris. “RFC883: DOMAIN NAMES - IMPLEMENTATION and SPECIFICATION”. In: (November 1983). URL: https://www.rfc-editor.org/rfc/rfc883.

[106] Michael Mühlhauser, Henning Pridöhl, and Dominik Herrmann. “How Private is Android's Private DNS Setting? Identifying Apps by Encrypted DNS Traffic”. In: *The 16th International Conference on Availability, Reliability and Security*. ACM, Aug. 2021. DOI: 10.1145/3465481.3465764. URL: https://doi.org/10.1145/3465481.3465764.

[107] The Things Network. *The Things Networkthethingsnetwork.org*. https://www.thethingsnetwork.org/. [Accessed 10-10-2023].

[108] Gerald Neufeld and Son Vuong. “An overview of ASN.1”. In: *Computer Networks and ISDN Systems* 23.5 (1992), pp. 393–415. ISSN: 0169-7552. DOI: https://doi.

org/10.1016/0169-7552(92)90014-H. URL: https://www.sciencedirect.com/science/article/pii/016975529290014H.

[109] Mattias Nordahl and Boris Magnusson. "A lightweight Data Interchange Format for Internet of Things in the PalCom Middleware Framework". In: 2015, pp. 284–291.

[110] Nils Agne Nordbotten. "XML and Web Services Security Standards". In: *IEEE Communications Surveys Tutorials* 11.3 (2009), pp. 4–21. DOI: 10.1109/SURV.2009.090302.

[111] Wouter van Oortmerssen. 2014. "Google". In: *FlatBuffers: Writing a Schema*. URL: https://google.github.io/flatbuffers/flatbuffers_guide_writing_schema.html.

[112] Wouter van Oortmerssen. 2017. "Google". In: *FlexBuffers*. URL: https://google.github.io/flatbuffers/flexbuffers.html.

[113] Shoumik Palkar et al. "Filter before you parse". In: *Proceedings of the VLDB Endowment* 11.11 (July 2018), pp. 1576–1589. DOI: 10.14778/3236187.3236207. URL: https://doi.org/10.14778/3236187.3236207.

[114] Francesca Palombini et al. "RFC Draft: Using EDHOC with CoAP and OSCORE". In: (July 2023). URL: https://datatracker.ietf.org/doc/draft-ietf-core-oscore%20edhoc/.

[115] Philippe Pegon. *Fiabilisation d'une architecture DNS*. Tech. rep. 2003. URL: https://2003.jres.org/actes/paper.149.pdf.

[116] Roberto Perdisci et al. "IoTFinder: Efficient Large-Scale Identification of IoT Devices via Passive DNS Traffic Analysis". In: *2020 IEEE European Symposium on Security and Privacy (EuroSP)*. 2020, pp. 474–489. DOI: 10.1109/EuroSP48549.2020.00037.

[117] A. Perez. *VoLTE and ViLTE - Voice and Conversational Video Services over the 4G Mobile Network*. Wiley, 2016.

[118] Bo Petersen et al. "Smart grid serialization comparison: Comparision of serialization for distributed control in the context of the Internet of Things". In: *2017 Computing Conference*. 2017, pp. 1339–1346. DOI: 10.1109/SAI.2017.8252264.

[119] *Pi-hole -github.com*. https://github.com/pi-hole. [Accessed 18-10-2023].

[120] S. Popi et al. "Performance evaluation of using Protocol Buffers in the Internet of Things communication". In: 10.12 (Aug. 2016), pp. 261–265.

[121] Ricardo Queirós. "JSON on Mobile: is there an Efficient Parser?" en. In: (2014). DOI: 10.4230/OASICS.SLATE.2014.93. URL: http://drops.dagstuhl.de/opus/volltexte/2014/4562/.

[122] Tirumaleswar Reddy.K. *RFC 8310: Usage Profiles for DNS over TLS and DNS over DTLS — datatracker.ietf.org*. https://datatracker.ietf.org/doc/html/rfc8310. [Accessed 28-07-2023]. 2018.

