Escola Tècnica Superior d'Enginyeria
Electrònica i Informàtica La Salle

Treball Final de Màster

Màster Universitari en Enginyeria de Telecomunicació

Estudi dels paradigmes de ICN i DTN en entorns de IoT

Alumne
Andrea López Santos

Professor Ponent
Rosa Maria Alsina Pagès

Professor Co-Ponent
Ramon Martín de Pozuelo

ACTA DE L'EXAMEN
DEL TREBALL FI DE CARRERA

Reunit el Tribunal qualificador en el dia de la data, l'alumne

D. Andrea López Santos

va exposar el seu Treball de Fi de Carrera, el qual va tractar sobre el tema següent:

Estudi dels paradigmes de ICN i DTN en entorns de IoT

Acabada l'exposició i contestades per part de l'alumne les objeccions formulades pels Srs. membres del tribunal, aquest valorà l'esmentat Treball amb la qualificació de

Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENT DEL TRIBUNAL

Abstract

La tecnologia de Internet of Things (IoT) promet connectar milers de milions d'objectes a Internet. Avui dia, el paisatge de IoT està molt fragmentat tant des de la perspectiva de la tecnologia com del servei. Com a conseqüència, és difícil integrar i correlacionar les dades provinents de contextos heterogenis i generar serveis de valor agregat en la part superior. Aquesta raó ha motivat la tendència actual de desenvolupar una arquitectura de IoT desfragmentada unificada perquè els objectes puguin ser accessibles a les aplicacions en organitzacions i dominis. Per abordar aquest problema, s'ha proposat una arquitectura de IoT unificada, basada en l'arquitectura de la Information Centric Networking (ICN).

Un altre dels reptes del IoT és poder fer front a l'alt nivell de dinamisme de la xarxa, d'aquesta manera una de les propostes per poder administrar la connectivitat intermitent en IoT és mitjançant una comunicació tolerant al retard, amb el model l'arquitectura de xarxa Delay Tolerant Networking (DTN).

El present document presenta un repte tecnològic de recol·lecció d'informació mitjançant un sistema IoT, que permeti actuar de forma distribuïda i no depenent d'un node central, utilitzant l'aplicació d'arquitectures de xarxa ICN i DTN en aquest entorn IoT, per l'intercanvi d'informació entre els diferents objectes de la xarxa. Aquest sistema es planteja com a alternativa allà on la cobertura dels canals habituals és limitada.

L'estudi ha escollit com a escenari realitzar una control intel·ligent de la temperatura d'un edifici, de manera que es pugui accionar remotament el termòstat. S'ha treballat amb una xarxa mallada de sensors de temperatura, la qual està interconnectada per poder orquestrar de manera autònoma els diferents punts, sense necessitat d'una gestió centralitzada des del centre de control.

Abstract

La tecnología de Internet of Things (IoT) promete conectar miles de millones de objetos a Internet. Hoy en día, el paisaje de IoT está muy fragmentado tanto desde la perspectiva de la tecnología como del servicio. Como consecuencia, es difícil integrar y correlacionar los datos que provienen de contextos heterogéneos y generar servicios de valor agregado en la parte superior. Esta razón ha motivado la tendencia actual de desarrollar una arquitectura de IoT desfragmentada unificada para que los objetos puedan ser accesibles a las aplicaciones en organizaciones y dominios. Para abordar este problema, se ha propuesto una arquitectura de IoT unificada, basada en la arquitectura de la Information Centric Networking (ICN).

Otro de los retos del IoT es poder hacer frente al alto nivel de dinamismo de la red, de este modo una de las propuestas para poder administrar la conectividad intermitente en IoT es mediante una comunicación tolerante al retraso, con el modelo la arquitectura de red Delay Tolerant Networking (DTN).

El presente documento presenta un reto tecnológico de recolección de información mediante un sistema IoT, que permita actuar de forma distribuida y no dependiente de un nodo central, utilizando la aplicación de arquitecturas de red ICN y DTN en este entorno IoT, para el intercambio de información entre los diferentes objetos de la red. Este sistema se plantea como alternativa allá donde la cobertura de los canales habituales es limitada.

Para el estudio se ha escogido como escenario realizar el control inteligente de la temperatura de un edificio, de manera que se pueda accionar remotamente el termostato. Se ha trabajado con una red mallada de sensores de temperatura, la cual está interconectada para poder orquestar de manera autónoma los diferentes puntos, sin necesidad de una gestión centralizada desde el centro de control.

Abstract

The Internet of Things (IoT) technology promises to connect billions of objects to the Internet. Nowadays, the IoT landscape is very fragmented from both the technology and service perspectives. Therefore, it is difficult to integrate and correlate the data that comes from heterogeneous contexts and generate value-added services at the top. This reason has motivated the current trend of developing a unified defragmented IoT architecture so that objects can be accessible to applications in organizations and domains. To address this problem, it has been proposed a unified IoT architecture, based on the Information Centric Networking (ICN) architecture.

Another of the challenges of the IoT is to be able to face the high level of dynamism of the network, in this way one of the proposals to be able to manage the intermittent connectivity in IoT is through a tolerant communication to the delay, with the network architecture model Delay Tolerant Networking (DTN).

This document presents a technological challenge of gathering information through an IoT system, which allows to act in a distributed way and not dependent on a central node, using the application of network architectures ICN and DTN in this IoT environment, for the exchange of information between the different objects of the network. This system is considered as an alternative where the coverage of the usual channels is limited.

For the study, it has been chosen as a scenario to perform the intelligent control of the temperature of a building, so that the thermostat can be operated remotely. We have worked with a meshed network of temperature sensors, which is interconnected to be able to independently orchestrate the different points, without the need for centralized management from the control centre.

Resum

La tecnologia de Internet of Things (IoT) promet connectar milers de milions d'objectes a Internet. Avui dia, el paisatge de IoT està molt fragmentat tant des de la perspectiva de la tecnologia com del servei. Com a conseqüència, és difícil integrar i correlacionar les dades provinents de contextos heterogenis i generar serveis de valor agregat en la part superior. Aquesta raó ha motivat la tendència actual de desenvolupar una arquitectura de IoT desfragmentada unificada perquè els objectes puguin ser accessibles a les aplicacions en organitzacions i dominis. Per abordar aquest problema, s'ha proposat una arquitectura de IoT unificada basada en l'arquitectura de la Information Centric Networking (ICN).

Un altre dels reptes per al IoT és poder fer front a l'alt nivell de dinamisme de la xarxa, es necessita poder manejar la connectivitat intermitent en IoT. Una de les propostes per solucionar-ho és considerar una comunicació tolerant al retard, mitjançant un model d'arquitectura de xarxa Delay Tolerant Networking (DTN), la qual permetrà que els objectes intel·ligents es comuniquin millor fins i tot amb la presència d'interrupcions en la seva connectivitat.

El projecte està enfocat com un repte tecnològic de recol·lecció d'informació mitjançant un sistema IoT, utilitzant l'aplicació d'arquitectures de xarxa ICN i DTN en aquest entorn IoT, per l'intercanvi d'informació entre els diferents objectes de la xarxa. L'objectiu és l'estudi del paradigma ICN com a capa d'adreçament, enrutament i servei (com a cerca semàntica d'informació) i DTN com a extensió de la xarxa IoT en el cas de xarxes de curt abast.

Aquest document planteja aquest sistema com a alternativa allà on la cobertura dels canals habituals és limitada. L'estudi ha escollit com a escenari realitzar una control intel·ligent de la temperatura d'un edifici, de manera que es pugui accionar remotament el termòstat. S'ha treballat amb una xarxa mallada de sensors de temperatura, la qual està interconnectada per poder orquestrar de manera autònoma els diferents punts, sense necessitat d'una gestió centralitzada des del centre de control. S'ha realitzat un disseny d'aquesta xarxa i s'ha implementat en unes motes Z1 de Zolertia emulant una comunicació del diversos punts de l'edifici.

El projecte parteix també del requeriment d'un sistema d'emmagatzematge distribuït, de l'estil "Fog/Edge Computing/Storage", que doni suport al desplegament del sistema, però el disseny i desplegament d'aquest queda fora de l'abast d'aquest projecte.

Índex

Acrònims

AM	*Authentication Manager*
AWS	*Amazon Web Services*
DAG	*Directed Acyclic Graph*
DAO	*Destination Advertisement Object*
DDoS	*Distributed Denial-of-Service Attack*
DID	*Delay-Tolerant ICN for Disaster-Management*
DIO	*DODAG Information Solicitation*
DIS	DAG *Information Object*
DODAG	*Destination Oriented Directed Acyclic Graph*
DTN	*Delay Tolerant Networking*
ES	*Embedded System*
HTTP	*Hypertext Transfer Protocol*
IaaS	*Infrastructure as a Service*
ICN	*Information Centric Networking*
IoT	*Internet of Things*
LAMPP	*Linux Apache MySQL/MariaDB Perl/PHP/Python*

LCD	*Leave a Copy Down*
LCE	*Leave a Copy Everywhere*
LFU	*Least Frequently Used*
LRU	*Least Recently Used*
MP2P	*Multi-Point-to-Point*
MQTT	*Message Queue Telemetry Transport*
NRR	*Nearest Replica Routing*
P2P	*Point-to-Point*
PaaS	*Platform as a Service*
P2MP	*Point-to-Multi-Point*
QoS	*Quality of Service*
RPL	*Routing Protocol for Low-Power and Lossy Networks*
SaaS	*Software as a Service*
SPR	*Shartest Path Routing*
UDP	*User Data Protocol*
WSN	*Wireless Sensor Networks*
XAMPP	*X (qualsevol dels diferents sistemes operatius) Apache MariaDB PHP Perl*

Llistat de figures

Llistat de taules

1 Introducció

1.1 Context

L'augment de sensors i actuadors de baix cost juntament amb l'avanç en tecnologies de comunicació sense fils ofereixen oportunitats sense precedents per al suport d'aplicacions que abasten moltes àrees de la vida moderna (per exemple, *smart city, smart home*, etc.). La necessitat de connectar i integrar aquests dispositius a escala global ha creat el concepte del Internet de les coses (IoT), on els objectes físics quotidians es converteixen en coses intel·ligents que senten, entenen i reaccionen al seu context.

Fins al moment, s'han proposat diferents solucions de xarxa per donar-li vida al IoT. A més de l'enfocament evolutiu centrat en el host basat en adreces IP, també s'han considerat recentment solucions revolucionàries, com la xarxa d'informació centrada a la xarxa (ICN).

Amb vista a l'evolució de la infraestructura d'Internet, la *Information Centric Networking* (ICN) esta promovent un model de comunicació que és fonamentalment diferent del model tradicional centrat en les adreces IP. L'enfocament que dóna ICN consisteix en la recuperació de contingut per noms exclusius, independentment de la ubicació del servidor d'origen, l'aplicació i el canal de distribució, la qual cosa permet el caching/replicació a la xarxa i la seguretat basada en contingut. Els beneficis esperats, en termes d'una millor eficiència i robustesa de la difusió de les dades en escenaris de comunicació desafiadores, indiquen l'alt potencial de ICN com un paradigma de xarxes innovadores en el domini del internet de les coses (IoT). [1]

Un altre dels reptes per al IoT és poder fer front a l'alt nivell de dinamisme de la xarxa, es necessita poder administrar la connectivitat intermitent en IoT. Una de les propostes per solucionar-ho és considerar una comunicació tolerant al retard mitjançant un model de xarxa *Delay Tolerant Network* (DTN), la qual permetria que els objectes intel·ligents es comuniquin millor fins i tot amb la presència d'interrupcions en la seva connectivitat. [2]

S'ha volgut realitzar l'estudi d'un sistema IoT que permeti actuar de forma distribuïda i no depenent d'un node central, d'aquesta manera s'augmentaria la resiliència de la solució. La idea es realitzar l'estudi en un entorn urbà on la cobertura dels canals habituals és limitada, tenint en compte aquestes mancances de seguretat i de distribució de contingut i on poder experimentar les millores en un cas pràctic amb la implementació de les xarxes ICN i les DTN.

1.2 Idea principal

Arribats a aquest punt, proposem un sistema IoT per al control intel·ligent de la temperatura d'un edifici, mitjançant una xarxa mallada de sensors de temperatura, la qual estigui interconnectada per poder orquestrar de manera autònoma els diferents punts, sense necessitat d'una gestió centralitzada des del centre de control.

D'acord amb la idea principal exposada en la secció anterior, aquest projecte té com a primer objectiu realitzar un estudi de l'estat de l'art de les xarxes ICN. Això engloba explorar quines solucions de ICN hi han actualment, si hi ha alguna proposta per IoT, analitzar quina arquitectura de ICN s'adequaria més a unes característiques IoT, etc.

El segon objectiu seria la realització d'una proposta de l'arquitectura bàsica que es podria implementar en un escenari d'aquestes característiques.

I l'últim objectiu seria l'adaptació i la implementació d'aquesta arquitectura en un cas pràctic, per poder analitzar el seu funcionament.

