République Algérienne Démocratique Et Populaire
Université Mohamed Boudiaf De M'sila
Faculté Des Mathématiques Et De L'informatique
Département D'informatique

MEMOIRE DE FIN D'ETUDE

Présenté pour l'obtention du Diplôme de **MASTER 2**
Domaine : Mathématiques et Informatique
Filière: Informatique
Option : Système d'information et génie logiciel

Par
Ouadah Abdenour

Sujet

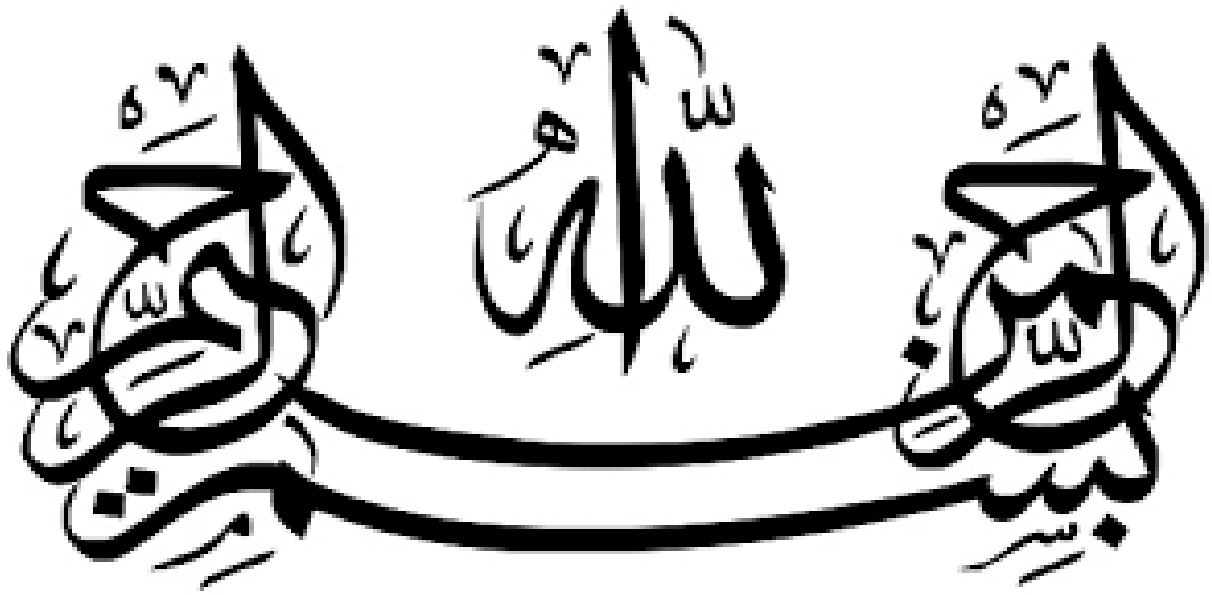
Modélisation et vérification formelle d'un protocole CoAP pour l'internet des objets

Encadré par : Monsieur Dr :**Lekehal Meftah**

Devant le jury :

Mr.Dr Chatra Mohammed	Prof. Univ de M'sila	Président
Mr.Dr Lekehal Meftah	Prof. Univ de M'sila	Rapporteur
Mr. Merzrag Fares	Prof. Univ de M'sila	Examinateur

Promotion : 2018 / 2019



Remerciement

Je tiens tout d'abord à remercier mon Dieu le tout puissant et miséricordieux, qui me a donné la force et la patience d'accomplir ce Modeste travail.

*En second lieu, je tiens à remercier mon encadreur Mr : (**dr : lekehal mefteh**), son précieux conseil et son aide durant toute la période du travail.*

*Mes remerciements vont également aux **membres du jury** pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail Et de l'enrichir par leurs propositions.*

*Mes remerciements vont également a mes collègues du groupe, spécialement à mon frère **Khaled Sharif.***

*Enfin, je tiens également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail et surtout **Mme makhloufi H**.*

Dédicace

Je dédie ce travail à l'esprit pur de mon père et je souhaite d'Allah Tout-Puissant de le faire vivre dans un paradis spacieux

Je dédie aussi ce travail à :

A ma chère mère.

A mes frères (abderrahmane,ahmed,abdessalam, hamza et mohammed) et leur famille et surtout (mahdi, issam)

A mes enfants (aroua, oumima, amar et iyed)

A tous mes amis surtout (bachir chikh ,moussa saadi, redouane ouadah, salah gherbi…etc)

Liste des figures

Liste des tableaux

Liste des abréviations

ARPANET : Advanced Research Projects Agency Network

AMQP : Advanced Message Queuing Protocol

CoAP : Constrained Application Protocol

DoS : Denial Of Service

DNS : Domain Name System

DNSSEC : Domain Name System Security Extensions

ECC : Elliptic Curve Cryptography

EPC : Electronic Product Code

EPCIS :Electronic Product Code Information Services

ETSI :European Telecommunications Standards Institute

GPS : Global Positioning System

HTTP : HyperText Transfer Protocol

HTTPS :Hypertext Transfer Protocol Secure

HVAC: Heating, Ventilation and Air-conditioning

Ipv6 : Internet Protocol version 6

ITU : International Telecommunication Union

RBAC: Role-Based Access Control

RPL : Routing Protocol for Low power and Lossy Networks

RFID : Radio Frequency Identification

RSA : Rivest, Adi Shamir et Leonard Adleman

REST : Representational State Transfer

SOAP: Simple Object Access Protocol

SPIN : Simple Promela INterpreter

TCP/IP : Transmission Control Protocol / Internet Protocol

TLS : Transport Layer Security

UDP : User Datagram Protocol

URI: Uniform Resource Identifier

VPN: Virtual Private Network

W3C : World Wide Web Consortium

WSN : Wireless Sensor Networks

XMPP : Extensible Messaging and Presence Protocol

W3C :World Wide Web Consortium

6loWPan IPv6: Low Power Wireless Area Networks

INTRODUCTION GENERALE

Introduction générale

De nos jours, il n'y a pas que les ordinateurs qui permettent aux humains d'accéder au monde en ligne, comme les téléphones intelligents, qui sont connectés a internet, de plus en plus d'autre appareils sont également en ligne : réfrigérateurs, machines à laver, thermostats et voiture, par exemple .Ces objets du quotidien contiennent des dispositifs informatiques reliés par les réseaux qui permettent d'échanger des données. La vision d'un réseau mondial interconnecté de ces objets capable de fournir des services avancés s'appelle l'internet des objets.

L'interaction entre ces objets nécessite différents types de technologies, dont l'une est la communication protocolaire. Le protocole COAP est l'un des protocoles de communication de l'IOT le plus per metteur pour la couche application. Il est standardisé par L'IETF et conçu pour utilise un minimum de ressource, ce qui le rend particulièrement adaptés aux nœuds et réseaux contraintes.

Comme les systèmes deviennent de plus en plus complexe, il est crucial non seulement d'assurer certaines performances, mais également de garantir l'absence de risques de dysfonctionnement. Pour assurer le bon fonctionnement de protocole COAP, l'implémentation doit respecter les normes et satisfaire les bonnes propriétés attendues. Les méthodes formelles peuvent assurer cette contrainte. Ces méthodes sont basées sur les mathématiques qui les rendent solides pour développer de preuve de correction et garantir l'absence de panne.

L'objectif de notre travail est de modéliser et vérifier formellement le protocole COAP. Nous allons modéliser ce protocole avec les automates temporisés probabilistes et nous allons utiliser l'outil UPPAAL, l'outil fait appel au model chocking statistique qui prend en charge les aspects temporel.

UPPAAL permet de vérifier des propriétés qualitatives (la réponse sera oui ou non) ou quantitatives (la propriété sera sure sous des contraintes particulières et avec une probabilité.

Notre mémoire est constitué une introduction générale, quatre chapitres , une conclusion générale et un résumé :

1- Le premier chapitre concerne les méthodes formelles.

2- La deuxième chapitre : il est consacré à la définition de l’ internet des objets et leurs avantages et leurs inconvénients, l’architecture et la standardisation, aussi le fonctionnement et enfin les protocoles de communication de l’internet des objets.
3- Le troisième chapitre : il est consacré à la définition de protocole COAP, le format de message, ces caractéristiques, son fonctionnement.
4- Dans le dernier chapitre, nous présenterons la modélisation du protocole COAP avec les automates temporisés probabilistes, et enfin la vérification des propriétés avec UPPAAL.

Nous terminons par une conclusion générale qui présente un bilan, et propose certaines perspectives envisagées.

CHAPITRE 01

LES MÉTHODES FORMELLES DE LA VÉRIFICATION

1.1 Introduction

Historiquement, les méthodes formelles ont été définies les premiers temps comme des méthodes associant une logique, un certain ensemble d'axiomes et des techniques de preuve. De nos jours et plus généralement, ces méthodes formelles sont définis comme un cadre mathématique utilise pour modéliser les systèmes et vérifier leur comportement loin de toute ambigüité*.*

Dans ce chapitre. Nous allons présenter les techniques de vérification formelle, nous mettrons l'accent sur le model checking, son problème d'explosion combinatoire et des solutions, Nous allons détailler, par la suite, les formalismes de modélisation du système et de spécification, et présenter les classifications de leur propriétés temporelles, et enfin les outils de modèle checking.

1.2 Méthodes de vérification formelle

Dans cette section, nous nous intéressons aux méthodes qui permettent d'établir formellement que le modèle d'un system respecte ses spécifications. Dans la littérature scientifique **[01]**, il existe deux célèbres méthodes de vérification formelle: la preuve de programmes et le Model- Checking.

1.2.1 Preuve de programme

Avec la preuve ou preuve de théorème, le système et les propriétés sont modélisées grâce à des formules de logique mathématique. La logique comporte des axiomes (qui sont admis) et des règles d'inférence qui permettent de produire les théorèmes. La réalisation de preuve consiste donc à appliquer les règles d'inférences sur le modèle du système. Ce processus peut être réalise à la main ou bien de manière semi-automatique grâce à des logiciels assistants de preuve **[02].**

1.2.2 Model Checking

Le model Checking **[03]** est une méthode de vérification formelle automatique. Le système étudié est décrit sous forme de modèle à états-transitions qui représente sa spécification.

A partir de ce premier modèle est généré un second modèle qui, est une interprétation sémantique du premier et qui représente tous les comportements possibles du système.

Des algorithmes permettent d'explorer exhaustivement de l'espace d'états des comportements, durant cette exploration, l'algorithme de model-checking vérifie certains types de propriétés parmet les quels :

a) Propriétés de model Checking : l'accessibilité :(il existe un chemin d'exécution vers un état donné), la sûreté (vrai quel que soit l'´etat), le blocage (un état n'a pas de transition sortante)

b) Procédure de model Checking : la vérification de tout système en utilisant la technique de model checker passe par trois étapes comme l'explique **la Figure 1.1 [04]**

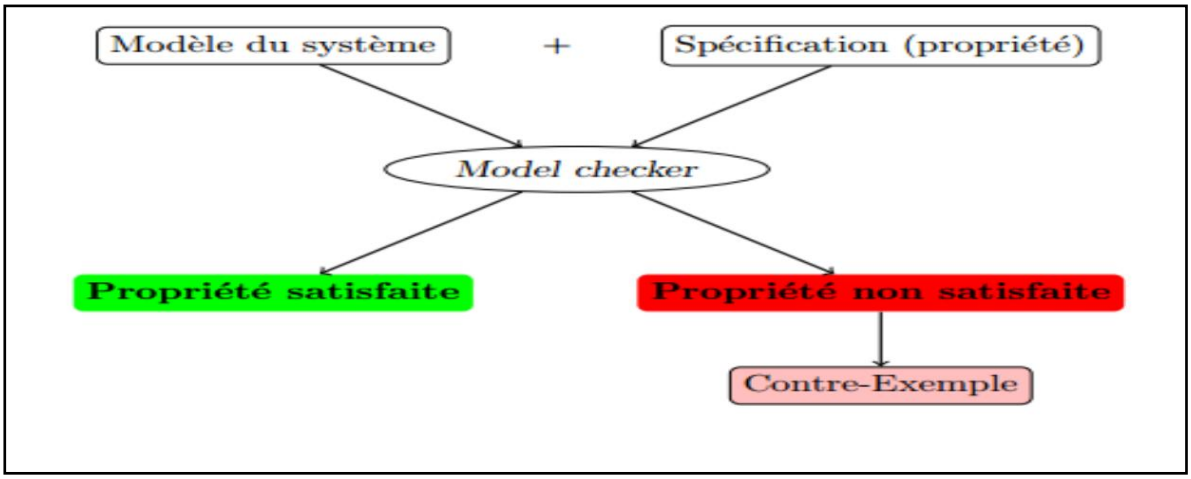


Figure1.1: Procédure du Model-checking

c) Modélisation : cette étape consiste à générer sous forme d'un modèle, une abstraction du système, ce modèle doit être assez expressif pour pouvoir décrire sans ambigüité les spécifiés du système, Il s'agit de traduire le système sous forme d'une structure finie interprétable par un algorithme de model checking. Parmi ces structures de données, on cite) : les automates, les réseaux de Petri, les diagrammes binaires de décision **[05].**

d) Spécification : cette étape permet de traduire les requis (les propriétés attendues) d'un système dans un langage formel. On distingue différents types de requis : la sûreté, la vivacité, l'accessibilité, l'équité et l'absence de blocage, etc. Les logiques temporelles sont les formalismes les plus utilisés pour spécifier des propriétés. **[05].**

e) Vérification: cette étape consiste à mettre en œuvre un algorithme qui prend en entrée un modèle (résultat de l'étape de modélisation) et une spécification (résultat de l'étape spécification). Le modèle checking renvoie vrai si la spécification est vérifiée et fournit un contre exemple (trace d'erreur), le cas échéant. **[05],** le principe de model checking dans **la figure 1.2** comme suite **:**

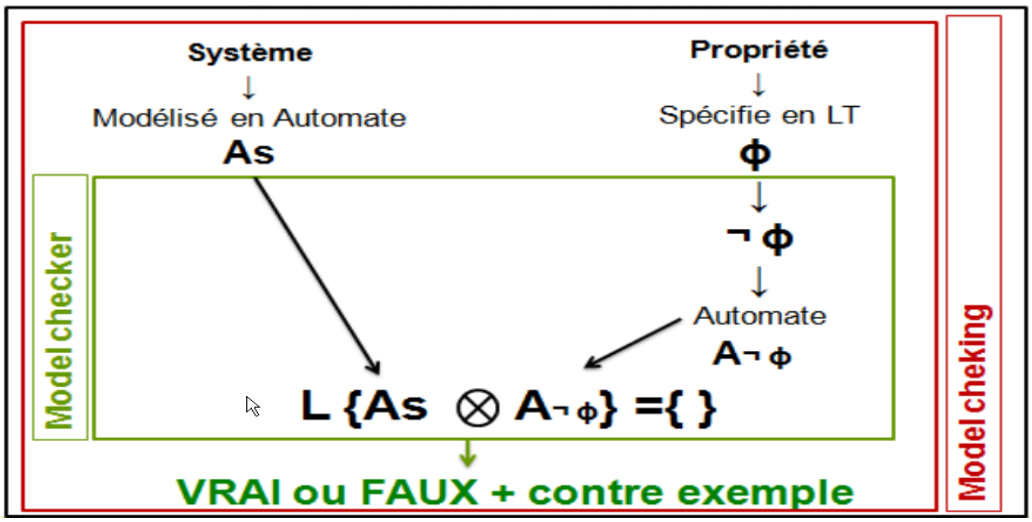


Figure 1.2: Principe de model checking

1.2.3 Les principaux avantages du Model Checking sont [06] :

a- L'exhaustivité de la vérification:

Tous les comportements du modèle sont explorés, cela permet de détecter toutes les violations de spécification.

b- Le fait que la méthode soit automatique:

Ce qui évite les erreurs humaines évoquées dans le cas des preuves et permet de se concentrer sur la qualité de la modélisation.

c-La capacité à fournir des contre-exemples:

Lorsqu'une propriété n'est pas vérifiée, le Model Checker est capable de fournir une trace permettant au concepteur du système de comprendre le problème.