[123] E. Rescorla. *RFC8446:The Transport Layer Security (TLS) Protocol Version 1.3*. Tech. rep. 2018. URL: https://datatracker.ietf.org/doc/html/rfc8446.

[124] *RFC 974: Mail routing and the domain system — datatracker.ietf.org*. https://datatracker.ietf.org/doc/html/rfc974. [Accessed 28-07-2023].

[125] *RFCs — ietf.org*. https://www.ietf.org/standards/rfcs/. [Accessed 27-07-2023].

[126] Leonard Richardson and Sam Ruby. "RESTful web services". In: *O'reilly book* (2007).

[127] Thomas Rix, Kai-Oliver Detken, and Marcel Jahnke. "Transformation between XML and CBOR for network load reduction". In: *2016 3rd International Symposium on Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS)*. 2016, pp. 106–111. DOI: 10.1109/IDAACS-SWS.2016.7805797.

[128] R. Rosenbaum. *RFC1464: Using the Domain Name System To Store Arbitrary String Attributes*. Tech. rep. 1993.

[129] Kristina Sahlmann, Alexander Lindemann, and Bettina Schnor. "Binary Representation of Device Descriptions: CBOR versus RDF HDT". In: Sept. 2018.

[130] J. Schaad. "RFC8152: CBOR Object Signing and Encryption (COSE)". In: (July 2017). URL: https://www.rfc-editor.org/rfc/rfc8152.

[131] *Secure e-mail for private and business customers | mailbox.org — mailbox.org*. https://mailbox.org/en/. [Accessed 28-07-2023].

[132] G. Selander et al. "RFC8613: Object Security for Constrained RESTful Environments (OSCORE)". In: (July 2019). URL: https://www.rfc-editor.org/rfc/rfc8613.

[133] Göran Selander, John Preuß Mattsson, and Francesca Palombini. "RFC Draft: Ephemeral Diffie-Hellman Over COSE (EDHOC)". In: (July 2023). URL: https://datatracker.ietf.org/doc/draft-ietf-lake-edhoc/.

[134] *Send Message (SNDMSG) — ibm.com*. https://www.ibm.com/docs/en/i/7.1?topic=ssw_ibm_i_71/cl/sndmsg.html. [Accessed 28-07-2023].

[135] *Set up 1.1.1.1 on macOS · Cloudflare 1.1.1.1 docs — developers.cloudflare.com*. https://developers.cloudflare.com/1.1.1.1/setup/macos/. [Accessed 28-07-2023]. 2020.

[136] Y. Sheffer, R. Holz, and P. Saint-Andre. *RFC 7525: Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) — rfc-editor.org*. https://www.rfc-editor.org/rfc/rfc7525. [Accessed 28-07-2023]. 2015.

[137] Z. Shelby, K. Hartke, and C. Bormann. "RFC7252: The Constrained Application Protocol (CoAP)". In: (June 2014). URL: https://www.rfc-editor.org/rfc/rfc7252.

[138] S. K. Siddiqui. *Roaming in Wireless Networks*. McGraw-Hill, 2006.

[139] Prabath Siriwardena. "Base64 URL Encoding". In: *Advanced API Security: OAuth 2.0 and Beyond*. Berkeley, CA: Apress, 2020, pp. 397–399. ISBN: 978-1-4842-2050-4. DOI: 10.1007/978-1-4842-2050-4_20. URL: https://doi.org/10.1007/978-1-4842-2050-4_20.

[140] *SSL Digital Certificate Authority | Encryption amp; Authentication | DigiCert.com — digicert.com*. https://www.digicert.com/. [Accessed 28-07-2023].

[141] B. Stephane. "RFC8427: Representing DNS Messages in JSON". In: (July 2018). URL: https://www.bortzmeyer.org/8427.html.

[142] Audie Sumaray and S. Makki. "A comparison of data serialization formats for optimal efficiency on a mobile platform". In: (Feb. 2012). DOI: 10.1145/2184751.2184810.