1.3 Pla de desenvolupament

El desenvolupament d'aquest projecte seguirà la següent estructura:

1. Definició del context i objectius del projecte
2. Estudi de l'estat de l'art dels paradigmes de comunicació ICN i DTN.
3. Estudi de l'estat de l'art de les tecnologies de la comunicació IoT de llarg i curt abast.
4. Disseny d'una solució (proposta d'aplicació d'ICN i DTN en entorns IoT).
5. Especificació de la solució: sistema d'adreçament, protocols, intercanvi de trames, sistema de enrutament, sistema d'emmagatzematge, etc.
6. Aplicació d'un cas d'ús: gestió d'una illa autònoma de punts, mitjançant una xarxa mallada de sensors de temperatura.
7. Anàlisis de resultats
8. Conclusions i línies de futur

2 Estat de l'art

En aquest apartat es fa un recull de tota la informació que s'ha trobat respecte a les tecnologies ICN i DTN, així com d'altres temes necessaris per poder dissenyar i implementar el sistema WSN per al control de temperatura d'un edifici.

2.1 Information Centric Networking

Les xarxes centrades en la informació ICN són una solució centrada en la informació que permet la comunicació sense tenir en compte una ubicació especifica. Deslligant la informació d'una ubicació determinada, aquesta pot estar replicada i distribuïda en diversos dispositius, i l'usuari no atacaria a un host específic per trobar la informació que busca, sinó demanaria directament la informació que vol.

Els nodes poden sol·licitar dades per mitjà d'identificadors únics i, en principi, qualsevol node pot proporcionar les dades, encara que no sigui la seva font original. En aquestes condicions, quan un node reenvia una dada, és convenient, des del punt de vista de la xarxa, que mantingui les dades en un caché local per satisfer més sol·licituds futures. [3]

S'enfoca en accedir i entregar informació en base al seu nom, es una opció natural per a la comunicació en les situacions de desastre, oferint tant l'oportunitat com la cobertura necessària per aquestes comunicacions crítiques. [4]

Ofereix una distribució més eficient del contingut, ja que la xarxa respondria a la petició de l'usuari amb la localització més propera del contingut i la hi enviaria. Això suposa avantatges en quant a l'ús dels recursos de xarxa i el temps d'enviament de les dades.

Està dissenyat per proporcionar autenticitat e integritat de les dades que s'identifiquen per el seu nom. La capacitat d'emmagatzematge i reenviament basada en el nom del contingut facilita considerablement la recuperació de dades en entorns molt distribuïts, ja que permet que les dades es repliquin en varies ubicacions per una millor difusió de la informació i accés ràpid inclús en situacions adverses. [4]

Pot exercir un paper clau cap a l'accés universal a Internet. Pot canviar la forma en què els usuaris es comuniquen i accedeixen a la informació, passant del paradigma d'accés centrat en el host tradicional, on l'accés a un contingut s'assigna a la seva ubicació (fixa) a un model centrat en la informació, que elimina aquest contingut d'assignació i admet l'accés independentment del lloc on es trobi el contingut. ICN fa una gestió eficient dels recursos la qual cosa que permet l'optimització conjunta de la capacitat de la xarxa, l'emmagatzematge i els recursos de càlcul. [5]

Tot i que hi ha propostes de ICN que varien en terminologia, implementació i API a client a operadors de xarxes, hi han quatre temes principals subjacents a totes les propostes [6]:

1. **Desacoblament del noms de les ubicacions:** les aplicacions i protocols de xarxa es reestructuren perquè la comunicació es basa en la cerca i transferència de contingut en contrast amb les abstraccions centrades en el host d'avui dia.
2. ***Pervasive Caching*:** En el límit, cada router de la xarxa també actua com a caché de contingut. Això significa que a més de les responsabilitats tradicionals de reexpedició, els routers també serveixen sol·licituds de contingut en els seus caches.
3. **Nearest-replica Routing (NRR)**: l'enrutament es basa en noms de contingut en lloc d'en hosts, de manera que les sol·licituds s'enruten a la còpia del contingut més propera. (En el pitjor dels casos, és el servidor d'origen que allotja el contingut.)
4. **Vincular noms a la intenció**: El nom del contingut està intrínsecament lligat a la intenció de l'editor de contingut i el consumidor. Aquest enllaç ajuda als usuaris (i routers) a comprovar la integritat i la procedència de les dades sense indicadors externs.

Aquets son els beneficis que aporten les ICN que s'han trobat a estudis previs [6]:

Benefit	Feature			
	Decoupling names from locations	Pervasive Caching	Nearest-replica routing	Intrinsic Binding
Latency (§4, §5)		✓	✓	
Traffic Engg. (§4, §5)		✓	✓	
Mobility (§6)	✓		✓	
Ad hoc mode (§6)	✓		✓	
Security (§6)	✓			✓

Taula 1. Característiques i beneficis de ICN. Font [6]

- **Menor latència de resposta:** una infraestructura amb *pervasive caching* significa que les sol·licituds no necessàriament necessiten recórrer tota la xarxa cap al servidor d'origen.
- **Enginyeria de tràfic simplificada:** El caching també ajuda als operadors de xarxa eliminant automàticament els hotspots de contingut, la qual cosa simplifica la lògica d'enginyeria de tràfic necessària per equilibrar la càrrega de la xarxa.
- **Seguretat:** En elevar el contingut com a ciutadà de primera classe, el ICN vincula intrínsecament la intenció de l'usuari a les dades finals que es lliuren sense haver de confiar en la confirmació externa de la procedència o autenticitat de les dades.
- **Mobilitat:** El canvi del enrutament centrat en el host o en el contingut també facilita el suport als clients mòbils, ja que els problemes tradicionals amb transferències, retransmissions, etc., simplement desapareixen.
- **Mode Ad hoc:** Un altre benefici de ICN és la capacitat dels nodes per comunicar i compartir contingut sense cap tipus de suport d'infraestructura.
- **Altres:** Hi ha altres beneficis percebuts com la resistència a la DDoS i la tolerància a la interrupció que són menys explorats. Aquests semblen ser instàncies específiques o combinacions dels beneficis anteriors. Per exemple, la tolerància a la interrupció sembla ser una combinació de suport per a la mobilitat i la manera ad hoc. De la mateixa manera, la resiliència DDoS prové d'evitar punts d'accés de contingut i caching universal.

El servei de *consulta* de ICN s'implementa definint dos tipus de formats de paquet de xarxa: paquets d'interès que sol·liciten contingut per nom i paquets de dades que transporten el

contingut sol·licitat. El funcionament del protocol *Request-Reply* és tal i com es pot veure en la Figura 1, el paquet de dades retornat ha de coincidir amb els paràmetres de la sol·licitud (per exemple, tenir un nom que coincideixi total o parcialment). Si la sol·licitud és ambigua i diversos paquets de dades satisfarien la sol·licitud, la xarxa del ICN típicament retornarà solament un paquet de dades coincident. **[7]**

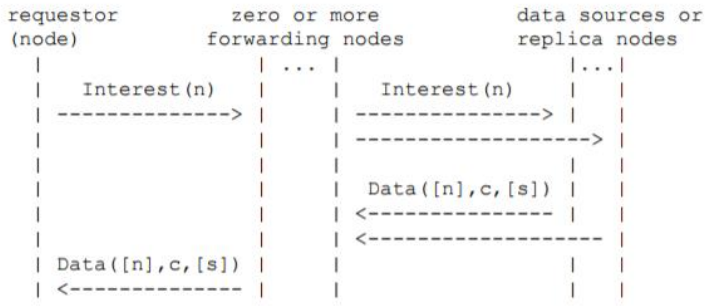

Figura 1. Protocol Request-Reply de ICN. Llegenda: n=name, c=content, s=signature. Font [7]

Els tipus de paquets son els següents:

- **Paquet de dades:** un paquet de nivell de xarxa que porta càrrega útil, identificat de manera única per un nom, i està directament protegit.
- **Paquet d'interès:** un paquet de nivell de xarxa que expressa la sol·licitud d'un paquet de dades utilitzant un nom exacte o un prefix de nom. Un paquet d'interès pot tenir opcionalment un conjunt de restriccions addicionals (per exemple, selectores d'interessos). Un interès pot associar-se amb informació addicional per facilitar la reexpedició i pot incloure interès per a tota la vida, límit de salt, pistes de reexpedició, etiquetes, etc. En diferents dissenys de ICN, el conjunt d'informació associada addicional pot variar.

ICN promet millorar els serveis orientats al contingut en permetre el caching dins de la xarxa i admetre l'enviament òptim de sol·licituds de contingut. Per aconseguir aquest objectiu, es proposen diferents esquemes de caché de contingut i reexpedició de sol·licituds en múltiples arquitectures de ICN.

L'esquema de contingut de caching determina on col·locar l'emmagatzematge en caché, si un node de caché ha d'emmagatzemar en caché un contingut i quin contingut reemplaçar si el caché està ple. Un proveïdor de serveis pot col·locar l'emmagatzematge en caché solament a la vora d'Internet, o distribuir-ho als nodes interns (*pervasive caching* en extrem). Quan les dades tornen, certs nodes de caché en ruta poden emmagatzemar en caché el contingut seguint cert algorisme d'emmagatzematge en caché, per exemple, leave copy everywhere (LCE), leave copy down (LCD), seleccionar el node amb la major intersecció. A més, una l'emmagatzematge en caché normalment usa LRU o de LFU com a política de reemplaçament.

Per a l'estratègia de reexpedició de sol·licituds, l'enrutador podria reexpedir una sol·licitud de contingut seguint el shortest path routing (SPR) cap al servidor de contingut original mentre s'explora el caché de contingut en el camí per atendre la sol·licitud; de forma més proactiva,

l'enrutador pot reexpedir directament la sol·licitud a un caché de contingut proper si el enrutador coneix la informació del caché, la qual cosa resulta en el nearest-replica routing (NRR).

ICN originalment opta pel *pervasive caching* i el NRR. No obstant això, recentment s'ha reavivat un debat sobre la necessitat de ICN, especialment la utilitat del *pervasive caching*. Es converteix en una necessitat urgent comprendre millor els pros i els contres de l'emmagatzematge en memòria caché de diferents continguts i sol·licitar opcions de disseny de reexpedició en ICN. De la comparació, aconseguim les següents observacions [8]:

- El *pervasive caching* pot no proporcionar beneficis substancials com proposa el ICN, mentre que una configuració de memòria caché més simple ja podria aconseguir la major part del guany. Trobem que l'emmagatzematge en memòria *pervasive caching* pot aconseguir com a màxim un 20,2% de capacitat de xarxa més gran en els nostres experiments.

- El *nearest-replica routing* (NRR) garanteix que s'atengui una sol·licitud des de la "millor" ubicació, la qual cosa ajuda a optimitzar el lliurament de contingut, especialment per a una xarxa gran.

- Per a un sistema amb *Edge caching* bàsic i funcionalitat de enrutament de ruta més curta (SPR), habilitar el NRR brinda un major guany de rendiment (per exemple, una capacitat de xarxa 98,4% major) que l'habilitació de *pervasive caching*.

S'han analitzat les següents opcions de disseny per representar diferents esquemes de caché a l'espai de disseny de ICN:

- **PERV-LCE-SPR:** aquest disseny adopta la col·locació pervasive cache i l'algorisme de caché LCE, on cada node de la xarxa té un magatzem de memòria caché i tots els nodes de la ruta emmagatzemen en caché el contingut quan les dades tornen. A més, assumeix l'enrutament de ruta més curta cap al servidor original per a la reexpedició de sol·licituds.
- **PERV-LCD-SPR:** aquest disseny és el mateix que PERV-LCESPR, excepte en l'algorisme d'emmagatzematge en caché que és LCP en lloc de LCE, és a dir, solament el següent node en la ruta guarda en caché el contingut quan les dades tornen.
- **PERV-LCE-NRR:** aquest disseny amplia PERV-LCE amb el enrutament de rèplica més proper en el qual suposem que sempre podem trobar una ruta ideal a la rèplica més propera per a una sol·licitud de contingut.
- **PERV-LCD-NRR:** aquest disseny amplia PERV-LCD amb el enrutament de rèplica més proper.
- **EDGE-SPR:** en aquest disseny, solament col·loquem el caché en la "vora" de la xarxa, per exemple, els nodes de vora d'una topologia d'arbre. Cada node de caché té la mateixa capacitat d'emmagatzematge en caché que en el disseny de caché *pervasive caching*.
- **EDGE-NRR:** és el mateix que EDGE-SPR, però usa la forwarding de NRR.
- **EDGE-SPR-NORM:** utilitza el mateix caché i mecanisme de reexpedició de sol·licituds com EDGE-SPR. No obstant això, el capacitat de caché total a la xarxa és igual al caché

total en el disseny de caché *pervasive caching*. Per tant, per a una topologia d'arbre binari, la capacitat de caché en cada node de caché es duplica.

- **EDGE-NRR-NORM:** comparteix el mateix esquema que EDGE-NRR, i té la mateixa capacitat de caché que el caché *pervasive caching*.

A partir de l'avaluació, s'ha conclòs que el *pervasive caching* podria no obtenir un guany significatiu sobre el *edge caching*, mentre que el NRR amb el *edge caching* podria aportar la majoria dels beneficis.

Per a servidors de caché de xarxa local realistes amb capacitats limitades d'emmagatzematge en caché, es requereixen polítiques de reemplaçament adequades per decidir si un contingut ha d'inserir-se en el caché i, si el caché està ple, quin contingut ha d'eliminar-se per generar espai. En general, els factors més importants que afecten l'eficiència de l'emmagatzematge en caché (generalment denominats taxa d'encerts de caché com es descriu a continuació) inclouen recurrència, freqüència, grandària, cost de cerca i temps de modificació/caducitat del contingut en caché. La política de reemplaçament de caché més utilitzada i més fàcil d'implementar és *Least recently used* (LRU)**,** que reemplaça el contingut utilitzat menys recentment. L'algorisme porta el seguiment del que es va usant, la qual cosa resulta car si es vol fer amb precisió. La implementació d'aquesta tècnica requereix portar el compte de l'edat de cada element de cache i buscar el menys usat sobre la base d'ella. En una implementació com aquesta, cada vegada que s'usa un element, l'edat de tots els altres elements canvia. Un altre variant que també s'utilitza bastant podria ser *Least frequently used* (LFU)**,** utilitzat per reemplaçar el contingut menys utilitzat.