En revanche, l'inconvénient majeur du Model Checking est la possibilité de l'explosion combinatoire. En effet, lorsque le système est complexe, distribué

et/ou large échelle, le nombre de comportements possibles devient très important. Il est alors impossible de stocker ou d'explorer en un temps raisonnable le modèle des comportements du système, ce qui rend la vérification infaisable en pratique.

1.3 Formalismes de modélisation du model checker

Parmi les formalismes de modélisation du model checker, nous nous intéressons, dans ce travail, aux automates temporises ainsi qu'à leurs extensions, notamment les automates probabilistes.

1.3.1 Les automates temporisés (Timed Automata : TA)

Un automate temporisé **[06]** est un automate fini utilisant un ensemble **d'horloges** qui sont des variables définies sur le domaine du temps, noté ***T***, évoluant de manière synchrone et continue dans le temps absolu. Le domaine du temps peut être l'ensemble des entiers naturels ***N***, ou l'ensemble des nombres rationnels positifs ***Q+***, ou, plus habituellement, l'ensemble des nombres réels positifs ***R+***. Dans chaque état, le temps peut s'écouler et la valeur d'une horloge est le temps écoulé depuis la dernière **initialisation** à **0** de cette horloge. Chaque transition de l'automate est instantanée, mais elle comporte une contrainte sur les horloges de l'automate, appelée **une garde**, qui indique quand celle-ci peut être exécutée. Les transitions peuvent réinitialiser une ou plusieurs d'horloges lorsqu'elles sont exécutées. De la même façon, chaque état de l'automate comporte **une contrainte** sur les horloges, appelée un **invariant [08]**, qui restreint la durée de stationnement dans cet état, et donc peut forcer l'exécution d'une transition.

Un automate temporise admet deux types de transitions :
-Une transition d'action qui consiste à franchir une transition quand sa garde est vraie.
-Une transition de durée qui consiste à séjourner dans le même état avec l'´evolution continue d'horloges en respectant l'invariant :

a-Syntaxe d'un A.T

Un automate temporisé A est un 6-uplet **(*Q, q*0, *X,* Σ, *Inv, T*)** ou **[09] :**

- *Q* est un ensemble fini et non vide d'´etats,
- *q*0 ∈ *Q* est l'´etat initial,
- *X* est un ensemble fini d'horloges. Soit *C*(*X*) l'ensemble de contraintes d'horloges appelées invariants, et $\mathbf{2}^{\boldsymbol{X}}$ l'ensemble de tous les sous-ensembles de *X*,
- Σ est un alphabet fini qui définit un ensemble d'actions,

• Inv : Q → C(X) est une application qui associe a chaque état un ensemble de contraintes d'horloges de la forme **x ∼ c ou` ∼ ∈ {<, <=, =, >=, >},**

C est une constante et **x ∈ X**.

• $T \subseteq Q * C(X) * \Sigma\ \mathbf{2^X} * Q$ est un ensemble fini de transitions. Chaque élément $t = (p, g, a, r, q)$ T définit une transition de l'´etat p vers l'´etat q en spécifiant sa garde g, son action a, et le sous ensemble d'horloges r a remettre à zéro suite a son franchissement.

b-Exemple :

Un automate temporisé simple **[09]** dans la **Figure 1.3**,

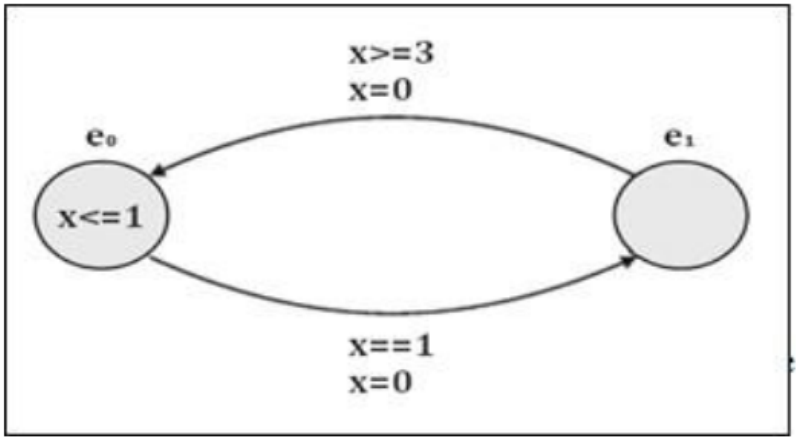


Figure 1.3 : Exemple d'automate Temporisé

L'automate possède :

- d'une horloge x.
- deux états e_0 et e_1. L'automate séjourne dans l'état e0 jusqu'à une unité de temps.
- Quand l'horloge x est égale à 1, l'automate franchit instantanément la transition dont la garde est satisfaite (x == 1) et réinitialise l'horloge x. Puisqu'il n'y ait pas d'invariant défini sur l'état e1,
- l'automate peut y séjourner au moins jusqu'à ce que la garde de sa prochaine destination soit satisfaite (x ≥ 3). Autrement dit, l'automate doit séjourner au moins trois unités de temps dans l'état e_1.

1.3.2 Les automates temporisés probabilistes

Les automates temporisés probabiliste (Probabilistic Time Automata : Prob.TA) **[10]** sont une extension des automates temporisée classiques, qui en plus du choix non déterministe entres les transitions permettent un choix probabiliste.

a-Définition : un automate temporisé probabiliste **[11]** est un tuple **(L, ¯ l, χ, Σ, inv, prob)** ou :

- **L** : un ensemble fini de localités (ou états) et ¯ l ⊆ L;
- **Σ** : un ensemble d'événements.
- **χ** : un ensemble de variables (dites horloges). χ ∈ T et T ∈R, N;
- **inv** : L → Zone {χ} une fonction qui associe a chaque localité un invariant.

Zone {χ} est l'ensemble **des expressions logiques** (des gardes) sous la forme: $x \sim c$, tel que $x \in \chi$, $\sim \in \{\leq, =, \geq\}$ et $c \in N$;

- **prob:** un ensemble fini de transitions probabilisées:
- **prob** $\subseteq$ $L^* Zone(\chi) ^*\Sigma ^*Dist (2^X * L)$ est l'ensemble de distributions probabilistes sur tous les sous ensembles dénombrables du produit: $2^X * L$. Autrement dit, chaque transition lie une localité vers une autre localité, et cette transition est étiquetée par: un événement, une garde, un ensemble de variables a **remettre à zéro**, et enfin une probabilité de transition. Donc une transition probabiliste est un tuple **(*l, g, σ, p*)** tel que *p* est une fonction de probabilité :

p = μ(X, lJ) définie pour chaque couple (X, lJ) ∈ 2^X × L; avec X ⊆ χ.

b-Exemple : Un automate temporisé probabiliste simple

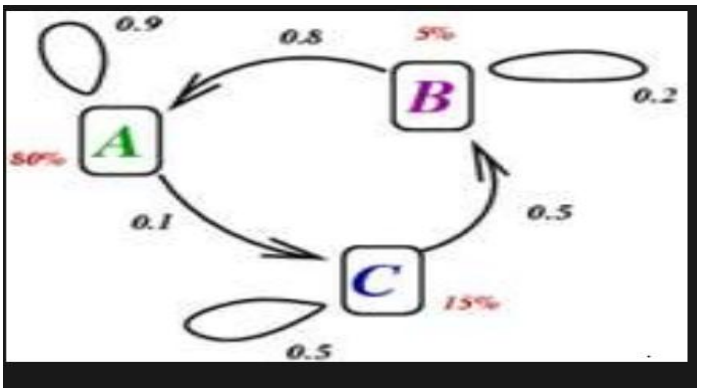


Figure 1.4 : Automate temporisés probabiliste

1.4 Formalisme de Spécification des propriétés des systèmes

Généralement, le model checking utilise une logique temporelle afin de spécifier les propriétés décrivant les requis du système étudié

1.4.1 Logique temporelle

Une logique temporelle est un langage de spécification qui permet d'exprimer des propriétés faisant intervenir la notion d'ordonnancement dans le temps. Une logique temporelle permet d'exprimer des propriétés de type après une instruction i un événement j se produira en utilisant des opérateurs temporels **[12].**On distingue principalement deux classes de logiques temporelles :

1.4.1. a Logiques temporelles linéaires (Linear Temporal Logic : LTL)

Permet de décrire le comportement des systèmes qui évoluent linéairement dans le temps. En effet, LTL permet d'exprimer des propriétés portant sur une séquence d'états donnée appelée exécution **[13][14].**

1.4.1.b Logiques temporelles arborescentes (Computation Tree Logique)

Le CTL permet d'analyser plusieurs états futurs à partir d'un état donné du système en considérant plusieurs chemins d'exécution **[15].**

La logique temporelle arborescente **[15]** permet d'exprimer des propriétés portant sur les arbres d'exécution qui représentent l'évolution du système.

a- Les Formules CTL sont composée de :

- Des propositions atomiques **: a,b, ….**
- Les connecteurs booléens :¬ (négation), ***c*** (et logique), ∨ (ou logique), ⇔ (équivalence), ⇒ (implication).
- Les opérations temporels **:** EF p | EG p | E p U q | EX p | AF p | AG p | A p U q | AX p avec E et A sont des quantificateurs de chemins.

b- La syntaxe d'une formule CTL:

$\phi, \Psi ::= \phi\ \phi\ \Psi\ \phi\ \Psi\ \phi\ \Psi\ \phi\ \Psi$ EF ϕ EG ϕ E ϕ U Ψ EX ϕ AF ϕ AG ϕ A ϕ U Ψ AX ϕ avec ϕ et Ψ sont deux formules CTL. Les formules CTL peuvent être interprétées comme suit :

- ***EFϕ*** : Il existe une séquence d'´etats (opérateur E) qui mène à un état ou` la propriété ϕ est vérifiée (opérateur F).
- ***AFϕ*** : Pour toute séquence d'´etats (opérateur A), il existe un état ou` la propriété ϕ est vérifiée (opérateur F).

-EGϕ : Il existe une séquence d'´etats ou` la propriété ϕ est toujours vérifiée (opérateur G).

-AGϕ : Pour toute séquence d'´etats, la propriété ϕ est toujours vérifiée (opérateur G).

-AXϕ : La propriété ϕ est vérifiée sur tous les états immédiatement successeurs de l'´etat courant (opérateur X).

-EXϕ : Il existe une séquence d'´etats ou` (opérateur X) satisfait la propriété ϕ.

il existe un état dont l'´etat successeur A ϕ U Ψ : Pour toute séquence d'´etats la propriété ϕ est vérifiée jusqu'à ce que la propriété Ψ Soit satisfaite.

E ϕ U Ψ : Il existe une séquence d'´etats sur laquelle la propriété ϕ est vérifiée jusqu''a ce que la propriété Ψ soit satisfaite.

Certaines formules CTL sont illustrées par **la figure 1.5.**

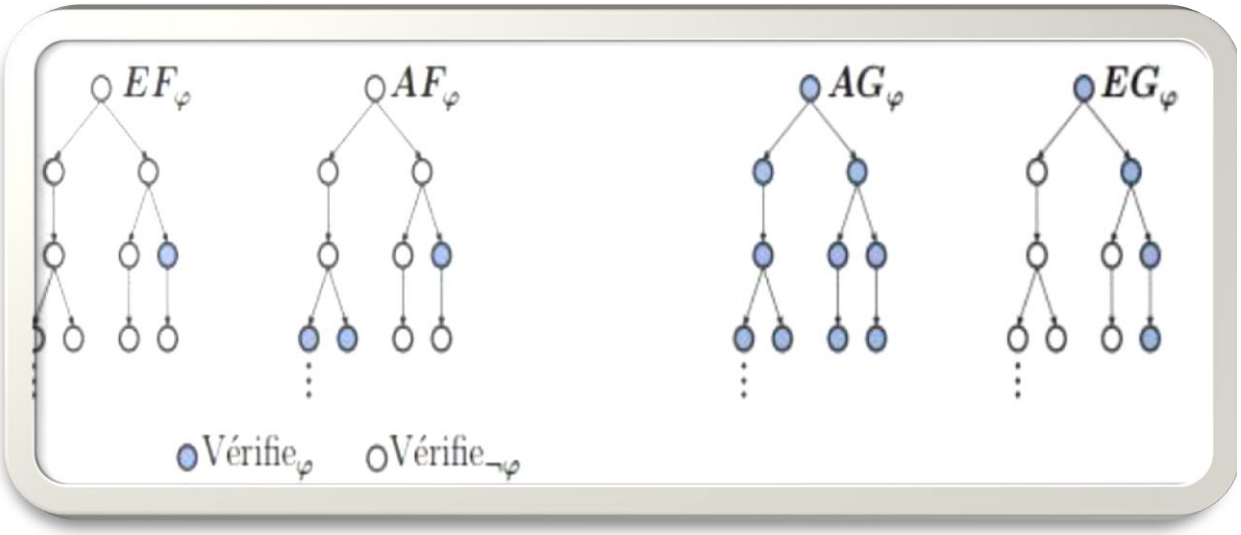


Figure1.5 : Illustration de certaines formules CTL

c- Sémantique de CTL :

Formellement, CTL est interprétée sur les états d'un système de transitions (qui caractérise les arbres issus de ces états).

Il ya deux types de la Logiques temporelles arborescentes comme suit :

a- La logique TCTL (Timed Computation Tree Logic)

Le TCTL **[16]** est une extension de la logique temporelle CTL qui permet d'exprimer des propriétés faisant intervenir des quantifications temporelles (ex: des propriétés de type un événement i se produit en moins de quatre secondes avant qu'un autre événement j ne soit produit). La syntaxe de la logique temporelle TCTL se base sur le formalisme de la logique

temporelle CTL en utilisant des propositions faisant intervenir des contraintes sur les horloges:

La syntaxe

φ, ψ **::= p** $|...|\neg\varphi| \vee \wedge\psi|\varphi \vee \psi|\varphi \Leftrightarrow \psi|\varphi \Rightarrow \psi|$ **EF** $\varphi|$ **EG** $\varphi|$ **E** φ **U** $\psi|$ **A** φ **U** $\psi|$ **EX** $\varphi|$ **AF** $\varphi|$ **AG** $\varphi|$ **AX** $\varphi|$ **x** $\sim$ **c** $|$ $x - y \sim$ **c ,** ou` φ et ψ sont deux formules CTL, $\sim \in \{<, >, \leq, \geq, =\}$ et $c \in N$, x et y sont des horloges et p est une proposition atomique.