[143] Ed. T. Bray. "The JavaScript Object Notation (JSON) Data Interchange Format". In: (December 2017). URL: https://www.rfc-editor.org/rfc/rfc8259.

[144] "The Apache Software Foundation". In: *The Apache Software Foundation. 2012. Apache Avro™ 1.10.0 Specification*. URL: https://avro.apache.org/docs/current/spec.html.

[145] *The scientific and technological research council of Turkey*. https://www.tubitak.gov.tr/en. [Accessed 28-07-2023].

[146] Elena M. Torroglosa et al. "Enabling Roaming Across Heterogeneous IoT Wireless Networks: LoRaWAN MEETS 5G". In: *IEEE Access* 8 (2020), pp. 103164–103180. DOI: 10.1109/access.2020.2998416.

[147] P. Tzerefos et al. "A comparative study of Simple Mail Transfer Protocol (SMTP), Post Office Protocol (POP) and X.400 Electronic Mail Protocols". In: *Proceedings of 22nd Annual Conference on Local Computer Networks*. 1997, pp. 545–554. DOI: 10.1109/LCN.1997.631025.

[148] Jan Vanura and Pavel Kriz. "Perfomance Evaluation of Java, JavaScript and PHP Serialization Libraries for XML, JSON and Binary Formats". In: June 2018, pp. 166–175. ISBN: 978-3-319-94375-6. DOI: 10.1007/978-3-319-94376-3_11.

[149] Santiago Vargas et al. “Characterizing JSON Traffic Patterns on a CDN.” In: *Proceedings of the Internet Measurement Conference* 7 (2019), pp. 195–201. DOI: 10.1145/3355369.3355594.

[150] *Verisign is a global provider of domain name registry services and internet infrastructure - Verisign — verisign.com*. https://www.verisign.com/. [Accessed 28-07-2023].

[151] Juan Cruz Viotti and Mital Kinderkhedia. *A Benchmark of JSON-compatible Binary Serialization Specifications*. 2022. arXiv: 2201.03051 [cs.SE].

[152] Matthias Wählisch. *A Concise Binary Object Representation (CBOR) of DNS Messages — ietf.org*. https://www.ietf.org/id/draft-lenders-dns-cbor-01.html. [Accessed 04-Jan-2023].

[153] Z C. Wallace and C. Gardiner. “RFC 6025: ASN.1 Translation”. In: (October 2010). URL: https://www.rfceditor.org/rfc/rfc6025.html.

[154] Canggih Wibowo. “Evaluation of Protocol Buffers as Data Serialization Format for Microblogging Communication”. In: Nov. 2011.

[155] K. Wierenga, S. Winter, and T. Wolniewicz. *RFC7593: The Eduroam Architecture for Network Roaming*. Tech. rep. 2015.

[156] Erik Wilde and Cesare Pautasso. “REST: From Research to Pratice”. In: *Springer book* (2011).

[157] Martin Wischenbart et al. “User Profile Integration Made Easy: Model-Driven Extraction and Transformation of Social Network Schemas”. In: *Proceedings of the 21st International Conference on World Wide Web (Lyon, France) (WWW '12 Companion). Association for Computing Machinery, New York, NY, USA,* pp. 939–948. URL: https://doi.org/10.1145/2187980.2188227.

[158] Zhiwei Yan et al. “Is DNS Ready for Ubiquitous Internet of Things?” In: *IEEE Access* 7 (2019), pp. 28835–28846. DOI: 10.1109/access.2019.2901801. URL: https://doi.org/10.1109/access.2019.2901801.

[159] Zhiwei Yan et al. “Is DNS Ready for Ubiquitous Internet of Things?” In: *IEEE Access* 7 (2019), pp. 28835–28846. DOI: 10.1109/ACCESS.2019.2901801.

[160] Qianchuan Ye and Benjamin Delaware. “A verified protocol buffer compiler”. In: *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs*. ACM, Jan. 2019. DOI: 10.1145/3293880.3294105. URL: https://doi.org/10.1145/3293880.3294105.