2.2 Delay Tolerant Networking

L'arquitectura de xarxa amb tolerància al retard DTN està dissenyada per treballar en entorns sense connectivitat extrem a extrem, amb grans retards en la comunicació, canals asimètrics i dispositius heterogenis. Per garantir la comunicació extrem a extrem es defineix el protocol de parquets (*Bundle Protocol*) que permet el lliurament dels missatges sense importar els protocols subjacents. Degut al seu disseny, la DTN és una arquitectura que podria resultar adequada para donar resposta a les necessitats comunicatives de la IoT. [9]

DTN fa ús de mecanismes de *store-carry and forward* que permeten a un node emmagatzemar els missatges mentre no hi ha comunicació i lliurar-los en quan es produeix el contacte amb altres nodes. [9] Aquesta funcionalitat s'aconsegueix amb la introducció d'una capa de missatges (bundle layer) situada entre la capa d'aplicació i la capa de transport amb l'objectiu d'unir xarxes diferents i de naturalesa heterogènia. D'aquesta manera, les capes inferiors que usen protocols específics depenent d'on es trobin aquestes xarxes, no siguin un impediment per a la comunicació de les aplicacions, situades en la capa superior del nou protocol introduït. La seva funció és dotar a les comunicacions entre diferents xarxes d'independència de les capes inferiors.

El nou protocol introduït és comú per totes les xarxes que formen les xarxes DTN. Aquesta capa és la que emmagatzema i reenvia els missatges entre els diferents nodes de les xarxes, a més, també és capaç de fragmentar els missatges de la mateixa manera que actua IP. [10]

Figura 2. Arquitectura de la xarxa DTN. Font [10]

L'emmagatzematge de les dades es realitza de forma persistent en els nodes intermedis per ser retransmesos davant una interrupció. Amb aquesta tècnica els missatges es mouen, en el seu conjunt o per fragments, d'un node en el qual es troba emmagatzemat cap a un altre node, es verifica que la informació estigui completa i després es reenvia al següent node, repetint-se aquest procés en cada node de la ruta fins a arribar al node destí Figura 3. Quan es produeixen errors, en tractar-se d'informació important, es requereix el reenviament de la informació. D'aquesta manera, quan un node envia un missatge, no ho pot eliminar en el moment doncs pot ser que es produeixin errors en la recepció; s'hauria d'esperar a una confirmació d'arribada per part del node destí. [10]

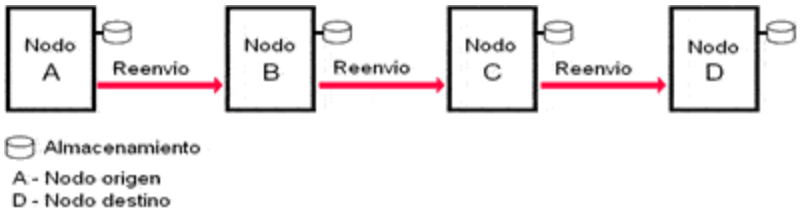

Figura 3. Mecanisme DTN *store-carry and forward.* Font [10]

Una altra característica de la xarxes DTN és la transferència missatges sota custòdia, la qual és important ja que en cas de reenviament el paquet no hagi de creuar la distància des del node que li va donar origen a l'enllaç fins a la seva destinació novament. Això és molt útil quan s'intenta enviar dades sobre enllaços intermitents Figura 4.

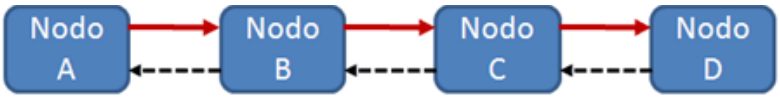

Figura 4. Transferència sota custodia. Font [10]

L'arquitectura DTN no exigeix que tots els nodes DTN acceptin la transferència sota custòdia. Per això no es considera un mecanisme de salt a salt legítim. En cas que no s'accepti la transferència sota custòdia, el mecanisme de retransmissió no és emprat de manera que l'èxit en el lliurament del missatge dependrà dels protocols de les capes inferiors. [10]

2.3 Fonaments teòrics i anàlisi de tecnologies

Tenint en compte totes aquestes característiques de ICN i DTN s'han estudiat diferents maneres i casos d'aplicació en funció de la utilitat que se li vulgui donar.

2.3.1 Arquitectura ICN-IoT

S'ha proposat una arquitectura de IoT unificada basada en l'arquitectura de la Xarxa Centrada en la Informació (ICN), que s'anomena ICN-IoT. El paisatge de IoT està molt fragmentat tant des de la perspectiva de la tecnologia com del servei, per això és difícil integrar i correlacionar les dades provinents de contextos heterogenis i generar serveis de valor agregat en la part superior. Aquesta raó ha motivat la tendència actual de desenvolupar una arquitectura de IoT unificada desfragmentada perquè els objectes puguin ser accessibles a les aplicacions en organitzacions i dominis.

ICN-IoT aprofita les excel·lents característiques de ICN, proporcionant la comunicació de dispositiu a dispositiu (D2D), suport de mobilitat, escalabilitat i contingut eficient i lliurament de serveis. La Figura 5 mostra l'enfocament proposat ICN-IoT, que se centra entorn de la xarxa del ICN en lloc del Internet d'avui.

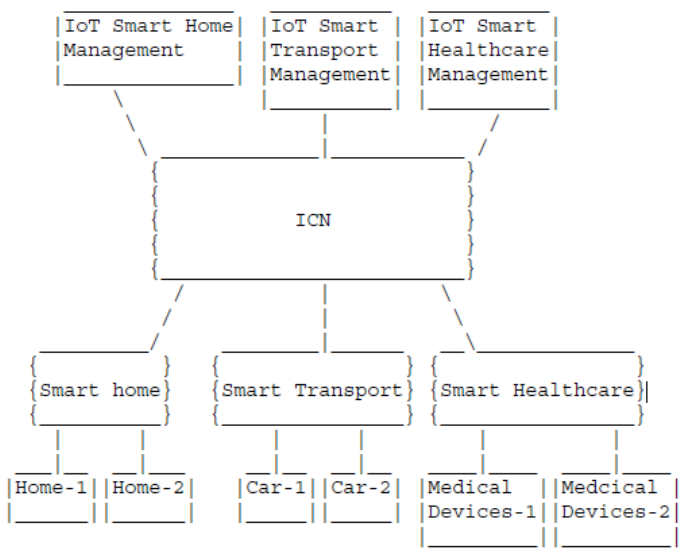

Figura 5. La proposta d'arquitectura unificada ICN-IoT. Font [11]

Els punts forts de ICN-IoT són els següents:

- **Nombrament.** S'assigna un nom exclusiu a un objecte IoT, un servei IoT o fins i tot un context. Aquests noms són persistents en tots els seus àmbits.
- **Seguretat i privadesa.** La unió segura entre noms i objectes de dades en lloc d'adreces IP per identificar dispositius/dades /serveis, és intrínsecament més segura que el paradigma IP tradicional.
- **Escalabilitat.** La resolució de noms es realitza en la capa de xarxa en línia o mitjançant un sistema de mapatge amb l'estat del nom distribuït dins de tota la xarxa. Per tant, pot

aconseguir un alt grau d'escalabilitat aprofitant característiques com la ubicació de contingut, la informàtica local i la multidifusió.

- **Eficiència de recursos.** Tant les parts restringides com les no restringides de la xarxa, solament es lliuraran aquelles dades subscrites per les aplicacions o sol·licitats pels clients en el context especificat.
- **Patró de tràfic local.** Es pot emmagatzemar fàcilment les dades o implementar serveis de computació a la xarxa, la qual cosa fa que les comunicacions siguin més localitzades i redueixi el volum de tràfic general.
- **Comunicacions amb coneixement de context.** Admet contextos en diferents capes, inclosa la capa del dispositiu, la capa de xarxa i la capa d'aplicació. Com a resultat de l'adopció de contextos i comunicacions contextuals, les comunicacions solament es produeixen en situacions especificades per les aplicacions, que poden reduir significativament el volum de tràfic de la xarxa.
- **Maneig sense problemes de la mobilitat**. La capa de resolució de noms de ICN permet múltiples nivells de mobilitat que depenen de la naturalesa orientada al receptor per la autorrecuperació dels consumidors, i l'ús de tècniques de multidifusió i late-binding per aconseguir un suport de mobilitat sense fissures dels nodes productors.
- **Autoorganització.** Les xarxes basades en noms permeten la autoconfiguració i l'arrencada de dispositius de nivell de servei de dispositius amb una configuració mínima proporcionada pel fabricant o administrador.
- **Emmagatzematge de dades i caching.** Les dades s'emmagatzemen localment, ja sigui pel dispositiu mòbil o pels nodes de la porta d'enllaç o en els punts de servei. El caching dins de la xarxa també accelera el lliurament de dades i també ofereix reparació local sobre enllaços no fiables com, per exemple, a través de mitjans sense fils.
- **Fiabilitat de la comunicació.** Suporta comunicacions tolerants a retards [23], que al seu torn proporciona comunicacions confiables sobre enllaços no fiables com es va esmentar anteriorment. A més, el caching oportunista permet augmentar les còpies de contingut a la xarxa per ajudar als consumidors amb diversos requisits d'aplicació (com operar a diferents taxes de sol·licitud) o situacions relacionades amb escenaris de mobilitat.
- **Mode ad hoc i infraestructura.** Admet ambdues aplicacions que funcionen en maneres ad-hoc i d'infraestructura.
- **Processament dins de la xarxa.** Ofereix un processament dins de la xarxa que admet diversos serveis de xarxa, com la resolució de context, l'agregació de dades i la compressió.

La arquitectura del sistema ICN-IoT, te els següents sis components principals del sistema:

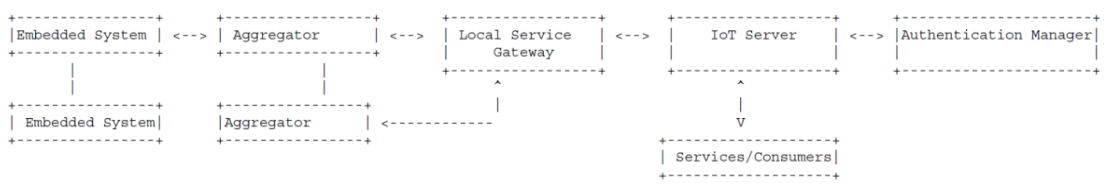

Figura 6. Sistema d'Arquitectura ICN-IoT. Font [11]

La topologia lògica del sistema IoT pot ser similar a una malla, amb cada ES (*Embedded System*) connectat a un o més agregadors o a una altra ES, mentre que cada agregador està connectat a un o més LSG (*Local Service Gateway*) i tots els LSG connectats al servidor IoT. Per tant, cada ÉS té els seus agregadors, i cadascun dels quals té el seu LSG. Tots els ES que pertanyen al mateix servei de IoT comparteixen el mateix servidor de IoT. Tots els agregadors que estan connectats a la mateixa administració de LSG poden comunicar-se entre si, per tant, poden sol·licitar serveis o contingut d'ells. Si bé una connectivitat tan rica millora la confiabilitat, també requereix sofisticació en el plànol de control per administrar la interconnexió.

El middleware ICN-IoT proposat pretén tancar la bretxa entre les funcions subjacents del ICN, les aplicacions IoT i els dispositius per aconseguir la capacitat d'autoorganització. Les funcions de middleware es mostren en la Figura 7 inclouen sis funcions bàsiques: detecció de dispositius, servei de nombrament, descobriment de serveis, processament i emmagatzematge de context, gestió de publicació-subscripció i seguretat la qual engloba totes aquestes funcions.