Par exemple AG (erreur AF (x 3 alarme) est une formule TCTL qui vérifie le déclenchement éventuel d'une alarme dans un délai inférieur `a trois unités de temps en cas d'erreur

b- La logique PCTL (Probabilistic Computation Tree Logic)

La logique temporelle PCTL [67] (Probabilistic Computation Tree Logic) permet de spécifier des propriétés sur des chaînes de Markov `a temps discrets. Elle a ´et´e définie en 1994 par Hans Hansson and Bengt Jonsson, elle étend la logique temporelle CTL par l'ajout de l'opérateur probabiliste P, et par les extensions quantitatives des opérateurs A et E.

La syntaxe

Formule PCTL est définie par **:**

- Des formules d'´etats **: φ := true|a|φ ∧ φ|¬φ|P∼p[ψ]** ;
- Des formules de chemins **: ψ := X φ|φ U ≤k φ|φ U φ ou:**
- a est une proposition atomique utilisée pour identifier les états qui intéressent l'analyse;
- $\sim \in \{<, >, \leq, \geq\}$ et p ∈ [0,1] est une probabilité;
- k ∈ N;
- Xφ signifie φ est vraie a` la prochaine tape (operateur next);
- φ1 U ≤k φ2 signifie φ2 est vraie en k étapes et φ1 est vraie jusqu'`a ce point (opérateur bounded until);
- φ1 U φ2 signifie φ2 est finalement vraie et φ1 est vraie jusque là (opérateur until).

Une propriété s'exprime toujours `a l'aide d'une formule d'´etat. Les formules de chemins ne peuvent apparaitre qu'`a l'intérieur de l'operateur probabiliste P. La formule d'état P∼p[ψ] signifie que (la formule de chemin ψ est vraie avec la probabilité p). Le résultat de l'analyse de cette propriété qualitative est vrai ou faux. Par ailleurs, il est possible d'´evaluer la probabilité que la formule de chemin ψ soit vraie. La propriété quantitative correspondante est P=[ψ].par

La formule probabilité pour l accès a **l'état e1**.

1.4 Classification des propriétés temporelle [18] :

Il existe cinq grandes classes de propriété pouvant être exprimé en logiques temporelles Dans ce volet nous présentons ces différents classes de propriété.

1.4.1 Propriété de L'accessibilité :

L'accessibilité d'une certaine configuration du système, c'est-à-dire que l'état correspondant dans son modèle peut être atteint. Cette propriété peut être exprimée en utilisant l'opérateur temporel **EF**.

1.5. 2 L'invariance de propriété:

C'est-à-dire que tous les états du modèle étudié satisfont cette propriété.

1.5.3 La sûreté :

C'est-à-dire que quelque chose de mauvais pour le système ne peut pas se produire, cette propriété peut être exprimée en utilisant l'opérateur temporel G (toujours).

1.5.4 La vivacité :

Le concept de vivacité pour s'assurer qu'une action finira par avoir lieu. Ce type de propriété peut être exprimé en logique LTL en utilisant l'opérateur **F** (éventuellement) et l'opérateur **U** (jusqu'à). Cette propriété peut être également exprimée en logique CTL en utilisant l'opérateur **AF**. Exemple : AF GOAL, cette propriété permet de vérifier que le processus va atteindre fatalement son état cible GOAL.

1.5.5 L'absence de blocage

C'est-à-dire que le système ne peut pas se trouver dans une configuration à partir de laquelle il ne peut plus évoluer. Une propriété d'absence de blocage peut être exprimée par la formule CTL : AG (EX true).

1.6 Les Outils de Model Checking

Parmi les outils qui possèdent des caractéristiques ayant un pouvoir d'analyse qui permet de vérifier à la fois des propriétés temporelles et probabilistes sur des PTA (Probabilistic Timed Automata), on trouve :

1.6.1 PRISM

PRISM est un vérificateur de modèle probabiliste, il permet la modélisation et l'analyse des systèmes qui présentent un comportement aléatoire ou probabiliste formelle. Il a été utilisé pour analyser les systèmes de nombreux domaines d'application différents, y compris les protocoles de communication et multimédia, algorithmes distribués aléatoires, protocoles de sécurité, les systèmes biologiques et de nombreux autres**. [19]**

1.6.2 FORTUNA

Fortuna est le premier outil de model checking probabiliste introduisant la notion de coût. Il peut gérer la combinaison de temps réel ainsi que des caractéristiques probabilistes et de coûts. Ceci est nécessaire pour concevoir de nombreuses applications telles que les protocoles zéro-conf, Bluetooth, Fire-wire, IEEE802.11 et les protocoles pour réseaux de capteurs, ou des problèmes d'ordonnancement avec les échecs**. [20].**

1.6.3 UPPAAL [21]

L'outil Uppaal est certainement le plus connu des outils de vérification temporisés **[22].** Il est un environnement d'outils intégrés pour la modélisation, la simulation et la vérification des systèmes temps réel, développé conjointement par la recherche fondamentale en sciences informatiques à l'Université d'Alborg au Danemark et le ministère des technologies de l'information à l'Université d'Uppsala en Suède. Il convient pour les systèmes qui peuvent être modélisés comme une collection de processus non déterministes avec structure de contrôle finie.

1.6.4 SPIN [23]

SPIN (Simple Promela INterpreter) est un model checker développé par Gerard IIozmann, il permet à la fois la simulation et la vérification il est particulièrement adapté à l'étude des systèmes distribués.

SPIN repose sur le principe des automates communicants qui sont représentés par un réseau de plusieurs automates ayant la particularité de pouvoir communiquant entre eux , c' est – à- dire qu'il peuvent partager des variables ou encore s' envoyer des messages .ces automates sont particulièrement adapté à la modélisation de systèmes de contrôles ou de protocoles de communication . une fois le model représentés sous formes d'un réseau d'automates communicants il faut regrouper (ou « synchroniser «) ces automates afin de représenter le comportement global du système.

SPIN est un outil qui peut vérifier l'exactitude des logiciels et son élaboration remonte dans les environs des années 80 , avec une version gratuite .il est considéré comme l' un des

pionniers , le plus puissant pour la vérification de logiciel .SPIN a été utilisé dans plusieurs domaines .

Utilise le langage PROMELA pour décrire les modèles .PROMELA et en effet, un langage de programmation très simple, la déclaration des variables et les expressions dans PROMELA sont écrites en utilisant la syntaxe du langage C.

1.7 Conclusion

Dans ce chapitre, nous avons vu les méthodes formelles, formalismes et les outils disponibles pour la vérification .Dans le quatrième chapitre nous allons spécifier et vérifier le protocole CoAP utilisant les automates temporisés probabiliste comme un langage de spécification et le model checking sous l'outil UPPAAL.

CHAPITRE 02

L'INTERNET DES OBJETS

2-1 Introduction

L'internet des objets est simplement un concept où des machines et des objets du quotidien sont connectés via l'internet et aussi capable d'émettre de l'information et éventuellement de recevoir des commandes. L'internet des objets et ses protocoles sont parmi les sujets les mieux financés dans l'industrie et étudiés dans le milieu universitaire **[24].**

Il n'existe pas de définition standard, unifiée et partagée d'IoT, par contre les technologies IoT permettent à des choses ou des appareils fonctionnent comme des ordinateurs, d'agir intelligemment et de prendre des décisions de collaboration qui sont bénéfiques pour certaines applications. L'IoT désigne une information qui se fond dans notre quotidien pour nous simplifier la vie.

Dans ce chapitre, nous allons présenter l'internet des objets et leurs technologies et la standardisation, l'architecture, en suite nous allons parler le fonctionnement de L'IoT et enfin nous avons représenté les protocoles de L'IoT.

2-2 Définition de L'internet des objets

L'internet des objets est définit par l'union international des télécommunication comme une infrastructure mondiale pour la société de l' information , qui permet de disposer de services évolués en interconnectant des objets (physiques ou virtuels) grâce aux technologies de l' information et de la communication s'interopérables existantes ou en évolution et cela vu dire que l'internet des objets est apparut dans le cadre d'une direction issue de la mécanisation et standardisation , appliquée à l'automatisation du traitement du document et de l' information sur support matériel puis numériques comme **la figure2.1 .[25]**

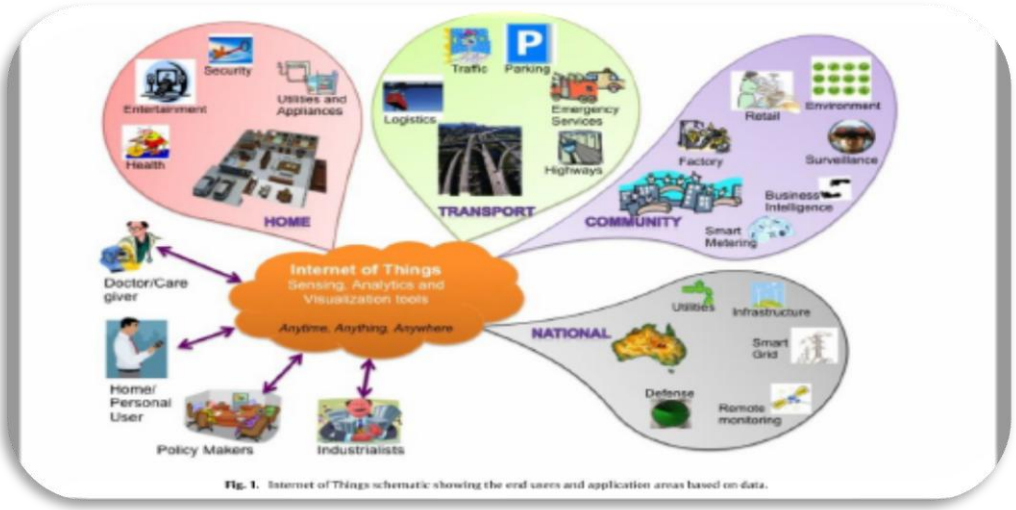


Figure 2.1 : l'internet des objets [40]

L' internet des objets c'est un réseau de réseau qui permet ,via des dispositifs d' identification électronique d'entités physiques ou virtuelles , dites objets connectés et des systèmes de communication appropriés ou virtuel , sans fil notamment ,de communiquer directement et sans ambiguïté , y compris au travers de l'internet avec ces objets connectés et ainsi de pouvoir récupérer , stocker, transférer et traiter sans discontinuités les données s'y rattachant **[26].**

La diversité des définitions proposées pour l'IoT, et leur longueur très variable montrent bien qu'il s'agit d' une notion complexe évolutive et susceptible de recevoir différentes formes d'instanciation, ces instanciations sont souvent décorées du vocables de smart ; smart Citizen , smart home , smart building, smart city , .. si le domaine d' application se trouve ainsi précisé le flou demeure sur le contour et le contenu du systèmes que l' on entend désigner .

L internet des objets apparait infinie comme un concept fonctionnel très général qui associe trois composantes fondamentales ; les objets en suite les réseaux de communication, les données **[27].**

2.2.1 Avantage de L'Internet des objets [27]

Les avantages de « loT S'étendent a tous les domaines du mode de vie et des affaires .on explique quelques avantages comme suit :

a) Amélioration de l'engagement des clients :

Les analyses actuelles souffrent les failles importantes dans l'exactitude ; il est indiqué, l'engagement demeure passif .L'IoT aide pour atteindre un engagement plus riche et plus efficace avec le public. Comme **la figure 2.2 :**

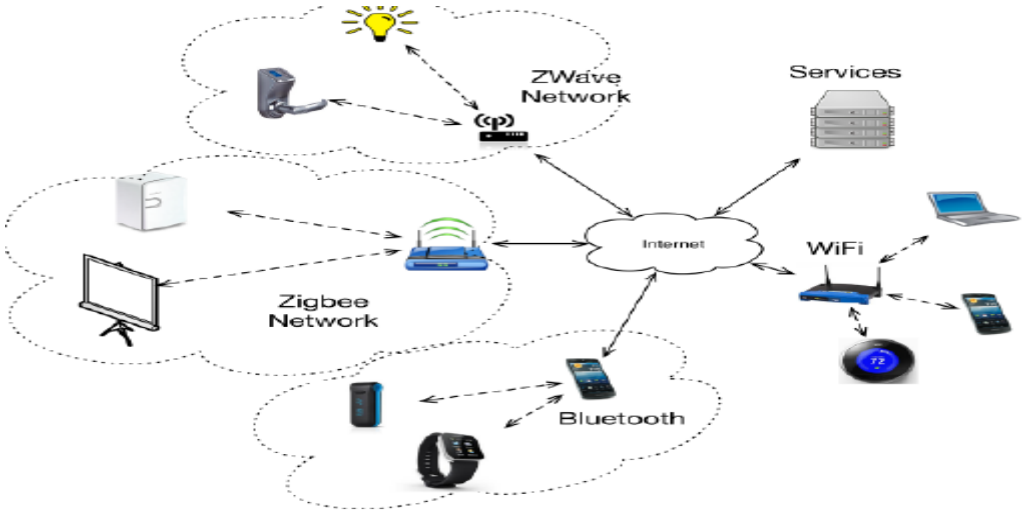


Figure 2.2 : amélioration de l'engagement client [40]

b) Optimisation de la technologie:

Les mêmes technologies et données qui améliorent l'expérience client améliorent également l'utilisation des périphériques et aident à améliorer la technologie. L'IoT déverrouille un monde de données fonctionnelles et de terrain critiques.

c) Réduction des déchets:

L'IoT rend les zones d'amélioration claires. Les analyses actuelles nous donnent un aperçu superficiel, mais l'IoT fournit des informations réelles conduisant à une gestion plus efficace des ressources.

d) Amélioration de la collecte de données:

La collecte de données moderne souffre de ses limites et de sa conception pour une utilisation passive. IoT le brise hors de ces espaces, et les place exactement où les humains veulent vraiment aller pour analyser notre monde. Il permet une image précise de tout.