[161] Xuebiao Yuchi et al. “Modeling DNS Activities Based on Probabilistic Latent Semantic Analysis”. In: *Advanced Data Mining and Applications*. Springer Berlin

Heidelberg, 2010, pp. 290–301. DOI: 10.1007/978-3-642-17313-4_29. URL: https://doi.org/10.1007/978-3-642-17313-4_29.

COLLEGES BRETAGNE LOIRE SCIENCES POUR L'INGENIEUR ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Reduction de l'impact du paradigme REST sur le DNS en utilisant les Technologies pour l'Internet des Objets

Mots clés : DNS, DoH, REST, IoT, CBOR, CoAP, Itinirance, LoRaWAN

Résumé : Le système de noms de domaines (DNS) utilisé dans l'Internet pour établir une correspondance effective entre les identifiants réseaux et leurs déclinaisons conviviales, s'appuie sur une architecture optimisée et résiliente d'une grande stabilité. Toutefois la préservation de l'efficacité des protocoles sous-jacents, n'a pas permis à ces derniers d'évoluer au même rythme que le reste de l'Internet. L'introduction du paradigme REST ouvre néanmoins la voie à l'évolution du DNS via sa RESTification. Cette dernière entraîne cependant une augmentation de la latence et de la verbosité qui pourraient à terme compromettre sa généralisation.
Dans cette thèse, nous nous intéressons à la réduction de l'impact de la RESTification sur le DNS. Notre exploitation des avancées obtenues dans le domaine de l'Internet des objets, nous conduit à deux contributions significatives concernant respectivement la taille des données véhiculées par les protocoles, et l'architecture du DNS.
Nous définissons ainsi un nouveau format reposant sur CBOR pour encoder efficacement les messages DNS. Nous introduisons une nouvelle représentation appelée efficient CBOR (e-CBOR) qui fournit des formats DNS/DNSSEC compacts et flexibles tout en préservant la compatibilité avec les protocoles existants. Les tests effectués sur l'implémentation d'e-CBOR montrent une réduction substantielle par rapport à la taille des messages encodés dans le format traditionnel. La seconde contribution concerne la conception et la mise en oeuvre d'une nouvelle entité le DNS-Broker. Ce dernier rend possible une résolution DNS privée pour contrôler l'accès à une ressource réseau, tout en réduisant le nombre de messages échangés dans le cas où la résolution est autorisée.
Un démonstrateur permet de valider la pertinence de nos propositions en les appliquant au cas d'usage de l'itinérance entre opérateurs de réseaux LoRaWAN.

Title : Reducing the impact of the REST paradigm on DNS using Technologies for the Internet of Things

Keywords : DNS, DoH, REST, IoT, CBOR, CoAP, Roaming, LoRaWAN

Abstract : The Domain Name System (DNS) used on the Internet to provide an effective correspondence between network identifiers and their user-friendly variants, is based on a highly stable, optimised and resilient architecture. However, maintaining the efficiency of the underlying protocols has not allowed them to keep pace with the rest of the Internet. The introduction of the REST paradigm has nevertheless paved the way for the evolution of DNS via its RESTification. However, the latter leads to an increase in latency and verbosity, which could ultimately compromise its widespread use.
In this thesis we focus on reducing the impact of RESTification on DNS. Leveraging advances in the Internet of Things, we make two significant contributions concerning the size of data carried by protocols, and DNS architecture.
We define a new format based on CBOR for efficient encoding of DNS messages. We present a new representation, called efficient CBOR (e-CBOR), which provides compact and flexible DNS/DNSSEC formats while maintaining compatibility with existing protocols. Tests performed on the e-CBOR implementation show a significant reduction in message size compared to messages encoded in the traditional format.
The second contribution concerns the design and implementation of a new entity called DNS-Broker. It allows private DNS resolutions to control access to network resources, while reducing the number of messages exchanged when these resolutions are authorised.
A demonstrator allows us to validate the relevance of our proposals by applying them to the use case of roaming between LoRaWAN network operators.