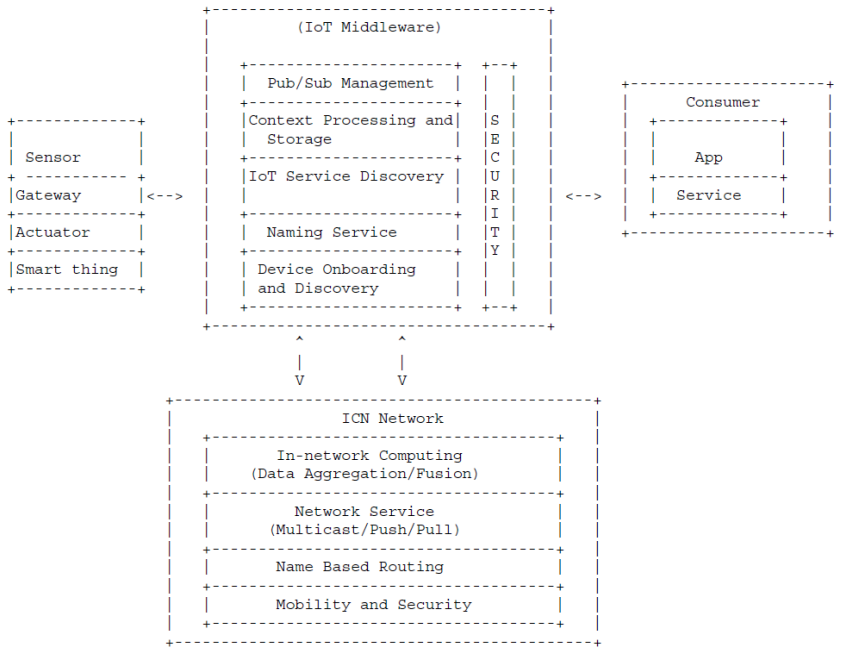

Figura 7. Funcions ICN-IoT Middleware. Font [11]

A diferència de la implementació centralitzada o superposada en la plataforma IoT heretada basada en IP, l'arquitectura ICN-IoT impulsa una gran part de les funcionalitats a la capa de xarxa, com la resolució de noms, la gestió de la mobilitat, el processament a la xarxa/caching, processament del context, la qual cosa simplifica bastant el disseny del middleware. [11]

2.3.2 Situacions desastre

Una de les casuístiques on ICN seria de gran utilitat seria en situacions de desastre, on els quals ocasionen interrupcions en la infraestructura de comunicació, normalment fragmentant una multitud de xarxes. La qual cosa fa que sigui difícil intercanviar missatges simples, encara que

crítics, en gran part de la població afectada, inclosos els que responen primer i les autoritats governamentals. Es proposa un model de recuperació de dades impulsat per el contingut, es proposa un enfoc millorat de ICN per proporcionar a la comunicació tolerància al retard en la gestió de desastres (DID – Delay-tolerant ICN for Disaster-management). Les principals contribucions de DID són:

- Millorar la capacitat de recuperació de les comunicacions després d'un desastre on les xarxes poden estar fragmentades, la qual cosa requereix una comunicació tolerant als retards entre els fragments.
- Separació entre una cara lògica d'un node i una interfície física, per permetre el funcionament de mules de dades que són mòbils i poden connectar-se a diferents punts lògics, com a comunitats fragmentades en diferents períodes de temps en l'entorn propens a les interrupcions típic de les situacions de desastre.
- Millora els protocols de ICN per permetre que l'interès i les dades viatgin per camins completament diferents i tolerin una demora gran i variable entre quan s'expressa un interès i quan les dades estan disponibles.
- Utilitza de manera eficient els recursos de comunicació disponibles limitats en assignar més recursos a les dades de major prioritat, però no a les dades de menor prioritat. La prioritat de l'intercanvi de missatges entre dos nodes té en compte factors com les característiques de les dades, el nombre de persones interessades en les dades, el patró de mobilitat de tots dos mules de dades i el consumidor/productor de dades.
- Les avaluacions preliminars mostren que el DID és de fet més eficient que els enfocaments com Epidemic i SprayAndWait pel que fa al retard, la probabilitat de resposta i el rendiment general. Té èxit en lliurar missatges d'alta prioritat sense passar per alt els de baixa prioritat.

Aquest disseny de DID pren en consideració el fet que les mules de dades, que viatgen entre comunitats fragmentades en un escenari de desastre, es comporten com enrutadors mòbils. Quan aquestes mules troben diferents nodes (portes d'enllaç comunitàries fragmentades, altres mules) en diferents períodes de temps amb diferents probabilitats, és important desenvolupar una solució que prioritzi quines dades s'intercanvien. A més, és important dissenyar la solució de manera que les dades no necessàriament hagin de prendre la ruta inversa en la qual es va lliurar l'interès [4].

Existeixen moltes situacions en les quals una infraestructura fixa manca o està temporalment no disponible, o en la qual es prefereix un enfocament distribuït i descentralitzat. Es creu que Opportunistic Networking representa una solució ideal per a aquests escenaris, gràcies a la seva facilitat d'implementació, resistència i naturalesa distribuïda. S'han considerat diferents possibilitats, centrades tant en l'aplicació del món real com en més estudis teòrics per millorar les prestacions del protocol, les quals intenten combinar ICN i DTN.

Per una banda, es proposa un ICN tolerant a la demora per a la gestió de desastres (DID), un protocol basat en la difusió controlada de la informació, que prioritza els missatges i presa en consideració l'historial i l'adreça de les trobades dels nodes

D’altre banda, la idea que es buscava és combinar protocols de enrutaments múltiples, permetent que cada node decideixi dinàmicament quin usar d'acord amb les condicions locals i el tipus de missatge [3].

2.3.3 Arquitectura RIFE

RIFE es una arquitectura unificada per proporcionar accés global a Internet. El que proposa és una plataforma arquitectònica integradora que reuneix a IP, ICN i DTN en un sol marc, en el qual DTN complementa les actuals solucionis IP i ICN com un candidat ideal per a la comunicació en entorns de xarxa, on la tolerància de retard i interrupció afegida permet l'operació en xarxes desconnectades entorns.

L'adopció de ICN i DTN podria desenvolupar mecanismes millorats de QoS per a la infraestructura backhaul, utilitzant la naturalesa generalment centrada en la informació dels seus desplegaments de xarxa de vora, més específicament la naturalesa oportunista de DTN.

El suport per als nous mecanismes de QoS impulsarà noves aplicacions i innovacions, no només buscant serveis d'alt ample de banda, sinó també utilitzant lower-than-best-efford QoS per reduir del preu del sovint tan car del backhaul per portar més contingut i més més serveis a àrees i comunitats remotes. En aquesta coincidencia de context i de contingut es recolza específicament en la informació centralitzada d’aquesta arquitectura, la qual cosa al seu torn condueix a noves solucions per a la gestió de QoS que s'implementen a través de la funció dedicada d'administració de recursos i topologia de la nostra arquitectura.

A través d'aquest suport per a noves solucions de QoS, així com el major desacoblament en temps i en espai, aquesta arquitectura serà capaç de complementar els models econòmics existents de preus d'accés a Internet amb nous que apunten a la revenda de la connectivitat permanent per proporcionar serveis lower-than-best-effort, els quals es poden utilitzar per omplir cachés altament distribuïts i proveïdors substituts a un cost significativament més baix [5].

2.4 Servidors IoT

La funció d’un servidor IoT es pot realitzar de diferents maneres, d’una banda seria crear el nostre propi servidor web que seria una solució “*on-premises*” i d’altre banda es podria fer ús d’alguna plataforma IoT la qual seria una solució del tipus IaaS, PaaS o SaaS. Segons el s’utilitzi es té un nivell de domini sobre la plataforma i la resta està al cloud, tal i com es pot observar a la Figura 8. L’elecció de quin es millor en aquest cas es en funció de les necesitats.

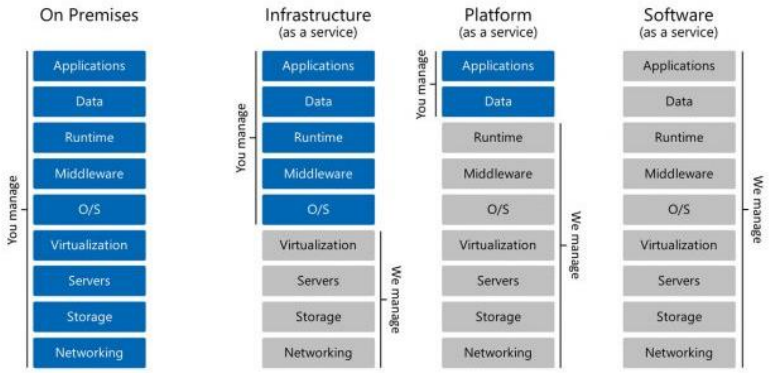

Figura 8. Models "On Premises" i de Servei al Cloud. Font [12]

2.4.1 Servidor Web

Un servidor web seria la opció d'una solució "*on premises*", és la opció que s'implementa de la manera més tradicional. Tot es troba instalat localment, això implica que és té control total tan de la infrastructura com de les dades en tot moment, pero d'altre banda això també comporta unes responsabilitats de gestió.

Una de les constants en els sistemes que donen suport a la Internet de les coses són les bases de dades perquè ofereixen un mètode molt eficaç equilibrat en esforç d'instal·lació, consum de recursos (processador i memòria a curt i llarg termini) manteniment i rendiment.

La manera més comuna d'explotar la base de dades és conforme a una arquitectura client-servidor. El dispositiu electrònic, l'objecte de la Internet de les coses, actuaria com a client i es disposaria d'un sistema autònom, freqüentment un ordinador complet, treballant com a servidor de les bases de dades.

Hi ha multitud de formes de connectar amb el sistema que suporta el servidor de bases de dades, tractant-se d'un equip de la Internet de les coses sembla que el més intuïtiu és pensar en una connexió de xarxa remota o una local, segurament WiFi, que possiblement connecti a per mitjà d'un router a una altra remota.

Com connectar directament amb la base de dades pot ser complex i si es tracta d'un sistema públic cal afegir altres qüestions que poden afectar a la seguretat i al rendiment, el més comú és accedir a la base de dades des d'un servidor web que, encara que segurament limiti una mica el rendiment, afegeix una capa al sistema que fa més senzilla la gestió. [13]

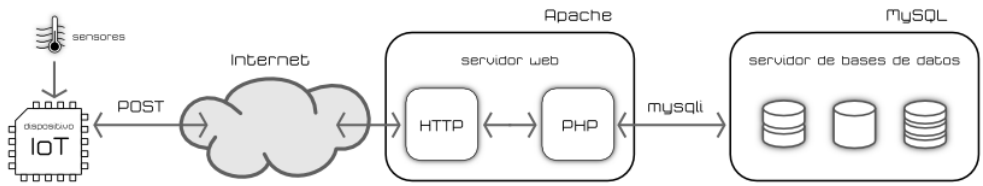

Figura 9. Servidor IoT. Font [13]

2.4.1.1 Servidor Web Apache

El sistema que s'usa amb més freqüència en aquests treballs és el basat en alguna distribució Linux, executant un servidor Apache des del qual es pot accedir a un servidor de bases de dades MySQL o MariaDB des del llenguatge de programació PHP. [13] A aquesta configuració se li sol donar el nom de LAMP (Linux Apache MySQL/MariaDB Perl/PHP/Python) per les sigles dels principals components.

Figura 10. Servidor Apache HTTP

El servidor Apache HTTP, també anomenat Apache, és un servidor web HTTP de codi obert per a la creació de pàgines i serveis web. És un servidor multiplataforma, gratuït, molt robust i que destaca per la seva seguretat i rendiment. Consisteix principalment en el sistema de gestió de bases de dades MySQL

2.4.1.1.1 XAMPP

XAMPP és una distribució d'Apache completament gratuïta i fàcil d'instal·lar que conté MariaDB, PHP i Perl. El nom és en realitat un acrònim: X (para qualsevol dels diferents sistemes operatius), Apache, MariaDB, PHP, Perl.

Figura 11. XAMPP

Molta gent coneix de primera mà que no és fàcil instal·lar un servidor de web Apatxe i la tasca es complica si li afegim MariaDB, PHP i Perl. L'objectiu de XAMPP és crear una distribució fàcil d'instal·lar per a desenvolupadors que s'estan iniciant al món d'Apache. XAMPP ve configurat per defecte amb totes els opcions activades. [14]

2.4.2 Plataformes IoT

Les plataformes IoT també són conegudes com el middleware d'Internet de les Coses, que significa que és tot allò que actua de mediador entre la capa hardware i la capa d'aplicació d'Internet de les Coses, amb la qual cosa, faciliten molt les coses a tot el que vol desenvolupar aplicacions IoT. [15]

A comparació dels servidors web les plataformes IoT ofereixen una opció més predefinida i estan basades en cloud computing oferint solucions IaaS, PaaS i SaaS. Les diferencies entre elles son les següents [16]:

- IaaS: els recursos informàtics oferts consisteixen, en particular, en maquinari virtual, o, en altres paraules, infraestructura de processament. La definició de IaaS abasta aspectes com l'espai en servidors virtuals, connexions de xarxa, ample de banda, adreces IP i balancejadors de càrrega. Físicament, el repertori de recursos de maquinari disponibles procedeix de multitud de servidors i xarxes, generalment distribuïts entre nombrosos centres de dades, el manteniment de les quals s'encarrega el proveïdor del servei cloud. El client, per la seva banda, obté accés als components virtuals per construir amb ells la seva pròpia plataforma informàtica.
- PaaS: permet als usuaris crear aplicacions de programari utilitzant eines subministrades pel proveïdor. Els serveis PaaS poden consistir en funcionalitats preconfiguradas a les quals els clients puguin subscriure's, triant les funcions que desitgin incloure per resoldre les seves necessitats i descartant aquelles que no necessitin. Així, els paquets poden variar des d'un senzill entorn que es manegi amb el ratolí i no requereixi cap tipus de coneixement o instal·lació especial pel costat de l'usuari, fins al subministrament d'opcions d'infraestructura per a desenvolupament avançat.
- SaaS: es coneix també de vegades com a "programari a demanda", i la forma d'utilitzar-ho s'assembla més a llogar el programari que a comprar-ho. Amb les aplicacions tradicionals, el programari es compra al principi com un paquet, i una vegada adquirit s'instal·la en l'ordinador de l'usuari. La llicència del programari pot també establir limitacions quant al nombre d'usuaris i/o dispositius en els quals pot instal·lar-se. Per contra, els usuaris del Programari com a Servei se subscriuen al programari, en lloc de comprar-ho, generalment per períodes mensuals. Les aplicacions es compren i utilitzen a través d'internet, i els arxius es guarden en el núvol, no en l'ordinador de l'usuari.