2.2.2 Les défis de l'Internet des Objets(IoT) [28]

Le développement d'une application réussie IoT n'est toujours pas une tâche facile en raison de multiples défis. Ces défis comprennent: la mobilité, la fiabilité, l'évolutivité, la gestion, la disponibilité, l'interopérabilité et la sécurité et la vie privée. Dans ce qui suit, nous décrivons brièvement chacun de ces défis.

a) Mobilité: Les périphériques IoT doivent se déplacer librement et changer leur adresse IP et leurs réseaux en fonction de leur emplacement. De plus, la mobilité peut entraîner un changement de fournisseur de services qui peut ajouter une autre couche de complexité en raison de l'interruption du service et de la modification de la passerelle.

b) Fiabilité: Le système devrait fonctionner parfaitement et fournir toutes ses spécifications correctement. C'est une exigence très critique dans les applications qui nécessite des réponses d'urgence. Dans les applications IoT, le système doit être très fiable et rapide dans la collecte des données, de les communiquer et de prendre des décisions et éventuellement mauvaises décisions peuvent conduire à des scénarios désastreux.

c) Évolutivité: L'évolutivité est un autre défi des applications IoT où des millions de périphériques pourraient être connectés sur le même réseau. Gérer leur distribution n'est pas une tâche facile. En outre, les applications IoT doivent être tolérantes à de nouveaux services et périphériques qui se connectent constamment au réseau et, par

conséquent, doivent être conçus pour permettre des services et des opérations extensibles.

d) **La gestion**: La gestion de tous ces périphériques et le suivi des défaillances, des configurations et des performances de ce grand nombre de périphériques est certainement un défi dans IoT. Les fournisseurs doivent gérer les défauts, la configuration, la comptabilité, la performance et la sécurité de leurs périphériques interconnectés et tenir compte de chaque aspect.

e) **Disponibilité:** La disponibilité d'IoT inclut des niveaux de logiciel et de matériel fournis à tout moment et n'importe où pour les abonnés de service. La disponibilité du logiciel signifie que le service est fourni à toute personne autorisée à l'avoir. La disponibilité matérielle signifie que les périphériques existants sont faciles d'accès et sont compatibles avec la fonctionnalité IoT et les protocoles. En outre ces protocoles doivent être compacts pour être en mesure d'être intégrés dans les dispositifs IoT.

f) **L'interopérabilité:** L'interopérabilité signifie que les dispositifs et les protocoles hétérogènes doivent pouvoir interagir les uns avec les autres. Ceci est difficile en raison du grand nombre de plates-formes différentes utilisées dans les systèmes IoT.

2.3 Technologies et standardisation de l'IoT :

La technologie: l'Internet des Objets s'appuie sur une variété de technologies et de protocoles existants ou à définir qui, combinés ensemble ouvre la porte à de nouvelles et intéressantes perspectives. Pour illustrer les aspects technologies.

Les racines de l'IoT remontent aux technologies M2M pour le contrôle des processus à distance. L'IoT qui est aujourd'hui un mélange de plusieurs technologies telles que la RFID, les capteurs et actionneurs sans fil, le M2M, l'ultrage bande ou 3/4G, IPv6, et RPL nécessite la définition d'une architectures et des standards afin de facilité son développement dans le future. L'ETSI propose une architecture découpée en trois domaines distinct, le domaine du réseau d'objets, le domaine du réseau cœur d'accès et le domaine des applications M2M et applications clients. [29] la figure2.3 montre ca :

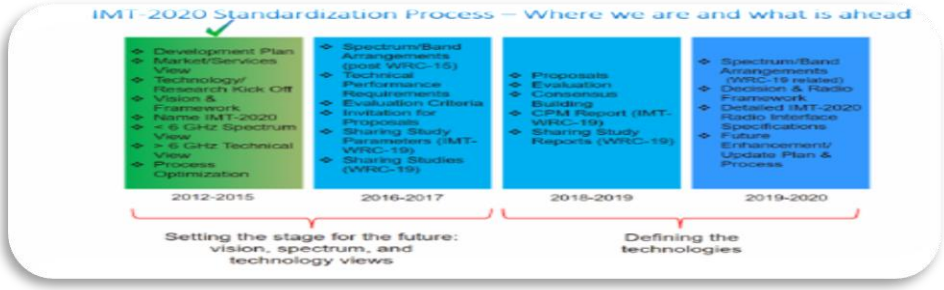


Figure 2.3 : définition technologie 2020

2.3.1 Le domaine du réseau d'objets :

Dans ce domaine nous trouvons les différentes technologies d'interconnexion des objets M2M, RFID, Bluetooth, low PAN, IETFRPL et des passerelles vers les réseaux cœur de transport**. [29].**

2.3.2. Le domaine du réseau cœur d'accès :

Dans ce domaine nous trouverons les différentes technologies de réseaux de transport et d'accès comme xDSL, WIMAX, WLAN, 3/4G, etc

2.3.3 Le domaine des applications M2M et applications clientes :

Ce domaine est composé de plateformes M2M, les Middlewares et API des applications M2M, processus métiers exploitant l'IoT, etc

- On peut présenter la technologie dans **la figure 2.4** .

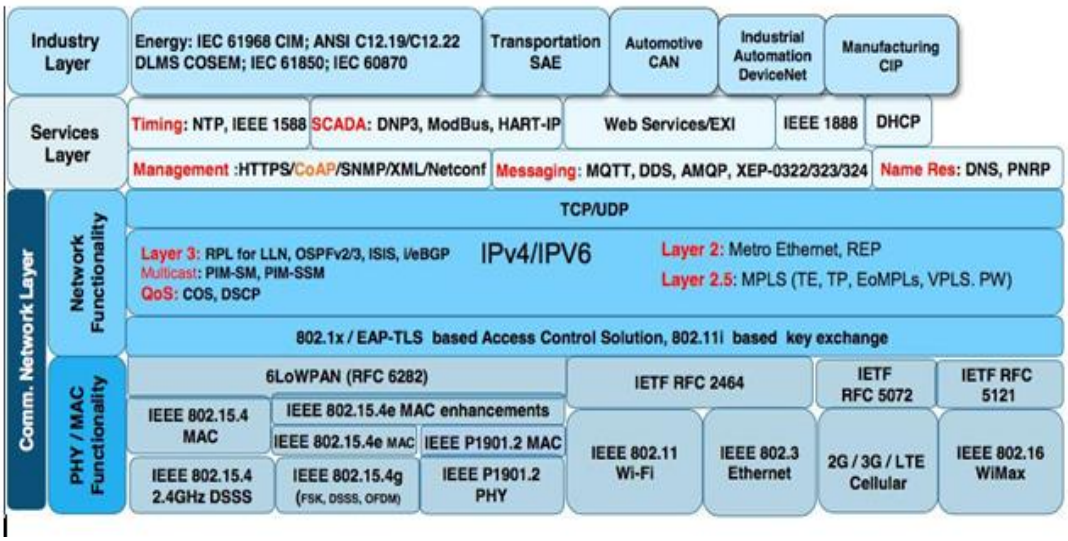


Figure 2.4: IoT d'un point de vue technique [41]

En effet, bien qu'il existe plusieurs technologies utilisées dans le fonctionnement de l'IoT, nous mettons l'accent seulement sur quelques-unes qui sont, selon Han et Zhongshan,

les technologies clés de l'IoT. Ces technologies sont les suivantes : RFID, WSN et M2M, et elles sont définies ci-dessous.

RFID : est une technologie sans fil qui est utilisée pour l'identification des objets, elle englobe toutes les technologies qui utilisent des ondes radio pour identifier automatiquement des objets ou des personnes. C'est une technologie qui permet de mémoriser et de récupérer des informations à distance grâce à une étiquette qui émet des ondes radio. Il s'agit d'une méthode utilisée pour transférer les données des étiquettes à des objets, ou pour identifier ces objets à distance. L'étiquette contient des informations stockées électroniquement être lues à distance **[34][35]** .

WSN : est un ensemble de nœuds qui communique sans fil et qui sont organiser en un réseau coopératif. Chaque nœud possède une capacité de traitement et peut contenir différent type de mémoire, un émetteur-récepteur RF et une source d'alimentation. Il peut aussi tenir compte des divers capteurs et actionneurs. Constitue un réseau de capteurs sans fil qui peut être une technologie nécessaire au Fonctionnement de l'IoT **[34].**

M2M : est l'association des technologies de l'information et de la communication avec des Objets intelligents dans le but de donner à ces derniers les moyens d'interagir sans intervention humaine avec le système d'information
d'une organisation d'une organisation ou d'une entreprise **[34][35]**

Les normes et la standardisation de l'IoT sont représentées dans **la Table 2.1** :

Emetteur	Norme/ standard	Définition
UIT	UIT-T Y.2060	Concept IoT
	UIT-T Y.2061	Interface machine-application
IEEE	IEEE 802.15.4	Couche liaison
IETF	6LoWPAN	IPv6 over Low Power Wireless Personal Area Networks
	CoAP	Constrained Application Protocol
	RPL	IPv6 Routing Protocol for Low-Power and Lossy Networks
GS1	ONS	Object Naming Service
	EPC	Electronic Product Code
OASIS	MQTT	Message Queue Telemetry Transport
	AMQP	Advanced Message Queuing Protocol
	DDS	Data Diffusion Service

Table 2.1 : les normes /standard de L'IoT.

2.4 L'architecture de L'IoT

Le schéma suivant représente L'architecture de L'IoT :

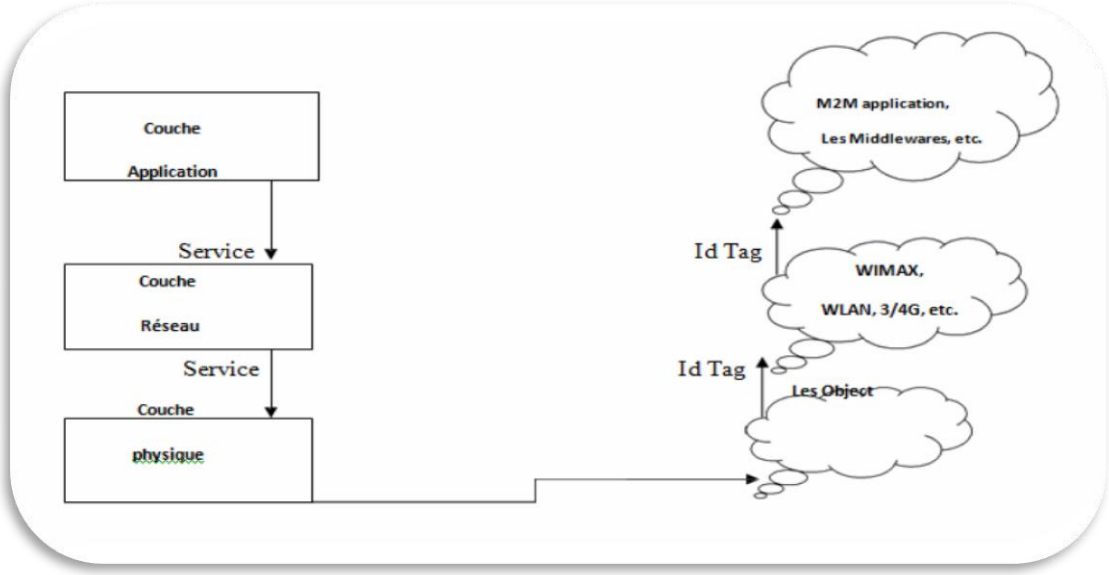


Figure 2.5 : Architecture de L'internet des Objets. [30]

2.5 La motivation de l'IoT [31]

Le matériel composée de capteurs, actionneurs et matériel de communication intégrer Par contre le middleware Stockage à la demande et outils informatiques pour l'analyse de données mais Présentation facile à comprendre la visualisation et des outils d'interprétation qui peut être largement accessibles sur différentes plates-formes et qui peuvent être conçus pour différentes application et la motivation comme **la Figure 2.6**

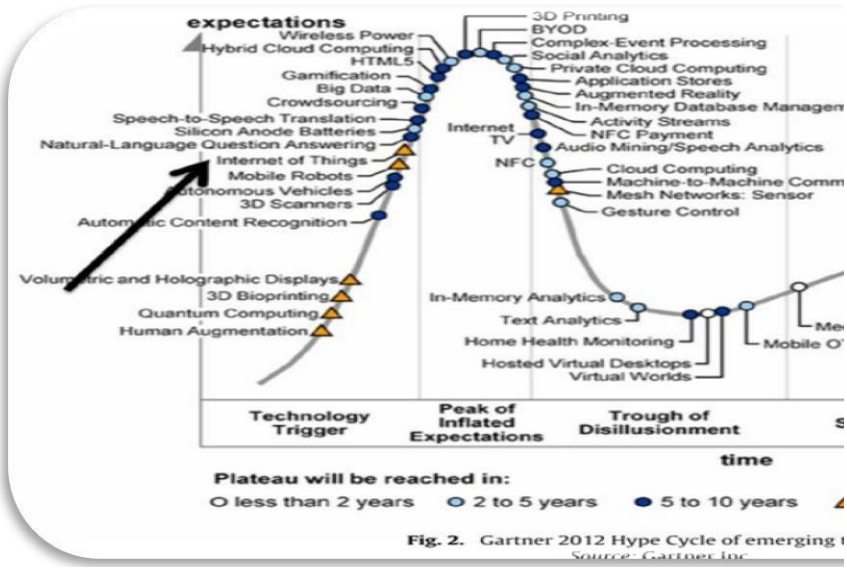


Figure2.6 : La motivation de l internet des objets

2.6 Le Fonctionnement de L'IoT [32]

Si certain objets fixes peuvent être connectés par des réseaux filaires, la croissance de l'internet des objets devrait majoritairement être portée par l'utilisation de technologies sans fil et mobiles. Les technologies de connectivité sans fil sont nombreuses et variées, et l'usage de l'une ou l'autre est souvent avant tout décidé par la portée du réseau envisagé. Certains cas d'usage nécessitent également l'association de technologies sans fil et filaire pour relier les équipements à des réseaux privés étendus ou à Internet.

Un point saillant des développements technologiques actuels autour de l'internet des objets à trait au bouleversement de la dichotomie classique dans le monde des fréquences être, d'une part des technologies à courte portée opérant en bandes libre souvent foisonnantes et déployées par l'utilisateur lui-même et, d'autre part, des technologies à grande portée opérant en bande de fréquence soumise à autorisation préalable sous licence, déployées par un nombre réduit d'opérateurs.

Le développement de réseaux à bas débit et longue portée, en bandes libres, entraine ainsi un foisonnement d'initiatives et de développement.

L'IoT permet l'interconnexion des différents objets intelligents via l'Internet. Ainsi, pour son fonctionnement, plusieurs systèmes technologiques sont nécessaires. L'IoT désigne diverses solutions techniques (RFID, TCP/IP, technologies mobiles, etc.) Qui permettent d'identifier des objets, capter, stocker, traiter, et transférer des données dans les environnements physiques, mais aussi entre des contextes physiques et des univers virtuels

Comme la Table 2.2 .

IoT Architecture layers	Components	
Application Layer	Environment, Energy, Healthcare, Transportation, People tracking, Surveillance, Supply Chain, Retail	
Management Service Layer	Device Modelling, Configuration and Management	Data flow Management, Security Control
Gateway and Network Layer	WAN (GSM, UMTS, LTE, LTE-A, 5G near future)	WiFi, Ethernet, Gateway Control
Sensors Connectivity and Network	Sensor Networks, Sensor/Actuators, Tags (RFID, Barcode)	

Table 2.2 : fonctionnement de L'IoT.