Hi ha proveïdors de plataformes IoT per a tot tipus de projectes i tecnologies, gratuïtes o de pagament, més personalitzables i/o programables o menys. Els grans proveïdors de cloud computin són Amazon Web Service (AWS) i Microsoft Azure, sent AWS aproximadament el doble de gran que Microsoft Azure que és el seu rival més proper, i 10 vegades més gran que els seus altres competidors junts. [17]

2.4.2.1 IoT AWS

Amazon Web Services (AWS) ofereix un ampli conjunt de productes globals basats en el núvol, incloses aplicacions d'informàtica, emmagatzematge, bases de dades, anàlisis, xarxes, mòbils, eines per a desenvolupadors, eines d'administració, IoT, seguretat i empresarials. Aquests serveis ajuden a les empreses a avançar amb major rapidesa, reduir els costos de TI i escalar.

AWS IoT proporciona comunicació segura i bidireccional entre elements connectats a Internet (dispositius com a sensors, actuadors, microcontroladors integrats o dispositius intel·ligents) i el núvol AWS. Això li permet recopilar dades de telemetria de diverses coses i emmagatzemar i

analitzar les dades. També pot crear aplicacions que permetin als seus usuaris controlar aquests dispositius des dels seus telèfons o tablets. [18]

AWS ha incorporat el IoT a serveis específics, com AWS Greengrass i AWS IoT. Aquests li ajuden a recopilar i enviar dades al núvol, facilitant la càrrega i l'anàlisi d'aquesta informació, i li permeten administrar els seus dispositius, perquè pugui concentrar-se a desenvolupar aplicacions que satisfan les seves necessitats [19].

2.4.2.1.1 AWS IoT Core

AWS IoT és una plataforma de cloud administrat que permet als dispositius connectats interactuar amb facilitat i seguretat amb les aplicacions en el cloud i altres dispositius. AWS IoT admet milers de milions de dispositius i bilions de missatges, i és capaç de processar i enrutar aquests missatges a punts d'enllaç de AWS i a altres dispositius de manera fiable i segura [19].

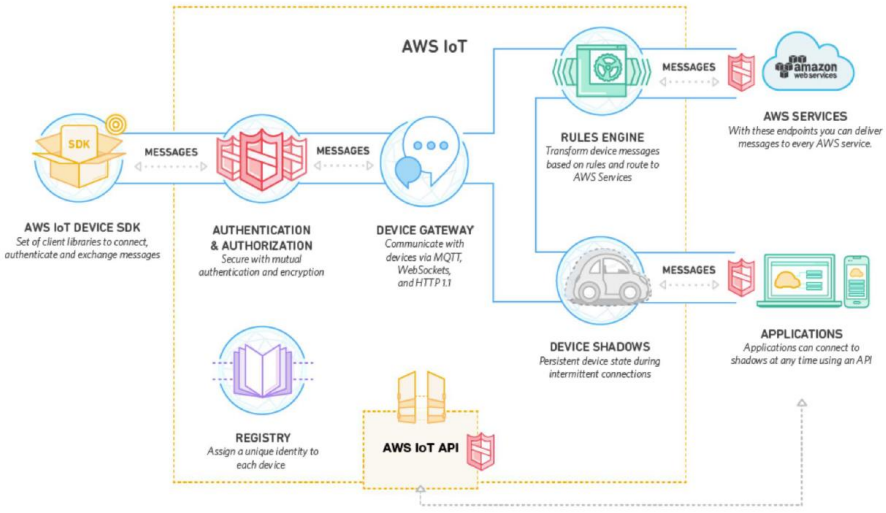

Figura 12. Arquitectura AWS IoT. Font [19]

Beneficis:

- **Connexió i administració de dispositius:** AWS IoT Core li permet connectar fàcilment dispositius al núvol i a altres dispositius. Admet HTTP, WebSockets i MQTT, un protocol de comunicació lleuger, especialment dissenyat per tolerar connexions intermitents, minimitzar la petjada de codi en els dispositius i reduir els requisits d'ample de banda de la xarxa. Així mateix, és compatible amb altres protocols personalitzats i propis del sector, i els dispositius poden comunicar-se entre si encara que utilitzin protocols diferents.
- **Protecció de dades i connexions de dispositius:** AWS IoT Core ofereix autenticació i xifrat integral en tots els punts de connexió, a fi que les dades mai s'intercanviïn entre dispositius i AWS IoT sense una identitat provada. Així mateix, pot protegir l'accés a dispositius i aplicacions mitjançant polítiques amb permisos granulars.

- **Processament i utilització de dades de dispositius:** Amb AWS IoT Core pot filtrar, transformar i utilitzar dades de dispositius sobre la marxa segons les regles empresarials que hagi establert. Pot actualitzar les regles per implementar noves característiques de dispositius i aplicacions quan ho desitgi. AWS IoT Core facilita la utilització de serveis de AWS, com AWS Lambda, Amazon Kinesis, Amazon S3, Amazon Machine Learning, Amazon DynamoDB, Amazon CloudWatch i Amazon Elasticsearch Service per aconseguir aplicacions de IoT fins i tot més potents.
- **Lectura i definició d'estats de dispositius a qualsevol moment:** AWS IoT Core emmagatzema l'últim estat d'un dispositiu perquè pugui llegir-se o definir-se a qualsevol moment, la qual cosa fa que el dispositiu aparegui en les aplicacions com si estigués connectat a tot moment. Això suposa que una aplicació pot llegir l'estat d'un dispositiu fins i tot si està desconnectat; també permet establir l'estat d'un dispositiu i implementar-ho quan es torna a connectar.

2.4.2.1.2 AWS Greengrasss

AWS Greengrass és l'aposta de AWS IoT per delegar part del processament IoT al edge, abans de ser enviat a plataforma. Proporciona un framework programari per construir aplicacions en els dispositius.

AWS Greengrass és un programari que amplia les capacitats del núvol de AWS als dispositius locals i els permet recopilar i analitzar dades més prop de l'origen de la informació, alhora que es comuniquen de forma segura entre si a les xarxes locals. Més en concret, els desenvolupadors que utilitzen AWS Greengrass poden crear codi sense servidor (funcions AWS Lambda) en el núvol i implementar-ho còmodament en dispositius per executar localment les aplicacions.

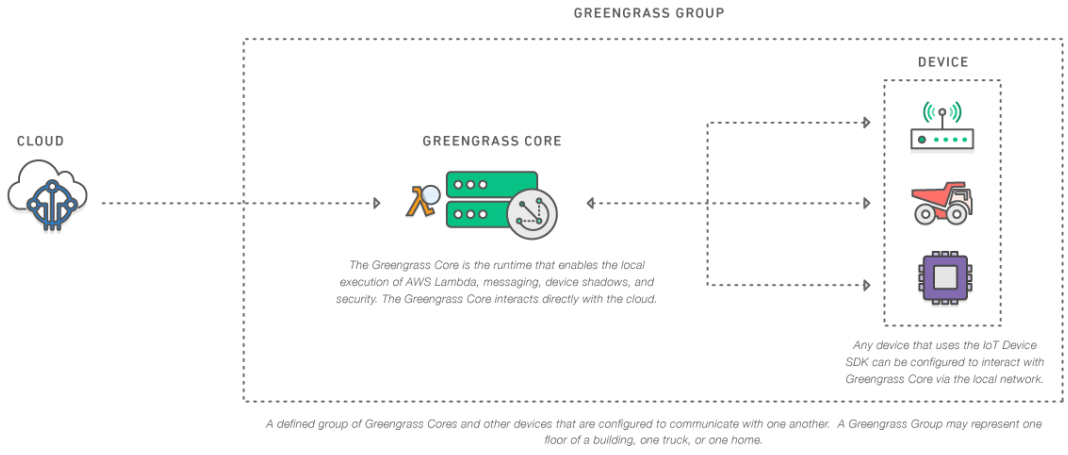

Figura 13. Arquitectura AWS Greengrass. Font [18]

En AWS Greengrass, els dispositius es comuniquen de manera segura a través d'una xarxa local i intercanvien missatges entre si sense haver de connectar-se al núvol. AWS Greengrass proporciona un administrador de missatges de publicació/envio local que pot emmagatzemar

missatges en búfer de manera intel·ligent si es perd la connectivitat, de manera que es conservin els missatges d'entrada i sortida en el núvol. [18]

AWS Greengrass protegeix les dades d'usuari [18]:

- A través de l'autorització i autenticació segures dels dispositius.
- A través de connectivitat segura a la xarxa local.
- Entre els dispositius locals i el núvol.

Les credencials de seguretat dels dispositius funcionen dins d'un grup fins que es revoquen, encara que s'interrompi la connectivitat al núvol, de manera que els dispositius puguin seguir comunicant-se de forma segura en el nivell local [18].

Amb AWS Lambda, Greengrass s'assegura que els seus dispositius de IoT poden respondre amb rapidesa a esdeveniments locals, operar amb connexions intermitents i minimitzar el cost de transmetre dades de IoT al núvol [19].

Un dels principals beneficis de AWS Lambda és que abstreu l'administració del servidor del professional de TI. Amb AWS Lambda, Amazon administra els servidors, la qual cosa permet que un desenvolupador es concentri més a escriure codi d'aplicació. Amb Lambda, pot executar codi per gairebé qualsevol tipus d'aplicació o servei back-end i tot sense administrar gens, solament s'ha de carregar el codi. Lambda s'encarregarà de tot el necessari per executar i escalar el codi amb alta disponibilitat. Solament es pagarà pel temps de còmput que consuma, no es cobra gens quan el codi no s'està executant [20].

Amb AWS Greengrass els dispositius no funcionen de forma aïllada entre si, sinó que és possible definir en camp una arquitectura de Gateway concentrador de dispositius, per gestionar conjunts d'ells en una xarxa local, de manera que encara que estiguin desconnectats de la plataforma AWS IoT, Greengrass proporcioni la capacitat de missatgeria MQTT (estàndard de facto de la plataforma AWS IoT) i el gateway concentrador de tots els dispositius, així com el API client del broker a la resta. [20]

Beneficis [19]:

- **Respon a esdeveniments locals en temps gairebé real:** Els dispositius de AWS Greengrass poden actuar a nivell local en funció de les dades que generen, al mateix temps que utilitzar el núvol per a tasques d'administració, anàlisi i emmagatzematge durador.
- **Opera en mode offline:** AWS Greengrass permet que els dispositius connectats operin fins i tot amb connectivitat intermitent al núvol. Una vegada que el dispositiu es reconnecta, Greengrass sincronitza els seus doneu-vos amb els de AWS IoT, proporcionant una funcionalitat excel·lent independentment de la connectivitat.
- **Comunicació segura:** AWS Greengrass autentica i xifra dades de dispositius en tots els punts de la connexió. D'aquesta manera, mai s'intercanvien dades entre dispositius i amb el núvol sense identitat demostrada.
- **Programació de dispositius simplificada sobre la base de funcions de AWS Lambda:** pel que es pot crear i provar el programari dels seus dispositius en el núvol primer i, a

continuació, implementar-ho amb facilitat en els dispositius. Greengrass permet executar funcions de Lambda a nivell local, reduint la complexitat de desenvolupar programari integrat.

- **Redueixi el cost d'executar aplicacions de IoT:** Amb AWS Greengrass, pot programar el dispositiu perquè filtri dades a nivell local i solament transmeti les dades necessàries per a les seves aplicacions al núvol.

AWS Greengrass es basa en dos components, para proporcionar l'arquitectura de Gateway concentrador de dispositius [20]:

- **Greengrass Core (GSC):** Proporciona la funcionalitat de concentrador donant les següents capacitats:
    - Execució de funcions Lambda, incloent Interacció directa amb altres serveis en AWS.
    - Seguretat integrada amb model definit en la plataforma cloud AWS IoT.
    - Gestió d'ombres de dispositiu. En AWS IoT, l'ombra del dispositiu és l'estat virtual que té un dispositiu en la plataforma en funció de l'ultimo estat reportat i les modificacions que s'hagin fet sobre ell i no estiguin confirmades amb el dispositiu físic, de manera que quan es realitzi la sincronització amb el dispositiu físic, el seu estat ha de ser consolidat amb la seva ombra en la plataforma.
    - Capacitats de broker de missatgeria MQTT en xarxa local
- **SDK de AWS IoT:** API que permet tant connectar directament amb el broker MQTT de AWS IoT, com amb el del node que fa les vegades de Greengrass Core del seu grup a la xarxa local

Funcionament [20]:

La gestió integral es fa mitjançant la plataforma cloud AWS IoT, on s'han de donar d'alta els dispositius i conjunts de dispositius que conformen un grup Greengrass. Un grup ha de tenir en el cloud de AWS IoT una configuració (definició de grup):

- El primer pas per crear un grup Greengrass és definir el dispositiu que fa les vegades de Greengrass core del grup en la seva configuració en el cloud de AWS IoT. Tots els grups necessiten tenir un Greengrass core, sent est, el dispositiu físic on s'instal·la el programari de Greengrass.
- Una vegada instal·lat el programari de Greengrass core en el dispositiu, aquest ja es pot connectar al cloud de AWS IoT.
- A continuació es contínua definint el grup en cloud en AWS IoT donant d'alta la resta de dispositius i/o les funcionis Lambda a executar en el dispositiu que té el Greengrass core, que es despleguen de forma remota en el dispositiu core.

Així que a grans trets Greengrass ens proporciona un node concentrador, el programari del qual (funcionis Lambda) i configuració s'ha fet en servidor, que proporciona un broker MQTT para centralitza el tràfic de la resta de dispositius del grup, i fa de passarel·la a AWS IoT.

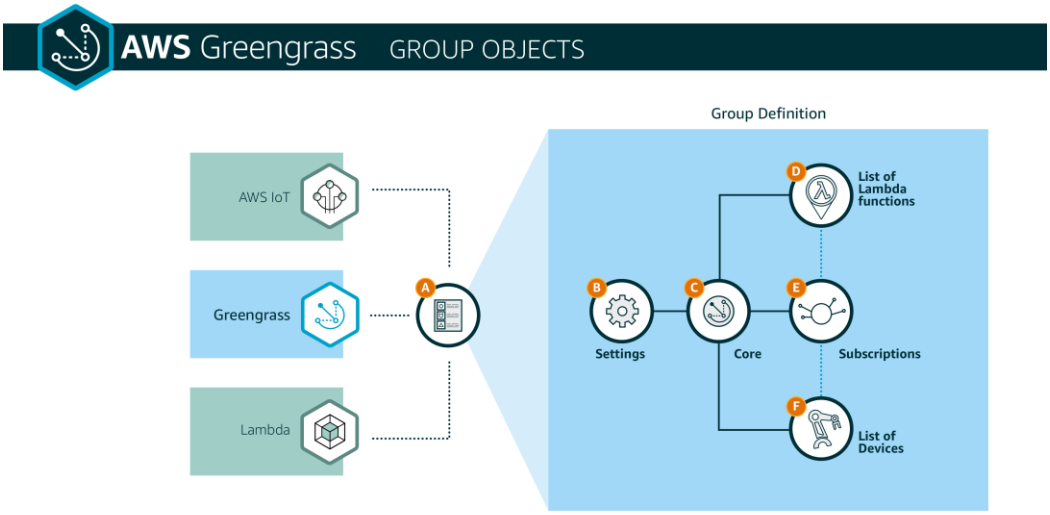

Figura 14. Grups AWS Greengrass. Font [18]

2.4.2.2 IoT Microsoft Azure

Azure és un conjunt integral de serveis en el núvol que els desenvolupadors i els professionals de TI utilitzen per crear, implementar i administrar aplicacions a través de la nostra xarxa global de centres de dades. Eines integrades, DevOps i un marketplace li ajuden a crear de manera eficaç qualsevol cosa, des d'aplicacions mòbils senzilles fins a solucions orientades a Internet. [21]

Es pot utilitzar Azure IoT Hub i els SDK de dispositiu IoT de Azure per crear solucions d'Internet de les coses (IoT) [22]:

- **Azure IoT Hub** és un servei completament administrat en el núvol que connecta, supervisa i administra els dispositius IoT de forma segura. S'utilitzen els SDK de dispositiu IoT de Azure per implementar els dispositius de IoT.
- S'utilitza una porta d'enllaç de IoT en escenaris IoT més complexos. Per exemple, on sigui necessari tenir en compte factors com a dispositius antics, costos d'ample de banda, directives de seguretat i privadesa o processament de dades perimetrals. En aquests escenaris, s'usa **Azure IoT Edge** per implementar una porta d'enllaç que connecta dispositius amb IoT Hub.

2.4.2.2.1 IoT Hub

IoT Hub permet connectar els seus dispositius IoT a Microsoft Azure de forma senzilla i segura. Pot establir comunicació bidireccional amb tots els seus dispositius IoT, treballar amb plataformes i protocols que ja coneix i actualitzar i administrar dispositius a escala i de forma fiable amb nova funcionalitat d'administració de dispositius. L'autenticació en IoT Hub es fa per dispositiu, de manera que la seva solució de IoT es configura per mantenir la confidencialitat dels missatges que s'envien del núvol als dispositius i viceversa. [21]

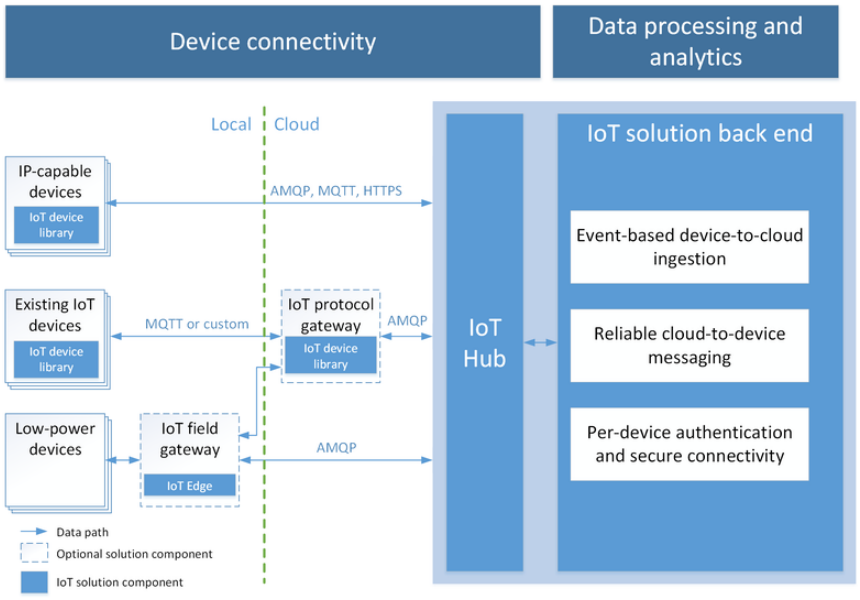

Figura 15. Arquitectura IoT Hub. Font [22]

Azure IoT Hub ofereix un gran conjunt d'opcions de comunicació de dispositiu a núvol i de núvol a dispositiu. A més, Azure IoT Hub afronta els problemes que poden aparèixer en connectar de forma confiable i segura als dispositius de les següents formes [22]:

- **Dispositius bessons**: Amb els dispositius bessons pot emmagatzemar, sincronitzar i consultar metadades i informació d'estat dels dispositius. Els dispositius bessons són documents JSON que emmagatzemen informació sobre l'estat del dispositiu com a metadades, configuracions i condicions. IoT Hub manté un dispositiu bessó per cada dispositiu que es connecta a IoT Hub.
- **Autenticació per dispositiu i connectivitat segura:** Pot aprovisionar cada dispositiu amb la seva pròpia clau de seguretat per permetre-li connectar-se a IoT Hub. El registre d'identitats de IoT Hub emmagatzema identitats i claus en una solució. Un back-end de solucions pot agregar dispositius individuals per permetre o denegar llistes que permetin controlar per complet l'accés als dispositius.
- **Enrutamiento de missatges de dispositiu a núvol a serveis de Azure segons regles declaratives:** IoT Hub permet definir rutes de missatges basades en regles de enrutamiento per controlar on el centre envia missatges de dispositiu a núvol. Les regles de enrutamiento no requereixen que s'escrigui cap codi i poden substituir als distribuïdors de missatges personalitzats posteriors a la ingesta.
- **Supervisió d'operacions de connectivitat del dispositiu:** Pot rebre registres d'operació detallats sobre operacions d'administració d'identitat de dispositius i esdeveniments de connectivitat de dispositius. Aquesta funcionalitat de supervisió permet a una solució de IoT identificar problemes de connectivitat. Aquests registres s'usen per identificar els dispositius que proporcionen credencials incorrectes, envien missatges amb massa freqüència o rebutgen tots els missatges del núvol al dispositiu.
- **Ampli conjunt de biblioteques de dispositius:** Els SDK de dispositiu IoT de Azure estan disponibles i són compatibles amb diversos llenguatges i plataformes: C per a moltes

distribucions de Linux, Windows i sistemes operatius en temps real. Els SDK de dispositius IoT de Azure admeten llenguatges administrats com a C#, Java i Javascript.

- **Extensibilitat i protocols de IoT:** Si la solució no pot usar les biblioteques de dispositius, IoT Hub exposa un protocol públic que permet als dispositius usar els protocols MQTT v3.1.1, HTTPS 1.1 o AMQP 1.0 de forma nativa. IoT Hub també es pot ampliar perquè admeti protocols personalitzats mitjançant la:
    - Creació d'una porta d'enllaç de camp amb Azure IoT Edge que converteix el seu protocol personalitzat en un dels tres protocols compatibles amb IoT Hub.
    - Personalització de la porta d'enllaç de protocol de IoT de Azure, un component de codi obert que s'executa en el núvol.
- **Escala:** El centre de IoT de Azure es pot escalar a milions de dispositius connectats de manera simultània i a milions d'esdeveniments per segon.

Azure IoT Hub implementa el model de comunicació assistida per servei per intervenir en les interaccions entre els dispositius i el seu back-end de solucions. L'objectiu de la comunicació assistida per servei és establir rutes d'accés de comunicació bidireccional de confiança entre un sistema de control, com IoT Hub, i els dispositius d'ús especial en un espai físic que no és de confiança. El patró estableix els principis següents:

- La seguretat té prioritat sobre la resta de les funcions.
- Els dispositius no accepten informació de xarxes no sol·licitades. Un dispositiu estableix totes les connexions i les rutes en manera de solament sortida. Perquè un dispositiu rebi comandos des del back-*end de la solució, dita dispositiva ha d'iniciar una connexió amb regularitat per comprovar si hi ha comandos pendents de processar.
- Els dispositius solament han de connectar-se o establir rutes a serveis coneguts als quals estan aparellats, com un Centre de IoT.
- La ruta de comunicació entre el dispositiu i el servei o el dispositiu i porta d'enllaç es protegeix en el nivell del protocol d'aplicacions.
- L'autenticació i l'autorització de nivell de sistema es basen en les identitats de cada dispositiu. Fan els permisos i les credencials d'accés revocables gairebé a l'instant.
- En el cas dels dispositius que es connecten esporàdicament a causa de problemes d'alimentació o de connectivitat, la comunicació bidireccional funciona mantenint els comandos i les notificacions fins que un dispositiu es connecta per rebre'ls. El Centre de IoT manté cues específiques de dispositius per als comandos que envia.
- Les dades de càrrega d'aplicacions es protegeixen per separat per protegir el trànsit a través de les portes d'enllaç a un servei determinat.

2.4.2.2.2 IoT Edge

IoT Edge és una solució de Iot híbrida perimetral i en el núvol, la qual permet organitzar fàcilment el codi amb els serveis, de manera que flueixin de forma segura entre el núvol i el perímetre per poder distribuir intel·ligència a una àmplia varietat de dispositius. IoT Edge obre un món de possibilitats, permet habilitar intel·ligència artificial i anàlisi avançada en el perímetre, reduir els costos de la seva solució de IoT i controlar els dispositius perifèrics sense connexió o amb connectivitat intermitent. [21]

És un servei d'Internet de les coses (IoT) que es basa en IoT Hub, el qual està pensat per als clients que desitgen analitzar dades en dispositius, situació també coneguda com "en el perímetre", en lloc d'en el núvol. En moure elements de la càrrega de treball al perímetre, els dispositius poden dedicar menys temps a enviar missatges al núvol i reaccionen més ràpidament als canvis d'estat.

Les anàlisis són un valor afegit empresarial per a les solucions de IoT, però no és necessari que totes les anàlisis estiguin en el núvol. Si desitja que un dispositiu respongui a emergències al més aviat possible, pot realitzar la detecció d'anomalies en el propi dispositiu. Anàlogament, si desitja reduir els costos d'ample de banda i evitar la transferència de terabytes de dades sense processar, pot realitzar la neteja i l'agregació de dades localment. A continuació, enviï la informació al núvol. [22]

Azure IoT Edge està format per tres components [22]:

1. **Els mòduls de IoT Edge** són contenidors que executen serveis de Azure, de tercers o codi propi de l'usuari. S'implementen en els dispositius de IoT Edge i executen la lògica de negocis en els dispositius perimetrals.
2. **L'entorn de temps d'execució de IoT Edge** s'executa en tots els dispositius de IoT Edge i administra els mòduls que s'implementen en cada dispositiu. Realitza diverses funcions:
    - Instal·la i actualitza les càrregues de treball en el dispositiu.
    - Manté els estàndards de seguretat de Azure IoT Edge en el dispositiu.
    - Garanteix que els mòduls de IoT Edge estan sempre en execució.
    - Informa de l'estat del mòdul al núvol per a la supervisió remota.
    - Facilita la comunicació entre els dispositius de fulla de nivell inferior i el dispositiu de IoT Edge.
    - Facilita la comunicació entre els mòduls i el dispositiu de IoT Edge.
    - Facilita la comunicació entre el dispositiu de IoT Edge i el núvol.

    Com usar els dispositius de Azure IoT Edge és decisió seva. L'entorn de temps d'execució sovint s'usa per implementar intel·ligència artificial en portes d'enllaç que agreguen i processen dades d'altres dispositius locals, però això és solament una de les opcions disponibles. Els dispositius de fulla també poden ser dispositius de Azure IoT Edge, independentment de si estan connectats a una porta d'enllaç o directament al núvol.