2. 7 Les protocoles de fonctionnement de L'IoT

De nombreuses normes IoT sont proposées pour faciliter et simplifier les taches des programmeurs d'applications et des fournisseurs de services. Différents groupes ont été créée pour fournir des protocoles, y compris les efforts menés par le W3C, IETF, EPC global, IEEE et l'ETSI **[32].**

L'IoT ambitionne de faire communiquer chaque système avec tous autres au moyen de protocoles communs. La mise en application à une large échelle du concept d'IoT apparait largement tributaire d'une standardisation de la communication entre objets dite M2M.

- Au niveau de la couche de liaison, le standard **IEEE 802.15.4** est le plus adapté que l'Ethernet aux environnements industriels difficiles.

- Au niveau réseau, le standard 6loWPan a réussi à adapter le protocole IPV6 aux communication sans fil entre nœud a très faible consommation.

- Au niveau routage, l'IETF a publié en 2011 le standard RPL.

- Au niveau de la couche application le protocole CoAP qui tente d'adapter http, beaucoup trop gourmand aux contraintes des communications entre nœuds à faible consommation.

2.7.1 CoAP (Constrained Application Protocol)

CoAP est un protocole de couche d'application pour les applications IoT. Il définit un protocole de transfert web basé sur les fonctionnalités HTTP, est lié à UDP (et non TCP) par défaut qui le rend plus approprié pour les applications IoT. En outre, CoAP modifie certaines fonctionnalités HTTP pour répondre aux exigences de l'IoT telles que la faible Consommation d'énergie et le fonctionnement en présence de liens à perte

CoAP a été conçu sur la base de REST (donner l'explication) qui représente un moyen plus simple d'échanger des données entre les clients et les serveurs via HTTP. REST Peut être considéré comme un protocole de connexion qui repose sur l'architecture sans serveur apatride. Il est utilisé dans les applications de réseaux Sociaux et mobiles et élimine l'ambiguïté en utilisant les méthodes HTTP get, post, put et delete. Il permet aux clients et aux serveurs d'exposer et de consommer des services Web comme le protocole d'accès aux objets simples (SOAP), mais de manière plus simple en utilisant les identificateurs de ressources uniformes (URI). CoAP vise à permettre à de minuscules appareils à faible puissance, le calcul et les capacités de communication à utiliser les interactions RESTful. Avec CoAP, Les interactions entre services web de l'Internet des PC et de l'Internet des

Objets Deviennent bien plus simples à réaliser, une passerelle applicative assez légère Correspondance entre les commandes REST et CoAP se charge de l'adaptation d'un monde à l'autre.

2.7.2 MQTT (Message Queue Telemetry Transport)

MQTT Représente un protocole De messagerie idéal pour les communications IoT et M2M. Il vise à connecter des périphériques et des réseaux intégrés aux applications et au middleware. MQTT Utilise le modèle de publication souscription pour offrir une flexibilité de transition Et une simplicité d'implémentation. Il convient aux périphériques à ressources Limitées qui utilisent des liens peu fiables ou à faible bande passante. MQTT est Construit en haut du protocole TCP. Il se compose de trois composants, abonnés, éditeurs et courtiers. De nombreuses applications utilisent MQTT telles que les Soins de santé, la surveillance, le compteur d'´energie et la notification de Facebook. Par conséquent, le protocole MQTT permet d'acheminer les périphériques de petite taille, à faible consommation et à faible mémoire dans des zones vulnérables et Réseaux à faible bande passante.

La comparaison entre les deux protocoles représentera dans **la Table 2.3**

2.7.3 XMPP (Extensible Message and Presence Protocole)

XMPP Est une Norme de messagerie instantanée IETF (IM) qui est utilisé pour les conversations Multipartis, les appels vocaux et vidéo et la télé présence. Il permet aux utilisateurs De communiquer entre eux en envoyant des messages instantanés sur Internet quel Que soit le système d'exploitation qu'ils utilisent. XMPP permet aux applications De messagerie instantanée d'accéder `a l'authentification, au contrôle d'accès, à la Mesure de la confidentialité, au cryptage hop-by-hop et à la compatibilité avec D'autres protocoles. Beaucoup de fonctionnalité XMPP en font un des protocoles Préfères par la plupart des applications de messageries instantanées et pertinentes Dans le cadre de l'IoT. Il fonctionne sur une vari´et´e de plateformes basées sur Internet de manière décentralisé. XMPP est sécurisé et permet d'ajouter de Nouvelles applications au-dessus des protocoles de base.

2.7.4 AMQP (Advanced Message Queing Protocol)

AMQP Est un Protocole de couche d'application standard ouvert pour l'IoT se concentrant sur des Environnements axés sur les messages. Il requiert un protocole de transport sécurisé.

Comme TCP pour échanger des messages. Il prend en charge une communication Fiable via des primitives de garantie de livraison de messages, en définissant un protocole Au niveau

Du fil, les implémentations AMQP peuvent inter opérer entre elles. Les communications sont traitées par deux composants principaux :

-Échanges et files d'attente de messages. Les échanges sont utilisés pour acheminer les messages vers les files d'attente appropriées.

Le routage entre les échanges et les files d'attente des messages repose sur certaines règles et conditions prédéfinies. Les messages peuvent être stockés dans les files d'attente, puis envoyés au récepteur par la suite. AMQP prend également en charge le modèle de communication publié.

2.8 Conclusion

Dans ce deuxième chapitre nous avons définis l'internet des objets (L'Iota), Nous avons cité brièvement sur les technologies et leur architecture, par la suit nous avons parlé de la motivation et son fonctionnement et quelques protocole de L'IoT

Dans le prochain chapitre nous expliquant le fonctionnement de protocole COAP.

CHAPITRE 03

LE PROTOCOL COAP POUR L' INTERNET DES OBJETS

3.1 Introduction

Les réseaux de capteurs sont des composants qui possèdent des ressources contraintes. Ils disposent notamment d'une capacité mémoire, d'une puissance, d'une bande passante et d'une réserve énergétique très limitées. Pourtant, son utilisation est de plus en plus fréquente de part son faible cout et de sa facilite de déploiement. Les réseaux de capteurs sont très polyvalents, ils peuvent être utilisés dans le cadre de la surveillance environnementale (glissements de terrain, volcans...) mais aussi dans des infrastructures intelligentes (domotique, gestion de la climatisation, lumières).

Les réseaux de capteurs vont tenir dans le futur une place dominante dans l'architecture **RESTful [37].** Le protocole CoAP est principalement destiné aux équipements et aux machines qui n'ont parfois qu'un microcontrôleur 8 bits pour tout processeur, très peu de mémoire et qui, en prime, sont connectés par des liens radio lents et peu fiables (les « **LowPAN** » ou les **LPWAN**), allant parfois à seulement quelques dizaines de kb/s**. [38].**

Dans ce chapitre, nous allons présenter la définition de la Protocol COAP pour l'internet des objets, et ces caractéristiques et ces fonctionnements, enfin nous allons représenter la segmentation de message.

3.2 Définition Le Protocole Contraintes Applicatif (CoAP)

CoAP (Constrained Application Protocol) est un protocole de transfert Web optimisé pour les périphériques et réseaux contraints utilisés dans les réseaux de capteurs sans fil pour former l'Internet des objets. Basé sur le style architectural **REST**, il permet de manipuler au travers d'un modèle d'interaction client-serveur les ressources des objets communicants et capteurs identifiées par des **URI** en s'appuyant sur l'échange de requêtes-réponses et méthodes similaires au protocole **HTTP**.

L'utilisation des services web est courante sur les applications Internet. **CoAP** étend ce paradigme à l'Internet des objets et aux applications **M2M** qui peuvent ainsi êtres développées avec des services web **RESTfu**l partagés et réutilisables. Tout en prenant en compte les contraintes et besoins de l'Internet des objets tel que le support de l'asynchrone ou du multicast. **CoAP** est prévu pour devenir un protocole d'application omniprésent dans le futur Internet des objets .

Le protocole **CoAP** se situe au niveau applicatif de la couche **OSI** et s'appuie sur **UDP** pour la communication. Il met en œuvre une méthode d'observation des ressources et fournit des fonctions de découverte des périphériques pour minimiser l'intervention humaine. Implémenté avec différents langages, ce protocole peut être utilisé dans des

domaines tels que la santé ou la gestion énergétique. Il offre des performances adaptées aux objets à faibles ressources ainsi que la sécurité pour les données sensibles

Qu'est illustré dans **la Table 3.1 :**

Protocol	Transport Protocol	Messaging	WAN (2G, 3G, 4G)	Power	Compute Resources	Security
HTTP/ REST	TCP	Rqst/Rspnse	Excellent	Fair	100Ks/RAM Flash	Low-Optimal
MQTT	TCP	Pub/Subsrb Rqst/Rspnse	Excellent	Good	10Ks/RAM Flash	Medium-Optimal
CoAP	UDP	Rqst/Rspnse	Excellent	Excellent	10Ks/RAM Flash	Medium-Optimal

Figure 3.1 : le protocole constrained applicatif CoAP

3.3 Caractéristiques du protocole CoAP

Avec l'achèvement de la spécification CoAP, il est prévu que des millions de périphériques seront déployés dans différents domaines d'application. Ces applications vont de l'énergie intelligente, du réseau intelligent, de la gestion technique des bâtiments, de la gestion intelligente de l'éclairage, des systèmes de contrôle industriels, du suivi des actifs à la surveillance de l'environnement. CoAP deviendrait le protocole standard permettant l'interaction entre les périphériques et la prise en charge de l'IoT applications **[37].** Les environnements RESTful restreints (CoRE) sont le groupe de travail dans **IEFT** qui conçoit le protocole CoAP.

Les principales caractéristiques de COAP sont les suivantes :

Un en-tête concise : Le protocole a un en-tête de base de seulement 4Ko et une taille totale de 10 a 20 Ko selon les requêtes (en ajoutant les options et les données).

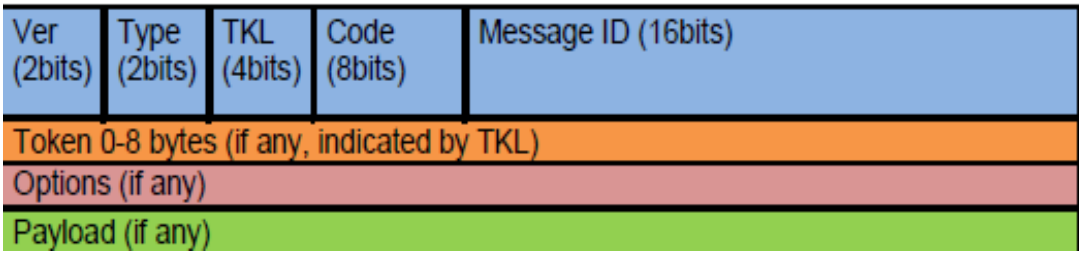


Figure 3.2 : En-tête du protocole COAP

Gestion des méthodes : COAP met a disposition les requêtes GET, PUT, POST et DELETE. Ces méthodes vont pouvoir être utilisées par le client pour accéder aux données. Leur utilisation est similaire à celle de HTTP.

Les URIs : Le protocole supporte les Uniform Resource Identifier ce qui va permettre de spécifier la cible grâce a un nom d'hôte, un port, un chemin et un paramètre de requête

Le type de contenu : Le protocole permet de transporter des données d'usages différents en mettant dans l'entête le type de données de la charge utile.

La découverte de ressources : La découverte de ressources est un besoin clé dans une architecture web et notamment pour les applications M2M, cette recherche doit être autonome et ne doit pas avoir besoin d'une interaction humaine. Les serveurs COAP doivent pouvoir fournir une URI à n que le serveur puisse être découvert.

3.4 Fonctionnement du protocole COAP

CoAP fonctionne sur UDP. Pour sécuriser les échanges, il est aussi possible d'utiliser COAP sur DTLS. Mais les messages COAP peuvent aussi être transportés sur SMS, TCP ou SCTP, CoAP fonctionne de manière asynchrone.

La taille du payload d'un message CoAP ne doit pas dépasser 1024 octets, mais un mécanisme de transfert de bloc de données CoAP permet l'envoi de différents fragments d'un même message, chaque fragment était considéré comme un message.

CoAP supporte l'envoi de messages en multicast.

3.4.1 Les Couches de La Protocole COAP

CoAP s'appuie sur une approche à **deux** couches Qu'est illustré dans, la **Figure3.2**:

Figure 3.3 : les deux couches de la protocole COAP

a- La Couche de Message : CoAP utilisée afin de traiter la non fiabilité d 'UDP ainsi que la nature asynchrone des interactions (**4 messages** sont définis **CON**, **ACK, NON, RST**) .

❖ **La sémantique** de la couche message: une requête est transportée dans un message CON (Confirmable) ou NON (Non-confirmable) et si la réponse est disponible immédiatement dans le cas d'un message CON, elle est transportée dans un message Acknowledgement (ACK). Si le message ACK est perdu, l'émetteur du message CON le retransmettra **[38].** Qu'est illustré dans **La Figure 3.3**.

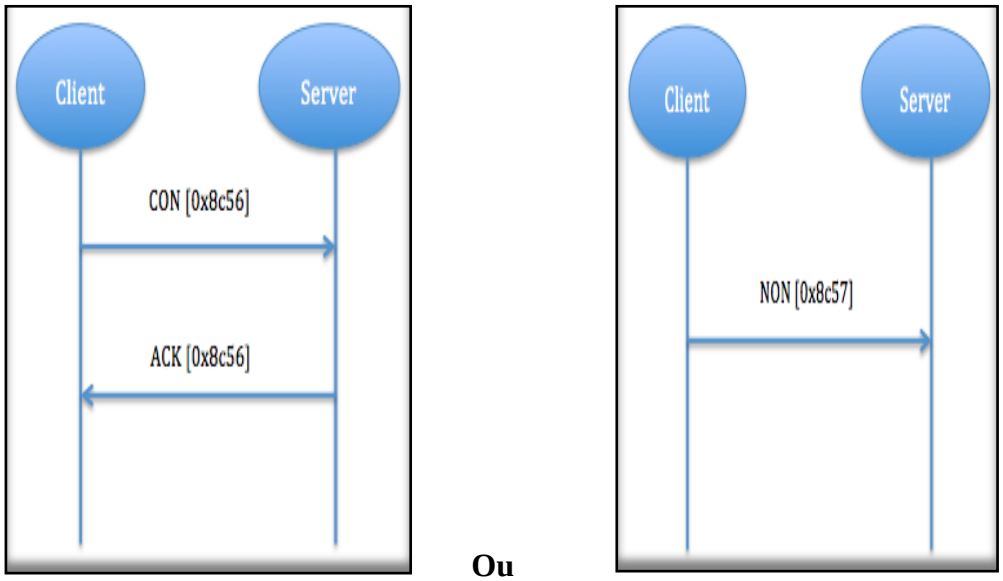


Figure 3.4 : La Sémantique de la Couche du Messagerie

b- La Couche requête et réponse (d'interaction) : sous forme de requête/réponse héritée du protocole HTTP (Requêtes **GET, POST, PUT, DELETE** et catégories de réponses).

❖ **La sémantique** des requêtes et réponses COAP est transportée dans des messages COAP qui incluent soit un code de méthode (GET, POST, DELETE, PUT) ou un code de réponse (e.g., 404, 205, etc.). Un jeton (token) est utilisé pour associer une requête à une réponse indépendamment des messages qui les transportent. Le Token est indépendant de l'identificateur de message**.** Qu'est illustré dans **La Figure 3.4**.

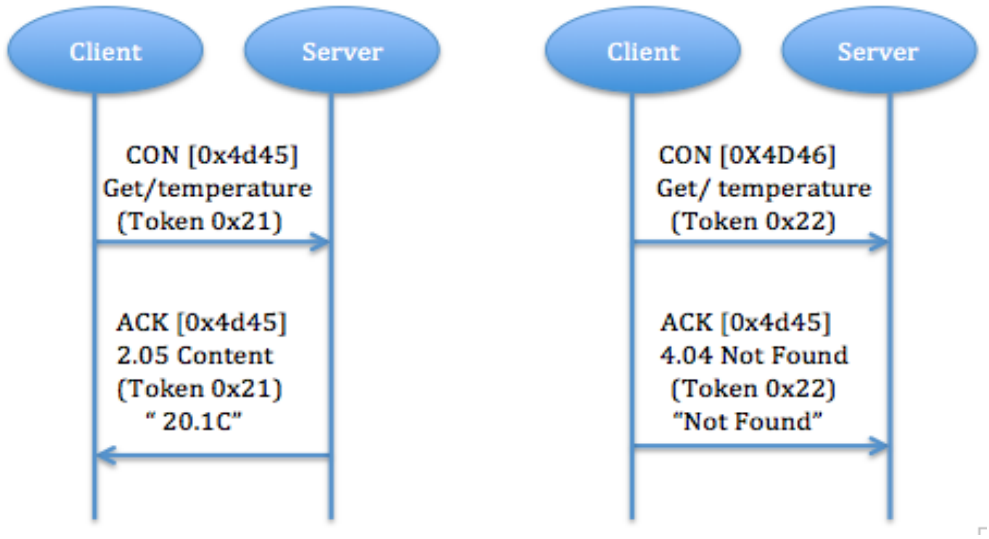


Figure 3.5: La Sémantique des requêtes et réponses

Exemple de communication COAP [37]

Pour mieux comprendre le fonctionnement de COAP, nous allons voir un exemple de communication entre un client et un serveur Qu' est illustré dans **La Figure 3.5**. .

Etape 1 : Le client envoie une requête GET avec un id = 123, de type confirmable CON et avec une uri-quer y = /light.

Le Client attend alors une réponse d'acquittement du serveur de destination. Si au terme de **temps T, le** client n 'a pas eu de réponse, il considère, que le paquet s'est perdu et retransmet le message dans un intervalle de temps aléatoire pour contrôler la congestion de réseau .Au terme de **MAX –Retransmis** retransmission, le serveur sera considéré comme inaccessible.

Etape 2 : Le serveur répond par un 2.00 OK avec le même id que le client, de type acquittement ACK et avec comme charge utile la réponse a la requête (non visible ici)

Etape 3 : Le client envoie une requête **GET** avec un id = 124, de type confirmable CON et avec une uri-query = /humidity.

Etape 4 : Le client retransmet la requête car le serveur n'a pas répondu dans les délais. Il y a eu un **timeout.**

Etape 5 : Le serveur répond par un 2.00 OK avec le même id que le client, de type acquittement ACK et avec comme charge utile la réponse a la requête (non visible ici)

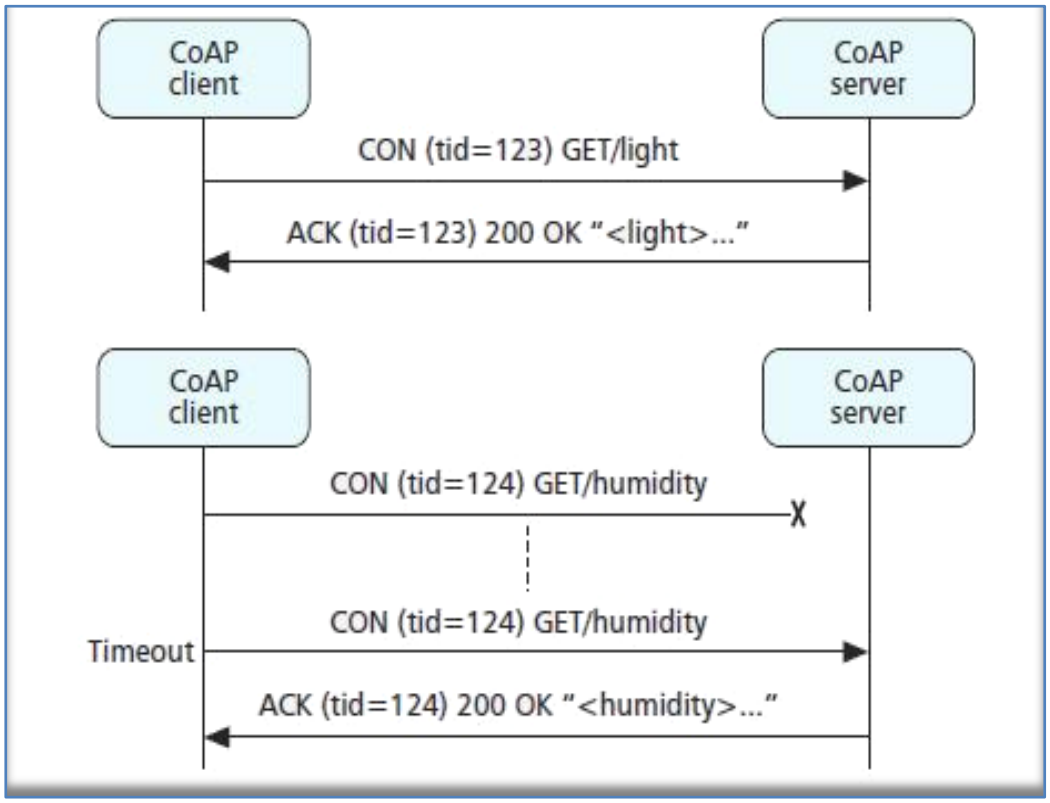


Figure 3.6 : Exemple de communication client / serveur COAP

3.5 Segmentation d'un message CoAP [38]

CoAP s'appuie sur UDO ou DTLS, ce qui limite la taille maximum des représentations des ressources qui peuvent être transférées sans fragmentation. Même si UDP supporte des tailles de segments de données pouvant atteindre 64000 octets en considérant qu'IP assurera la fragmentation, cela ne fonctionne pas réellement pour des applications contraintes comme celles relatives à M2M. Plutôt que de s'appuyer sur la fragmentation IP, CoAP propose ses propres mécanismes de fragmentation via le transfert des représentations des ressources en utilisant plusieurs blocs de requête/réponse. Cette option permet au serveur de fonctionner sans état, sans établissement de connexion et sans avoir à mémoriser les précédents transferts de bloc. L'option bloc permet donc de manière minimaliste un transfert de représentation de ressource important.

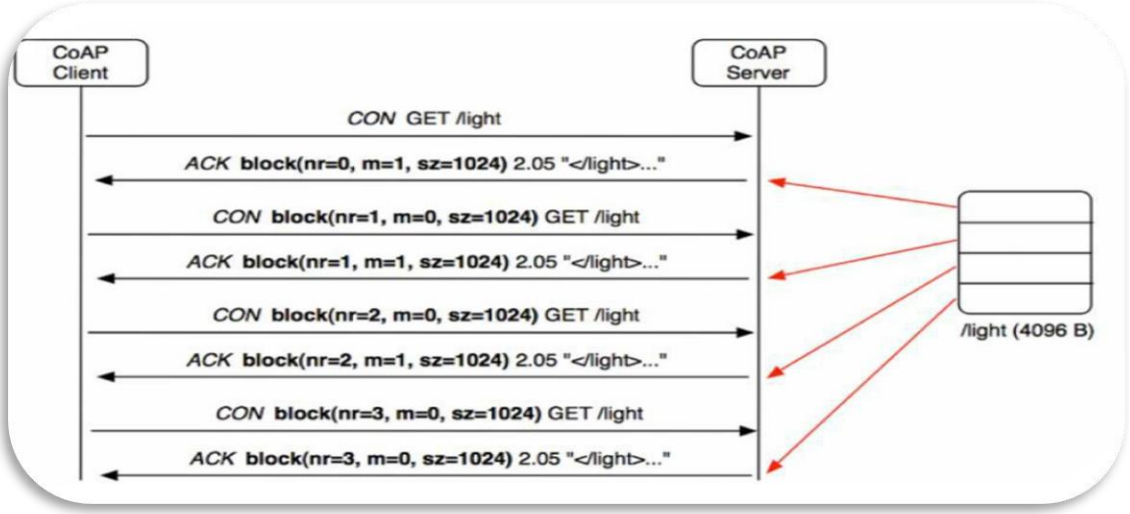


Figure 3.7 : segmentation de Message COAP

3.6 Conclusion

Dans ce chapitre nous avons vu que **COAP** est un protocole est conçu pour utiliser un minimum de ressource et par conséquent, est particulièrement adapté aux appareils et aux réseaux contrainte. Qu'il s'agit d'un protocole de transfert de document, et est donc particulièrement adapté au contexte de **l'IoT**. C'est la raison pour laquelle nous avons choisi ce protocole pour l'analyser avec le model checking pour réaliser dans la prochaine chapitre , la vérification et la modélisation du protocole capable de s'assurer des propriétés que l'on doit être en mesure d'attendre des futurs « objets » d'Internet.

CHAPITRE 04

ETUDE FORMEELE DU PROTOCOL COAP

4.1 Introduction

Après avoir présenté une description informelle du protocole dans le troisième chapitre, nous allons dans ce chapitre l'étudier formellement passant par : une modélisation formelle et une vérification de quelques propriétés du protocole en utilisant l'outil de vérification UPPAAL.

Dans ce chapitre nous allons présenter la proposition d'un modèle d' études, modèle .Ensuite, et l'outil nécessaire pour atteindre l'objectif fixé.

4.2 Proposition d'un modèle d'étude:

Pour valider un protocole, on le modélise à l'aide d'un formalisme, choisi en fonction de trois critères majeurs : le pouvoir d'expression (capacité du modèle à exprimer les caractéristiques du système à étudier), le pouvoir de modélisation (pouvoir les exprimer simplement, critère assez subjectif, qui dépend beaucoup de l'expertise de la personne qui modélise) et le pouvoir d'analyse ou de vérification (capacité à exprimer et à vérifier des propriétés sur le système)
Une fois le modèle formel construit, une exploration exhaustive de tous les chemins peut être menée par model-checking. Dans un premier temps, les contraintes que devra satisfaire le système doivent être définies formellement et seront utilisées comme propriétés au cours du model-checking. Il s'agit à la fois de contraintes de comportement et de contraintes temporelles. Parallèlement, un jeu de scénarios est créé, sur lequel les propriétés seront vérifiées. Un moteur de model-checking effectue la validation formelle de l'ensemble des propriétés sur le modèle, en s'appuyant sur le jeu de scénarios. La méthodologie synthétisée

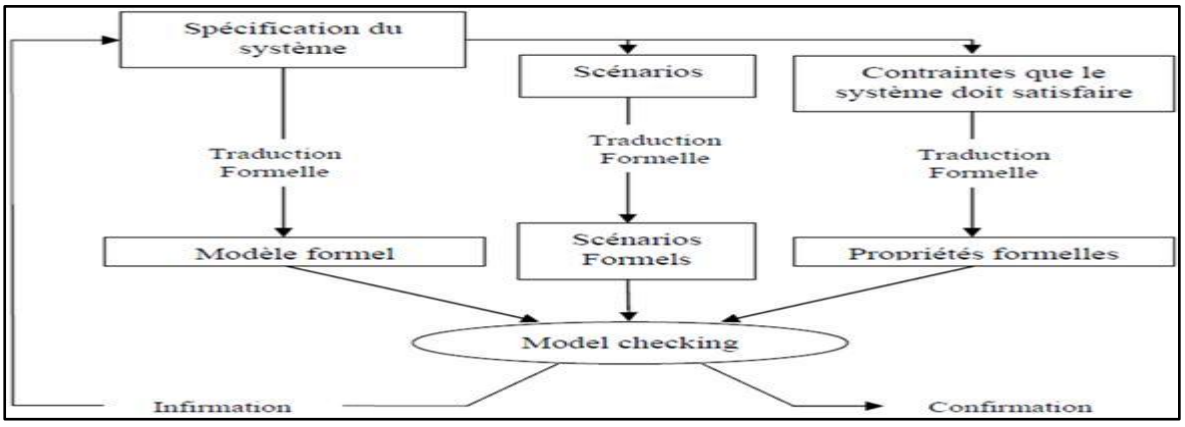


Figure 4.1: Méthodologie adoptée

Qu'est illustré dans la **Figure 4.1 [42] .**

4.3 L'outil UPPAAL

UPPAAL est un outil de modélisation et vérification de systèmes temps réel qui peuvent être représentes sous formes de processus non-déterministes et interactifs ayant une structure de contrôle finie et des horloges avec des valeurs réelles. Un modèle d'un système dans UPPAAL se compose d'un réseau de processus décrits par des automates temporisés étendus communiquant par des canaux ou des structures de données partagées.

La conception de l'outil UPPAAL est basée sur l'architecture client /serveur le serveur est représenté par la machine UPPAAL développée en C++ et le client est représenté par l'interface GUI développée en JAVA TM. La communication entre la machine UPPAAL et son interface graphique

4.3.1 Justification du choix d'UPPAAL

Nous allons travailler par l' outil UPPAAL , qui utilise les automates temporisés comme un modèle pour modéliser les systèmes temps réel .ce choix donc est motivé par la simplicité des AT (des machines états transitions) et leur extension temporelle qui facilite la modélisation du passage du temps ou les autres contraintes temporelles (qui caractérisent les systèmes temps réel) .

4.3.2 Le Model Checher UPPAAL

UPPAAL est un outil intégré pour l'environnement de modélisation , la simulation et la vérification des systèmes temps réel ,développé conjointement par la recherche fondamentale en informatique à l' université d' Aalborg au Danemark et au Département des technologies de l 'information à l' université d'Uppsala **[43]** .

Il est approprié pour les systèmes qui peuvent être modélisés comme un ensemble de processus son déterministe fini, avec la structure de contrôle et valeur réelle d'horloges, de communiquer à travers des canaux ou des variables partagées.

4.3.2.a Description

UPPAAL se compose de trois parties principales :

- **Un langage de description** : est un langage de commande surveillé non déterministe avec des types de donnée (par exemple tableaux , délimitée ….) il sert de modélisation ou de conception de langage pour d' écrire le comportement du système comme des réseaux d' automates étendus avec horloges et des données variables .

- **Le simulateur** : est un outil de validation qui permet l'examen de possibles exécutions dynamiques d'un système ou début de la conception (ou de modélisation) il ouvre ainsi un moyen peut couteux de détection des défaut avant vérification par le vérificateur de model exhaustif qui couvre le comportement dynamique de système .
- **Le modèle vérificateur** : peut l' accessibilité et l'invariabilité par l'exploration de propriété de l'état d' un système , c' est –a- dire l accessibilité de l'analyse en terme symbolique représenté par des contraintes .