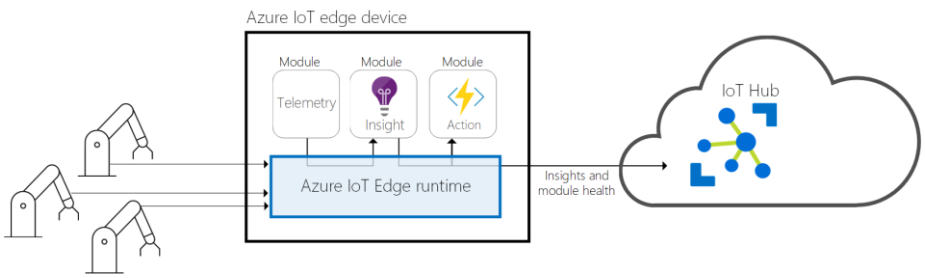

Figura 16. Entorn de temps d'execució de IoT Edge

3. **Una interfície basada en el núvol** permet supervisar i administrar els dispositius de IoT Edge de forma remota. Azure IoT Edge s'integra perfectament amb Conjunt d'aplicacions de IoT de Azure per aconseguir un pla de control que s'adapta a les necessitats de la solució. Els serveis en el núvol permeten als usuaris:
    - Crear i configurar una càrrega de treball que s'executi en un tipus específic de dispositiu.
    - Enviar una càrrega de treball a un conjunt de dispositius.
    - Supervisar les càrregues de treball que s'executen en els dispositius de camp.

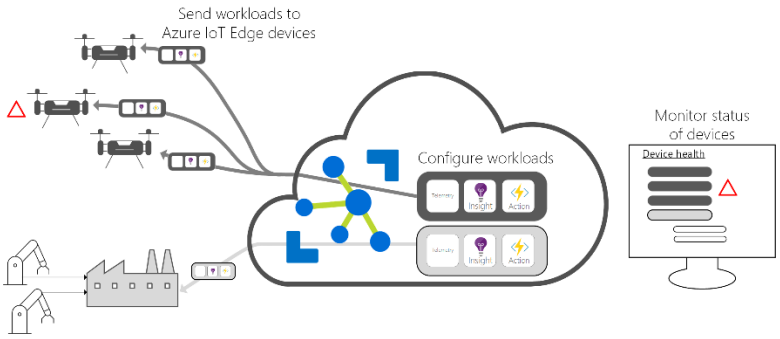

Figura 17. Interficie en el núvol de IoT Edge

Permet l'anàlisi i el processament de dades en dispositius perimetrals. Amb IoT Edge es pot permetre que els dispositius amb codi de contenidor incloguin qualsevol lògica extreta directament des dels serveis de Azure que ja utilitzi o des del seu propi codi específic de la solució. Permet als dispositius [22]:

- Actuar com a dispositius de porta d'enllaç, i agregar i processar dades des de diversos dispositius de fulla.
- Realitzar la detecció d'anomalies i reaccionar als canvis en l'entorn sense haver d'esperar instruccions del núvol.
- Minimitzar els costos d'ample de banda i emmagatzematge mitjançant el processament de les dades i l'enviament dels resultats.

IoT Edge també inclou una interfície en el núvol que permet l'administració remota de dispositius. Sense haver d'accedir físicament als dispositius, pot implementar codi, i supervisar l'estat i actualitzar-ho. Pot administrar dispositius únics o crear implementacions que afectin a grans conjunts de dispositius que defineixi de manera remota. [22]

2.4.2.3 Comparació de AWS IoT i Microsoft Azure

AWS i Azure ofereixen rangs similars en capacitat de computació, emmagatzematge i xarxes, i serveis IoT per a control de dispositius i maneig de dades, la Taula 2. Comparació de AWS IoT i Microsoft Azure. Font

mostra la comparació entre les dues principals plataformes de IoT.

	AMAZON AWS IOT	MICROSOFT AZURE IOT HUB
PROTOCOLS	HTTP, MQTT	HTTP, AMQP, MQTT i protocols personalitzats
PATRONS DE COMUNICACIÓ	Telemetria, Comandes (canvi d'estat)	Telemetria, Comandes
PLATAFORMES DE DISPOSITIUS CERTICADES	Broadcom, Marvell, Renesas, Texas Instruments, Microchip, Intel, Mediatek, Qualcomm, Seeed, BeagleBoard	Intel, Raspberry Pi 2, Freescale, Texas Instruments, MinnowBoard, BeagleBoard, Seeed, resin.io
SDK / LANGUATGES	C, NodeJS	.Net and UWP, Java, C, NodeJS
SEGURETAT	TLS (Autentificació mutua)	TLS (Només autentificació de servidor)
AUTENTIFICACIÓ	Autenticació del client de certificat X.509, servei IAM, servei Cognito	Per dispositiu amb el token de SAS (*Shared Access SIgnature*)

Taula 2. Comparació de AWS IoT i Microsoft Azure. Font [23]

Les dues plataformes utilitzen la els protocols de comunicació comunament utilitzats com són HTTP i MQTT

Els mecanismes per proporcionar el patró de "telemetria" podrien considerar-se bastant similars, però el mecanisme per al patró de "comanda" amb un camí de sol·licitud/resposta és bastant diferent; és una funció incorporada per AMQP, però necessita una capa addicional damunt de MQTT.

La interacció amb els dispositius és diferent, es basa en intercanvis de missatges en IoT Hub i més estrictament relacionat amb el concepte d'estat del dispositiu en AWS IoT gràcies a les ombres de coses. En el primer costat tenim un missatge que pot portar dades d'aplicacions i comandes per a accions; en l'altre costat, tenim la representació de l'estat a través d'un document JSON (per descomptat, dins d'un missatge MQTT). És com tenir una capa més d'abstracció de dades en AWS IoT.

Les implementacions de seguretat són bastant similars tenint en compte l'ús del protocol TLS i la identitat es maneja d'una manera diferent però amb gairebé els mateixos resultats.

En quant al maquinari i els SDK, gràcies al suport de llenguatge C per a ambdues plataformes IoT, tenim solucions bastant similars. [23]

2.4.2.4 Altres plataformes

Tot i que les plataformes comercials més utilitzades a nivell mundial siguin Amazon Web Service (AWS) i Microsoft Azure cada cop hi han moltes més opcions, com podria ser Google Cloud, IBM Cloud, Oracle Cloud, etc. A la Figura 18 es pot veure les estadístiques de les plataformes més utilitzades actualment en comparació a l'any passat.

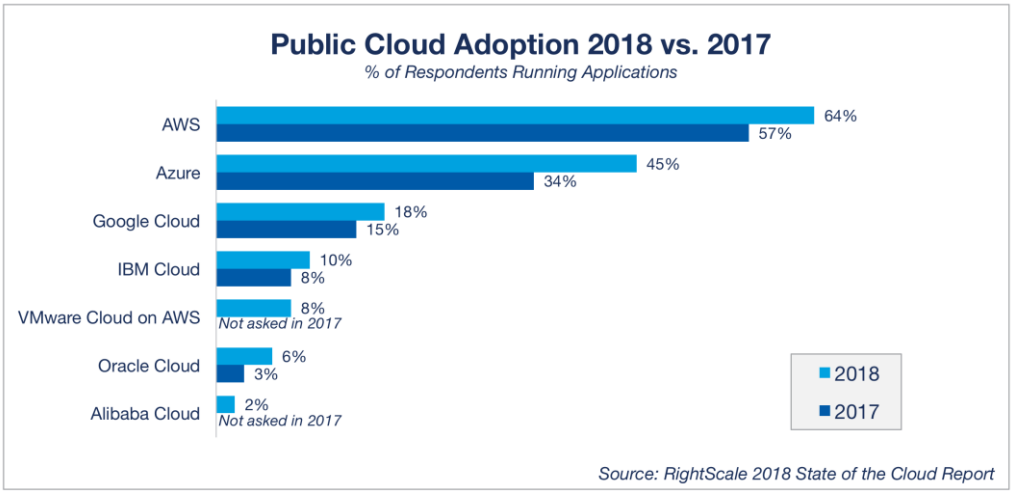

Figura 18. Ranking plataformas de *Cloud Computing* [24]

2.5 Altres tecnologies implementades a l'escenari

En la implementació de l'escenari WSN s'han utilitzat algunes tecnologies més de les mencionades fins ara. S'ha decidit realitzar amb les motes Z1 de Zolertia, implementant-los el sistema operatiu Contiki i utilitzant el protocol RPL per a la seva comunicació. A continuació es detallaran les característiques que formen aquest escenari.

2.5.1 Mota de WSN - Z1

Els recents avanços tecnològics en àrees com a informàtica, xarxes, detecció i comunicació sense fil, ara permeten un nou paradigma on els objectes i els entorns poden interactuar de forma fluïda per monitorejar i controlar els entorns. Ha arribat el moment dels objectes intel·ligents en entorns intel·ligents, [25] com es el cas del mòduls de la empresa Zolertia.

Zolertia és una empresa espanyola especialitzada en el disseny i desenvolupament de solucions hardware per a la creació de aplicacions IoT. Ofereix una línia de diferents plataformes IoT per a prototips i dispositius connectats que poden convertir-se fàcilment en un producte final real en un curt període de temps, totes aquestes característiques gràcies al seu mòdul principal, The Zoul. [26]

Les motes Z1 són un mòduls sense fils de baixa potència que compleixen amb els protocols IEEE 802.15.4 i Zigbee destinats a ajudar als desenvolupadors de WSN a provar i desplegar les seves

pròpies aplicacions i prototips amb la millor compensació entre el temps de desenvolupament i la flexibilitat del hardware. Z1 te com a propòsit general ser una plataforma de desenvolupament per a xarxes sense fils de sensors (WSN) dissenyada per a investigadors, desenvolupadors, entusiastes i aficionats. [25]

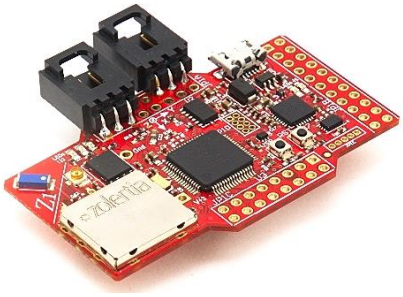

Figura 19. Mota Z1 de Zolertia. Font [26]

La plataforma Z1 té les següents principals característiques [27]:

- **MCU de ultra-baixa potència i transceptor de 2.4GHz:** Z1 està equipat amb un microcontrolador de segona generació MSP430F2617 de baixa potència, que compta amb una potent 16 bits RISC CPU @16MHz de velocitat de rellotge, calibratge de fàbrica incorporada del rellotge, 8KB de RAM i 92KB de memòria Flaix. També inclou el ben conegut transceptor CC2420, compatible tant IEEE 802.15.4 com amb 6LoWPAN i ZigBee, que opera a 2,4 GHz amb una velocitat de dades efectiva de 250 Kbps. La selecció de *hardware* Z1 garanteix la màxima eficiència i robustesa amb un baix cost d'energia.
- **2 sensors digitals incorporats:** Z1 ve amb sensors digitals incorporats llests per funcionar: un acceleròmetre digital programable (ADXL345) i un sensor de temperatura digital (TMP102) estan en la placa principal.
- **Fins a 4 sensors externs:** Z1 és compatible amb Phidgets i molts altres sensors analògics i digitals.
- **Programació per USB:** Z1 no requereix cap maquinari extern per ser programat, la capacitat completa d'USB incorporada permet el desenvolupament ràpid d'aplicacions WSN i una ràpida integració amb múltiples sistemes.
- **Alimentació flexible:** La placa es pot alimentar mitjançant un paquet de bateria (2xAA o 2xAAA), una moneda (fins a 3,6 V), alimentació USB, connexió directa a través de dos cables procedents d'una font d'alimentació.
- **Compatibilitat amb diferents sistemes operatius**: TinyOS i Contiki.

2.5.2 Sistema operatiu Contiki

Contiki és un sistema operatiu de codi lliure, portable i multitasca per a la IoT. Està escrit en llenguatge C i utilitza el concepte de *Protothread,* la qual cosa ho caracteritza amb un ús molt reduït de la memòria del microprocessador en comparació d'altres arquitectures multitasca. Està enfocat per usar-se en sistemes senzills i encastats sobre microcontroladors, com poden ser els nodes d'una xarxa de sensors.

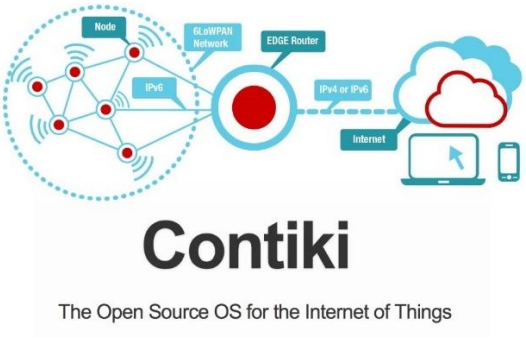

Figura 20. Contiki

La idea d'aquest sistema operatiu és tenir diversos fils d'execució en paral·lel, els quals esperen certes condicions per activar-se. Els fils en activar-se realitzen les seves tasques pertinents i en acabar-les queden novament en espera fins que un altre esdeveniment els activi. Els esdeveniments són els que s'encarreguen d'activar als fils. Aquests es guarden en una cua circular i es processen per ordre cronològic. Cada esdeveniment generat té un fil associat per despertar, al que li pot enviar informació quan ho desperta.