Un autre élément important pour l' efficacité et l' application d' une technique qui réduit symboliquement la vérification des problèmes a celle de la manipulation efficace et résolution de contrainte .

Pour facilité la modélisation et le débogage, le vérificateur du model UPPAAL genre automatiquement une trace de diagnostique qui explique pour quoi la propriété est (ou n'est pas satisfaite par une description du système .

Le diagnostique trace génère par le model de vérification peut être chargé automatiquement ou simulateur, qu'est utilise pour la visualisation de trace .

4.3.3 Caractéristique d'UPPAAL

Un système graphique éditeur permet la description des systèmes. Une simulation graphique qui fournit des graphique de visualisation et d'enregistrement de la dynamique de comportement possible du système il est également utilisé pour visualiser les traces générées par le model de vérification.

4.3.3. a le syntaxe

Un modèle a part est basé sur les automates temporisés la description d' un model se compose de trois partie **:**

a-.les déclaration globale et local

b -.les modèles d'automates et

c- La définition du système.

Ce voulait présente la syntaxe d' UPPAAL

- **Déclarations :** les déclarations peuvent être locale ou globale **.UPPAAL** permet de déclarer des horloges (type clock , des entiers bornée type int) , des booléens (type bool) , des canaux de synchronisation (type chan) , des tableaux , des structures de donnée et de définir de type donnée (type def) .

- **Location :** Un automate temporisé est considérer comme un graphe a états .les location représentent les états dans ce graphe elles ce communiquent via les transitions .un identifiant optionnel peut être associé a chaque location afin de l'identifier, elles peuvent être étiquetés des invariant. un invariant peut être une expression composer d'un ensemble des contraintes sur des horloges, une différence entre des horloges ou des expressions booléennes ne faisant pas intervenir des horloges UPPAAL distingues trois types de locations :
- **location initiale** : Chaque automate doit avoir une location initiale représentée pas un double cercle
- ***location urgente*** : la location urgente ne permet pas de faire passer le temps quand le processus s'y trouve. sémantiquement, une location urgente est équivalente à une location dont l'invariant est x <= 0 ou x est une horloge remise a zéro sur toutes les transitions entrantes de la location en question.
- **location engagée :** la location engagée ne permet pas l'écoulement du temps quand un processus s'y trouve. Elles sont utiles pour assurer l'atomicité de la synchronisation entre trois processus ou plus .elles sont plus prioritaires que les locations urgentes. symbolique.
- **Transition** : les locations sont connectées entre elles par les transitions.ces derniers sont étiquetées par des sélections des gardes, des synchronisations et des mises à jour des variables et des foncions.
- **sélection :** cette option permet d'associer, d'une façon inde minée a une variable donnée une valeur dans un intervalle bien déterminé .les gardes, la synchronisation et les mises a jour peuvent être spécifiées en fonction de la valeur sélectionnée.
- **Grade :** Une transition est franchie si et seulement si la grade spécifiée est satisfaite .les grades expriment des contraintes d'horloges ou de données nécessaires pour franchir une transition. Formellement, les grades peuvent être spécifiées comme suit : Soient x et y sont deux horloges ou deux variables de données.
- **Synchronisation** : Les différents automates dans un modèle UPPAAL peuvent se synchroniser via des canaux de synchronisation permettent l'envoi et la réception d'un message. Deux transitions dans deux processus différents peuvent se synchroniser si et seulement si les grades des deux transitions sont satisfaits et elles admettent d'étiquettes de synchronisation **a1 ! (envoi)** et **a2 ? (réception)** ou a1 et a2 représentent le même canal de synchronisation .quand deux processus se synchronisent, les deux transitions sont franches aux mêmes temps.les mise à jour de

la transition portant l'étiquette **a1 !** se font avant celles de la transition portant l'étiquette **a1 ?** .il existe deux types de canaux de synchronisation :

a) **Les canaux de synchronisation binaires** qui permettent de synchroniser deux processus via un seul canal de synchronisation (au moyen de deux actions **a ?** et **a !**) .

b) **Les canaux de synchronisation de diffusion** (broadcast) qui permettent de synchroniser un processus avec plusieurs autre processus, une transition avec une étiquette de synchronisation **a !** avec " a " est un canal **broadcast**, peut synchroniser avec chaque transition ayant une étiquette de synchronisation **a ?** et dont les gardes sont satisfaites.

c) **Mise a jour** : l'exécution de la mise jour sur une transition permet de changer l'état du système.

Une mise a jour est une liste expressions séparées par des virgules et exécutées dans un ordre séquentiel. l'opérateur d'assignements est "=" .

Dans notre travail, nous avons utilisés le model checker UPPAAL pour vérifier les propriétés du protocole COAP. On présente l'outil d'UPPAAL dans la **Figure 4.2**.

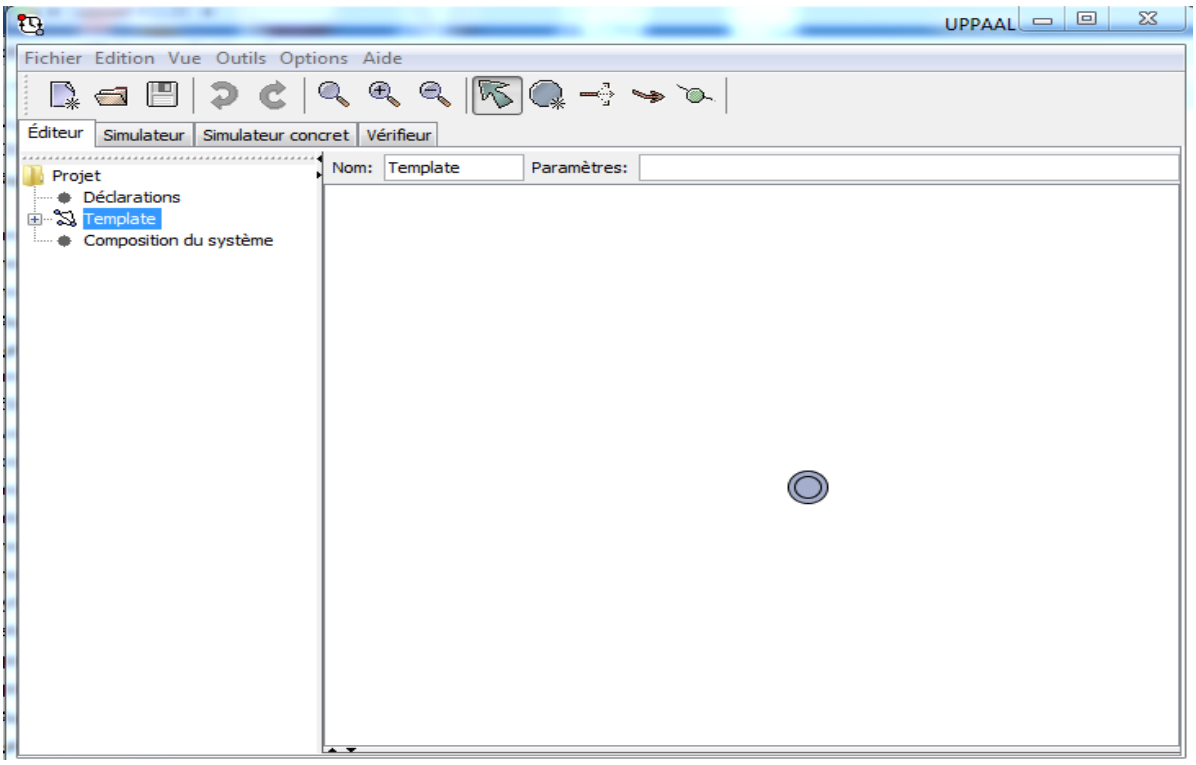


Figure 4.2 : L'outil d'UPPAAL

UPPAAL contient :

* un éditeur des modèles du protocol COAP sous forme d'automate.

* partie déclaration

* le simulateur le vérificateur

Un vérificateur pour vérifier les propriétés qui sont formulées avec logique temporelle CTL ou TCTL ou :

- **G** est présenté par **[].**
- **F** est présenté par**<>.**
- ¬ est présenté par **not.**
- ^ est présenté par **&&.**
- **v** est présenté par **| |.**

Et les autres opérateurs restent les mêmes**.**

4.4 Modélisation du protocole COAP

Notre système composé de deux composant : le première Template COAP client et la deuxième COAP server

Qu'est illustré dans la **Figure 4.3**

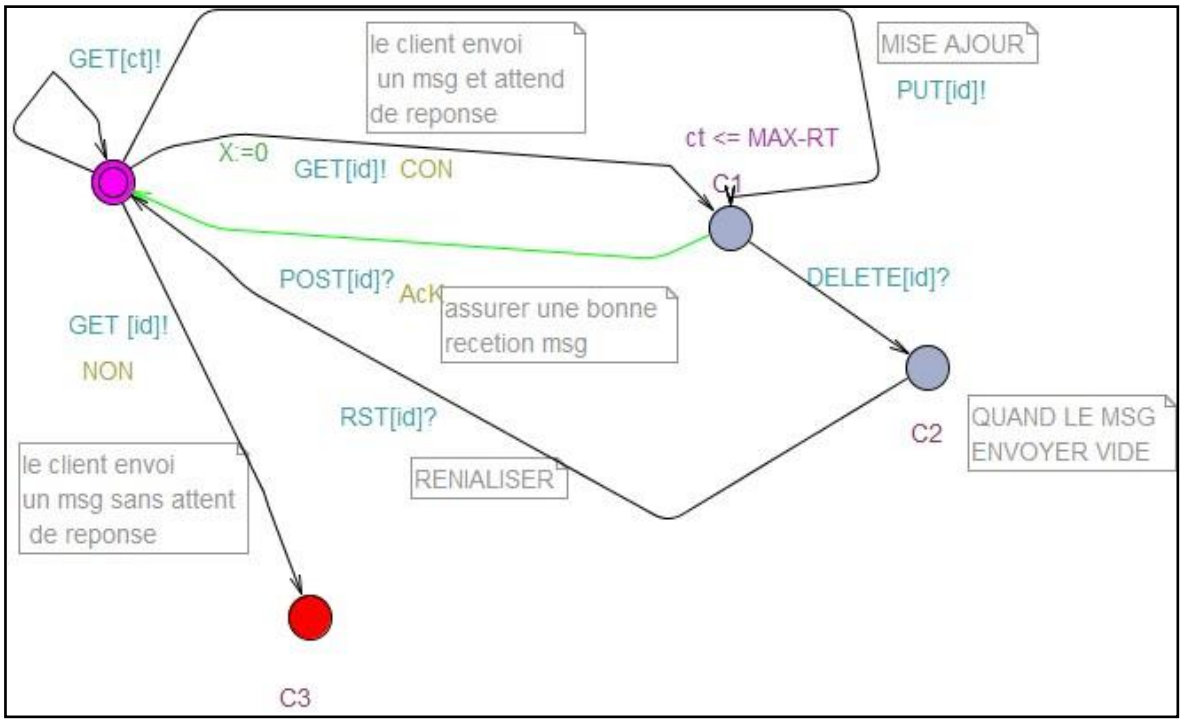


Figure 4.3 : modèle COAP client

En suit, le modèle qui représente COAP server. **La figure 4.4**

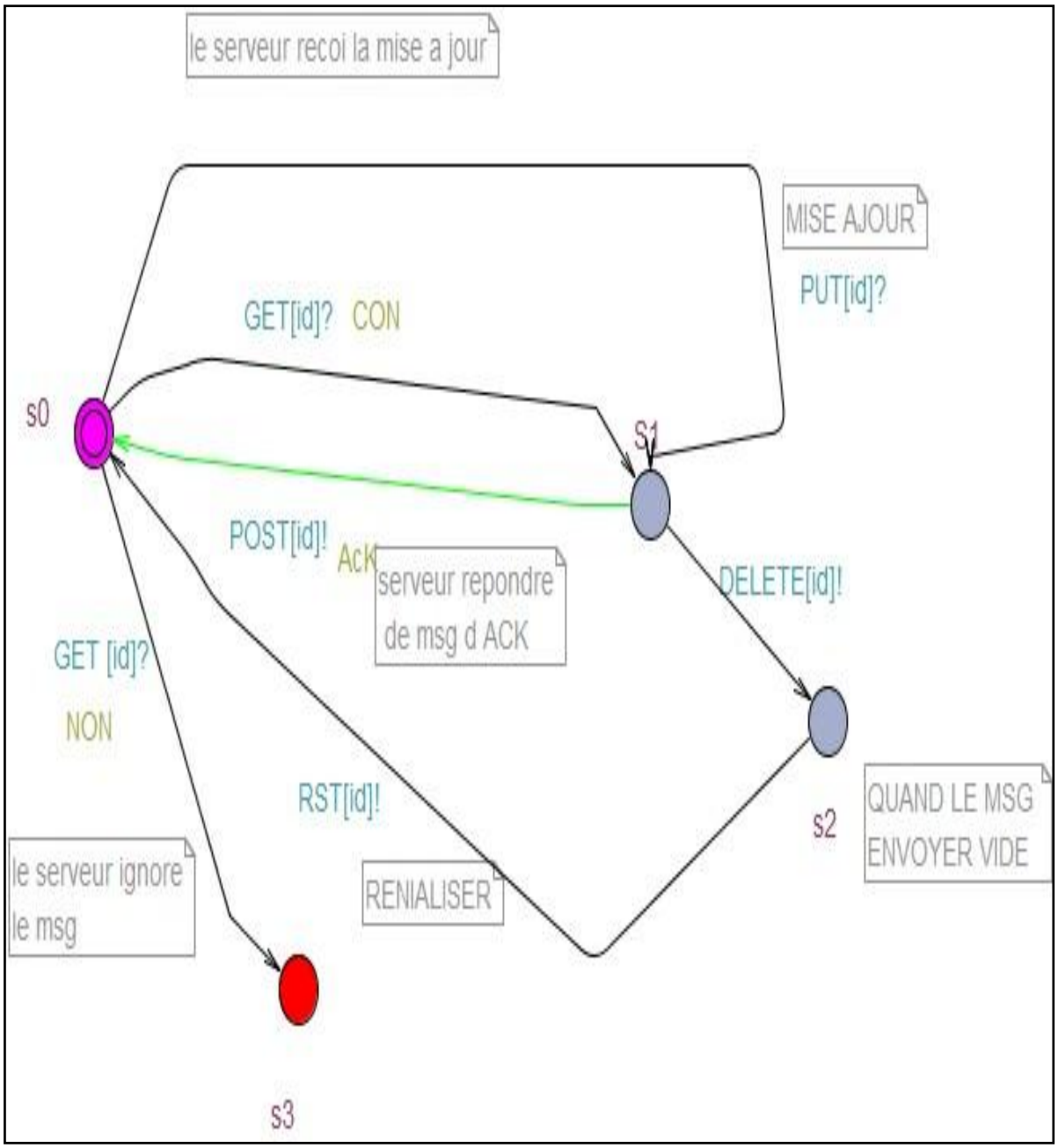


Figure 4.4 : modèle COAP server

4.5 Vérification du protocole COAP

L'objectif principal de l'usage UPPAAL c'est de vérifier certaines propriétés dans le système spécifié .Nous avons établi les propriétés qu'on va essayer de les vérifier.

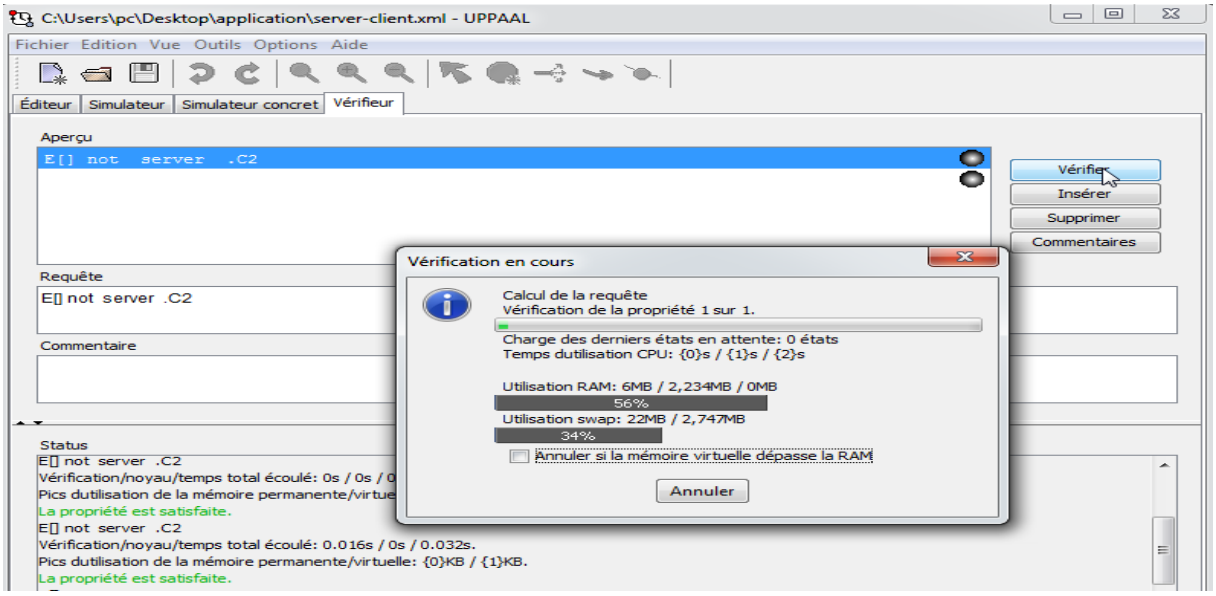


Figure 4. 5 : Exemple de vérification

4.6 Conclusion

Ce chapitre a été consacré pour présenter notre travail .Nous réussi à modéliser l'émetteur COAP client ainsi que le récepteur COAP server en utilisant les automates temporisés étendus (ceux utilisés par UPPAAL). Cette modélisation prend en charge les fonctionnalités de l émetteur et le récepteur. Ces fonctionnalités se résument dans l'envoi et de la réception des messages du protocole COAP, cette modélisation nous aidé à mieux comprendre le fonctionnement du protocole COAP.

Nous avons aussi vérifié quelques propriétés de ce protocole en utilisant le model checker de l'outil UPPAAL.

CONCLUSION GENERALE

Conclusion générale

L'internet des objets est un concept qui repose sur l'idée que tous les objets seront connectés un jour à l'internet de l'information et éventuellement de recevoir des commandes .En quelques années seulement depuis son apparition, il est fut adopté dans divers secteurs et cela à grâce à son potentiel énorme. Cependant, sa forte intégration soulève plusieurs interrogations dont le principal est « comment assurer la privance et instaurer une sécurité robuste pour cette nouvelle technologie fortement hétérogène et ubiquitaire.

La communication constante entre les objets se fait par de multiples protocoles, y compris le protocole COAP.

Ce travail rentre dans le cadre de la spécification et la vérification formelle .ces deux derniers sont très importantes pour garantir la fiabilité de plusieurs types de systèmes .Ces système sont souvent des systèmes ou leur dysfonctionnement engendre des catastrophes .Pour cette raison , nous avons fait appel à ces méthodes formelles pour modéliser et vérifier un protocole d'application COAP , qu'il Son fonctionnement similaire du protocole http **.**Nous ramenés à l'utilisation des méthodes formelles pour garantir un bon fonctionnement de ces systèmes importants.

Ce travail nous a aidés à :

- Comprendre le fonctionnement général du protocole COAP.
- Acquérir des connaissances plus au moins avancés sur la modélisation par l'outil UPPAAL.
- Mieux comprendre le fonctionnement du protocole COAP.
- Vérifier des propriétés attendues de ce protocole.

Dans le futur, nous envisageons de faire :

- Vérifier d'autres protocoles avec UPPAAL.
- Vérifier ce protocole avec un autre outil.

BIBLIOGRAPHIE

Bibliographie

[1] Alexandre Mouradian, Proposition et v´erifi´ecation formelle de protocoles de communications temps-r´eel pour les r´eseaux de capteurs sans fil, L'Institut National des Sciences Appliqu´ees de Lyon, Th`ese de Doctorat.

[2] Bertot and P. Cast´eran, Interactive theorem proving and program development–coq' art: the calculus of inductive constructions, 2004.

[3] Alur, R., C. Courcoubetis, T.A. Henzinger et P.H. Ho, Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems, hybrid systems.

[4] Guthmuller, MarionModel checking ET verification de proprieties de vivacité dans SimGrid.

[5] Edmund MClARKE, William,KLEBER,MillosNOVAKpaolo ZulAN" Modelchecking and the state explosion "in Tools for Tools for pratical software Verification.page1.30 spring 2013.

[6] D. Johnson, C. Perkins, J. Arkko. "Mobility support in ipv6", IETF RFC3775, 2004

[7] Rajeev Alur et D.L. Dill, Automata for modeling real-time systems,in proceedings of the seventeenth international colloquium on automata, languages and programming, New York, NY, USA, 1990.

[8] Rajeev Alur et D.L. Dill, Automata for modeling real-time systems, in proceedings of the seventeenth international colloquium on automata, languages and programming, New York, NY, USA, 1990.

[9] Robert ABO, Approches formelles pour l'analyse de la performabilit´e des syst`emes commu- nicants mobiles applications aux r´eseaux de capteurs sans fil, Th`ese docteur, 06 d´ecembre 2011.

[10] P, Fournier." Vérification paramétré de protocole sur les réseaux de capteur sans fil " Mémoires de Master. Université de Rrennes1.

[11] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real- time systems with discrete probability distributions. Theoretical Computer Science, 286, 2002.

[12] MAJDA MOUSSA. "V´erification et configuration automatiques de pare-feux par model checking et synth`ese de contrˆoleur", Université De Montréal,Mémoire De Maitrisées Sciences Appliquées, 2014, pp 14-18.

[13] VARDI, M.Y. "An automata-theoretic approach to linear temporal logic", Logics for con- currency, Springer, 1996, pp 238–266.

[14] ROZIER, K.Y. ”Linear temporal logic symbolic model checking”, Computer Science Re- view, 2011, pp 163–203.

[15] KUGLER, H., HAREL, D., PNUELI, A., LU, Y.et BONTEMPS, Y. ”Temporal logic for scenario-based sp´ecifications, Tools and Algorithms for the Construction and Analysis of Systems”, 2005, Springer, pp 445–460.

[16] Franck Cassez. ”V´erification qualitative”, Ecole d’´et´e Temps-R´eel, 2003, pp 2–13

[17] H. Hansson et B. Jonsson. ”A logic for reasoning about time and reliability”. Formal As- pects of Computing , 1994, pp 512–535.

[18] MAJDA MOUSSA. Vérification et configuration automatiques de pare- feux par model checking et synthèse de contrôleur, Université De Montréal, Mémoire De Maitrisées Sciences Appliquées, 2014, pp 14-18.

[19] Jos C. M. Baeten: A brief history of process algebra. Journal of Theoretical, Computer Science, 335(2-3):131_146, 2005.

[20] Christel Baier and Joost-Pieter Katoen, Principles of model checking, the mit press, ambridge, Massachusetts London, England, 2008, pp 673-880.

[21] Le model-checker UPPAAL est accessible du site: http://www.uppaal.org/

[22] Kim Guldstrand Larsen, Paul Pettersson etWang Yi : Uppaal in a nutshell. International Journal on Software Tools for Technology Transfer, 1(1-2):134_152, 1997.

[23] M,Gueffaz.”model checking et websémantique”.These de doctorat.Université de Bourgone,2012.

[24] "Gartner's 2014 hype cycle for emerging technologies maps the journey to digital business", August 2014, http://www.gartner.com/newsroom/id/2819918 .

[25] (books.openedition.org\editionsmsh\8u?lang=Fr#tocFrom2n1.)

[26] Consulté le: http://fr.slideshare.net/mobile/mousski/9-linternet-des-objets.

[27] Internet of Things tutorialspoint, simply easy learning, 2016

[28] Tara Salman (A paper written under the guidance of Prof. Raj Jain), Networking Protocols and Standards for Internet of Things , p 20

[29] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things : A survey on enabling technologies, protocols, and applications.
IEEE Communications Surveys & Tutorials, 17(4) :2347–2376, 2015.

[30] Y.ait mouhoub, F.Bouchebbah . Propotion d'un modèle de confaince pour l'internet des Objets, Université A/MIRA de Bejaia . 2015

[31] GROUP DE L IOT SALAH PHILIPSE,'' internet of things : une visions des elements , architecturaux , des oriontation futur

[32] Consulté le: https://www.itu.int/rec/T-RREC-E.164-201011-1/fr

[33] Sécurité des objets connectés - Travaux des auditeurs, p 16,URL : http://images.cigref.fr/Publication/2014-INHESJ-CIGREF-securite_objets_connectes.pdf

[34] ACHOUR, N.Makhloufi, Authentification dans l'internet des objets . Université A/MIRAde Bejaia, 2017.

[35] R. Saad. Modèle collaboratif pour l'Internet of Things (IoT). PhD thesis, Université du Québec a Chicoutimi, 2016.

[36] S. Feng, J. Cerles, H. Dalmas, T. Dˆo-Khac, and B. Paulin. Sécurité des objets Connectés. Institut national des hautes études de la sécurit´e et de la justice,2014

[37] VIPRET Julien Laboratoire LIG - Grenoble INP – Ensimag julien.vipret at ensimag.fr les reseaux de capteurs Rapport d'IRL Utilisation du protocole COAP pour la decouverte de ressources dans les reseaux de capteurs .

[38] EFORT http://www.efort.com COAP (Constrained Application Protocol) Protocole d'Application pour l'Internet des Objets .

[39] S. Keoh , Recherche Philips, Z. Shelby , Sensinode . Profilage de DTLS pour les applications IoT basées sur CoAP draft-keoh-dice-dtls-profile-iot-00 [2013-11-05] http://tools.ietf.org/html/draft-keoh-dice-dtls-profile -iot-00

[40] SIMON MENSEH '' internet des chose un examen sur les protocoles de passrelle de connectivité sémantiques interepebilité

[41] Internet Of Things: Et d'un point de vue technique,URL: http://www.iplogos.fr/wp-content/uploads/2015/08/IoT-Stack.png

[42] Thomas Watteyne, Proposition et validation formelle d'un protocole MAC temps réel pour réseaux de capteurs linéaires sans fils, Mémoire de Master Recherche Informatique-Spécialité Réseaux, Télécommunications et Services-l'INSA de Lyon,2004/2005

[43] Mr Kaid Omar Ilyes ''Modelisation d' un controleur de temperature d'un réacteur d' avion par UPPAAL '' Mémoire fin étude licence en informatique de l université d'Oran 2008-2009 .

[44] Faiz CHARFI ;' UNE approched interfacage de CoD a Uppaal pour la spécification et la vérification des systèmes temps réel', MEMOIRE DE D.E.A en informatique d'UNIVERSITE DE TUNIS EL MANAR .Septembre 2003

[45] Majda Moussa, Vérification et configuration automatique de pare-feux par model checking et sythése. université Abou Bakr belkaid –Tlemcen,2012

Résumé :

Au cours des dernières années, nous avons assisté à grand augmentation d'utilisation d'IoT et un nombre croissant de capteurs intégrés dans des appareils intelligents (par exemple, téléphones mobiles, montres intelligentes ou lunettes intelligentes). Ces objets intelligents sont de plus en plus connectés à l'internet et les données sont envoyées au web pour créer des applications 'Internet of Things' (IoT).avec l'utilisation des protocoles et différents technologie. Ce dernier nécessite non seulement le développement d'infrastructure et des logiciels d'ingénierie, mais aussi la conception et le déploiement de nouveaux services capables de supporter des applications multiples, évolutives et interopérables (multi-domaines). De toutes les difficultés posées par l'Internet des objets, l'interconnexion des objets et leur interopérabilité demeurent les plus importantes. En effet, conformément aux usages pour lesquels ces objets ont été conçus, l'échange et la sémantique d'information provenus d'une une telle interaction varie significativement contribuant à faire de l'Internet des objets un monde riche mais aussi hétérogène. On s'inscrit ici dans une évolution de l'Internet des Objets dans 3 directions complémentaires :

1- L'abstraction des dispositifs utilisés comme intermédiaires pour offrir une présence en réseau à des objets.

2- Le passage de l'Internet des Objets au « Web des Objets »

3- La représentation des informations correspondantes à un niveau sémantique qui permet de la partager et de la rendre interopérable indépendamment des infrastructures sous-jacentes.

ملخص :

في السنوات الأخيرة، شهدنا زيادة كبيرة في استخدام إنترنت الأشياء وعدد متزايد من أجهزة الاستشعار المدمجة في الأجهزة الذكية (على سبيل المثال، الهواتف المحمولة، والساعات الذكية أو النظارات الذكية). يتم توصيل هذه الكائنات الذكية بشكل متزايد بالانترنت ويتم إرسال البيانات إلى الشبكة لإنشاء تطبيقات "إنترنت الأشياء .

يتطلب -هذا الأخير ليس فقط تطوير البنية التحتية والبرمجيات الهندسية- بل أيضا تصميم ونشر خدمات جديدة قادرة على دعم التطبيقات المتعددة والقابلة للتحجيم والقابلة للتشغيل البيني (متعدد المجالات)، ومن بين جميع التحديات التي تطرحها شبكة إنترنت الأشياء، يظل الربط البيني بين الأشياء وقابليتها للتشغيل البيني هو الأكثر أهمية في الواقع.

ويكون تطور إنترنت الأشياء في ثلاثة اتجاهات تكميلية :

1-تسخير الأجهزة المستخدمة كأجهزة وسيطة لتوفير وجود شبكي للأجهزة.

2-الانتقال من إنترنت الأشياء إلى "شبكة الأشياء ".

3-تمثيل المعلومات المقابلة على مستوى دلالي يسمح بتقاسمها وجعلها قابلة للتشغيل البيني بغض النظر عن البنى التحتية الأساسية.