La distribució de Contiki empleada inclou els fitxers del kernel de sistema, els diferents microcontrolodores suportats i les adaptacions de les diferents llibreries segons la plataforma maquinari que s'empri. Està basat en protocols i estàndards com IPv4, IPv6, 6lowpan, RPL i CoAP. Addicionalment inclou diverses utilitats i implementacions de protocols per operar a nivell d'aplicació (CoAP, FTP, Telnet, etc.) en les quals l'usuari podrà recolzar-se per al desenvolupament de les seves solucions. A tot això, cal unir les eines necessàries per a la compilació tant del propi sistema operatiu com dels programes desenvolupats per l'usuari. [28]

2.5.3 Protocol d'enrutament RPL

RPL (*Routing Protocol for Low-Power and Lossy Networks*) és un protocol d'enrutament per a xarxes sense fils de baix consum d'energia com WSN i típicament susceptibles a pèrdues de paquets. És un protocol proactiu basat vectors de distància, opera en la capa d'enrutament, per sobre de IEEE 802.15.4, i és responsable de la transmissió de paquets a través de múltiples salts que separen els nodes origen i destí. La seva funció es divideix en dues parts, per una banda, disposa d'un motor de reenviament que utilitza una taula d'enrutament per decidir quin node veí és el següent salt per al paquet. D'altra banda, un protocol d'encaminament que omple les dades de la taula d'enrutament descrita anteriorment. [29]

Està optimitzat per xarxes de recollida amb comunicació poc freqüent des del punt de recollida als nodes, on els nodes envien les dades capturades periòdicament per un nombre petit de punts de recollida. RPL suporta trànsit de recollida *Multi-Point-to-Point* (MP2P) i trànsit de configuració *Point-To-Multi-Point* (P2MP). No suporta bé el trànsit *Point-To-Point* (P2P), encara que es poden desenvolupar solucions en aquest sentit. [30]

Per al trànsit MP2P, RPL construeix un gradient a la xarxa, amb connexió a terra en els punts de recollida anomenat *Low power and lossy network Border Router* (LBR). Cada node dins de la xarxa té un rang assignat (Rank), el qual augmenta a mesura que els equips s'allunyen del node LBR. El gradient en RPL s'anomenat *Destination Oriented Directed Acyclic Graph* (DODAG). Els nodes reenvien paquets utilitzant com a criteri de selecció de ruta, aquella amb el rang més baix. [30]

Per mantenir l'estat de la xarxa, evitar rutes cícliques, anunciar l'existència del *Directed Acyclic Graph* (DAG) a altres equips i permetre la seva incorporació, es defineixen tres tipus de paquets ICPMv6 [31]:

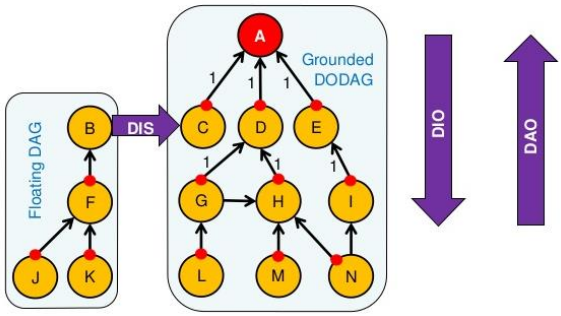

Figura 21. Missatges ICMPv6 RPL: DIO, DAO y DIS. Font [31]

DIS (*DODAG Information Solicitation*): utilitzat per sol·licitar informació de DODAG propers, anàleg a missatges de sol·licitud de routers utilitzats per descobrir xarxes existents. S'utilitza per afegir nous nodes al DODAG.

DIO (DAG *Information Object*): transporta informació que permet conèixer al node la instancia de RPL, els seus paràmetres de configuració, seleccionar un conjunt DODAG "pare" i mantindre el DODAG. Es l'encarregat de la configuració i manteniment de la xarxa.

DAO (*Destination Advertisement Object*): enviat en adreça cap al DODAG, és un missatge enviat pels equips per actualitzar la informació dels seus nodes "pares" al llarg del DAG.

Les possibles mètriques i limitacions d'aquest protocol d'encaminament són: energia del node, nombre de salts, rendiment de l'enllaç, latència, fiabilitat de l'enllaç i color de l'enllaç. El terme color de l'enllaç en RPL es refereix a les propietats dels enllaços, en funció del color es poden utilitzar o no els enllaços pels camins. [31]

2.5.4 Protocol UDP

UDP (*User Data Protocol*) és un protocol de nivell 4 de la torre OSI (transport) que ofereix un model molt simple per a transmissions no orientades a connexió. UDP no realitza cap diàleg handshake entre l'emissor i receptor, exposant així qualsevol falta de fiabilitat del protocol de xarxa subjacent al programa de l'usuari. No existeix cap garantia de lliurament, sol·licitud o duplicat de paquets.

UDP és adequat per a finalitats en els quals la comprovació i correcció d'errors no és necessària o es realitza en l'aplicació, evitant la sobrecarregués d'aquest processament en el nivell de xarxa. Les aplicacions time-sensitive usen UDP perquè és preferible deixar paquets a esperar paquets retardats. En les WSN, UDP suposa una menor quantitat de transaccions, alhora que aquestes són de menor pes, la qual cosa es tradueix en un menor consum d'energia.

Les transaccions UDP requereixen d'un model servidor-client on la comunicació es realitza usant un socket, és a dir, mitjançant una adreça IP i un port.

3 Disseny

En aquest apartat s'explica el disseny que s'ha realitzat per crear el sistema de la xarxa de sensors per al control de temperatura d'un edifici de manera remota. Hi ha dos dissenys: un primer on s'utilitza la tecnologia ICN i un altre on s'afegeix a més la tecnologia DTN.

3.1 Disseny implementació ICN

Amb tota la informació recollida a l'apartat de l'estat de l'art, s'han analitzat tots els aspectes en relació a aquest escenari i s'han decidit les següents configuracions per al disseny de la seva implementació:

Per realitzar el disseny de l'arquitectura ens basem en l'arquitectura del sistema ICN-IoT, que em vist en l'anterior apartar, de manera que realitzarem la implementació dels següents mòduls que podem veure a la Figura 22: [11]

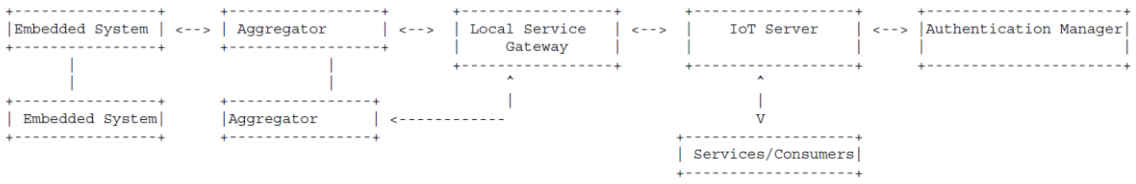

Figura 22. Model d'arquitectura. Font [11]

- ***Embedded System* (ES)**: el sensor integrat té funcions de detecció i activació i també pot transmetre dades per a altres sensors al Agregador, a través d'enllaços sense fils o per cable.
- **Agregador:** interconnecta diverses entitats en una xarxa local de IoT. Els agregadors tenen les següents funcions: descobriment de dispositius, descobriment de serveis i assignació de noms. Els agregadors poden comunicar-se entre ells directament o a través de la porta d'enllaç del servei local.
- ***Local Service Gateway* (LSG):** un LSG compleix les següents funcions: (1) es troba en el límit administratiu, com una casa o una empresa, que connecta el sistema de IoT local amb la resta del sistema de IoT global (2) serveix per assignar noms de ICN a sensors locals, (3) aplica polítiques d'accés a dades per a dispositius de IoT locals i (4) executa serveis de processament de context per generar informació especificada per contextos específics de l'aplicació (en lloc de dades sense processar) per a Servidor IoT.
- **Servidor IoT:** dins d'un context de servei IoT donat, el servidor IoT és un servidor centralitzat que manté subscripció d'afiliats i proporciona el servei de cerca per a subscriptors. A diferència dels servidors IoT heretats que estan involucrats en la ruta de dades dels editors als subscriptors, la qual cosa augmenta la preocupació que les seves interfícies siguin un coll d'ampolla, el servidor de IoT en aquesta arquitectura solament participa en la ruta de control on els editors i subscriptors intercanvien els seus noms, certificats i imposen altres funcions de seguretat com el control d'accés.

- ***Authentication Manager* (AM):** l'administrador d'autenticació serveix per habilitar l'autenticació dels dispositius integrats quan s'incorporen a la xarxa i també si la seva identitat necessita ser validada en el nivell general del sistema. L'administrador d'autenticació pot ser co-resident amb el LSG o el servidor de IoT, però també pot ser independent dins del límit administratiu o fora d'ell.
- **Serveis/Consumidor:** són altres instàncies de l'aplicació que interactuen amb el servidor IoT per buscar o ser notificat de qualsevol cosa d'interès dins de l'abast del servei IoT.

Per a la implementació d'aquesta arquitectura realitzarem algunes simplificacions. Per una banda el mòdul ES és el que formen les motes de Zolertia, una de les motes estarà connectada a l'ordinador i farà la funció de Gateway. D'altre banda agruparem en un sol mòdul els mòduls de l'agregador i del LSG, i per últim obviarem el mòdul AM, tal i com mostra la Figura 23.

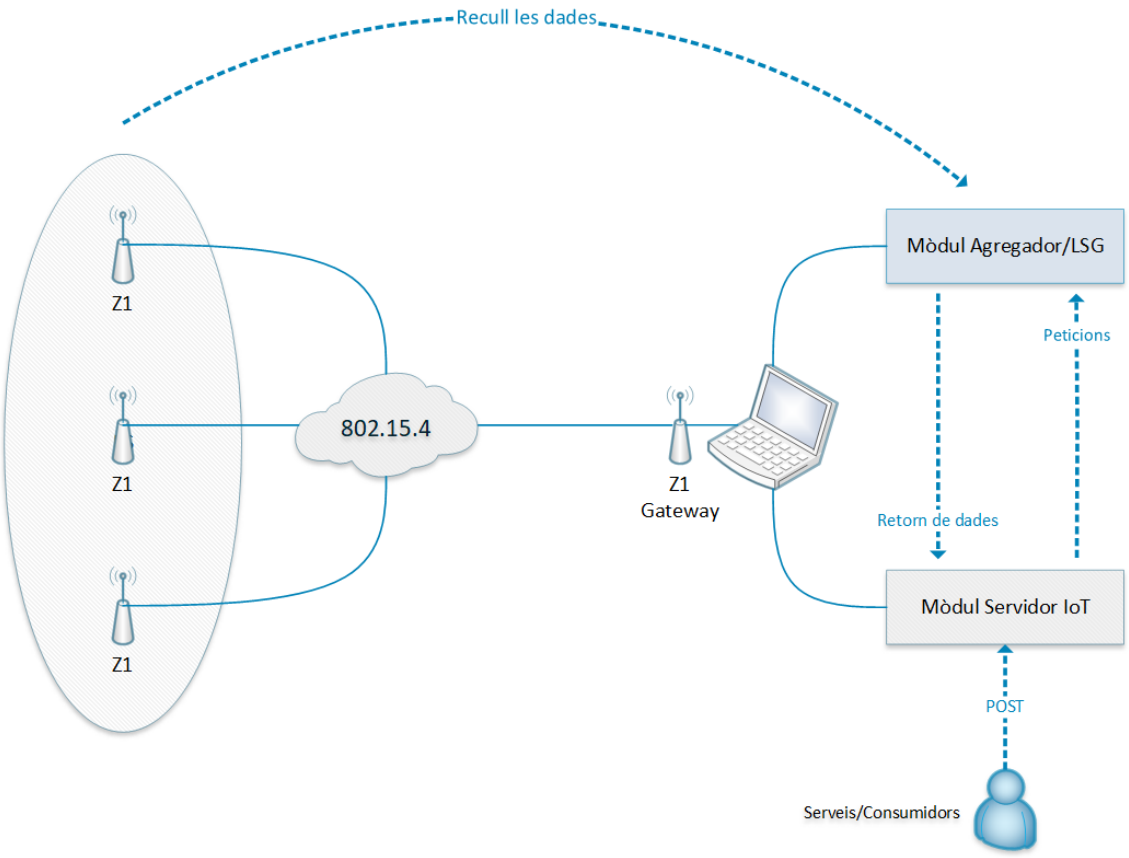

Figura 23. Disseny implementació d'arquitectura

El funcionament seria el següent:

- El mòdul agregador/LSG rebrà les dades de la xarxa IoT que formen les motes Z1 a través de la mota que fa de Gateway, de manera periòdica. Serà l'encarregat d'assignar noms ICN a cadascun del nodes dels quals rep informació.
- Quan el mòdul servidor IoT rebi un post amb peticions provinents del consumidors, aquest enviarà les peticions al mòdul agregador/LSG amb el nom del node del qual desitja la informació i aquest últim contestarà a les peticions amb les dades sol·licitades, les quals han sigut recollides anteriorment de les motes.

Per implementar aquesta arquitectura per al control intel·ligent de temperatura d'un edifici, s'haurà de replicar aquest mòdul representat a la Figura 22 a cadascuna de les plantes. La idea és tenir els sensors distribuïts per les diferents localitzacions que es vulguin controlar de cada planta, de manera que aquest enviïn les dades al Gateway que hi hagi a la seva mateixa planta, tal i com mostra la Figura 24.

La comunicació ICN es produirà entre els mòduls agregadors/LSG, d'aquesta manera l'agregador podrà obtenir dades tan de la seva planta com de qualsevol altre, per tant si no trobés les dades podria replicar la petició a un altre mòdul agregador/LSG.

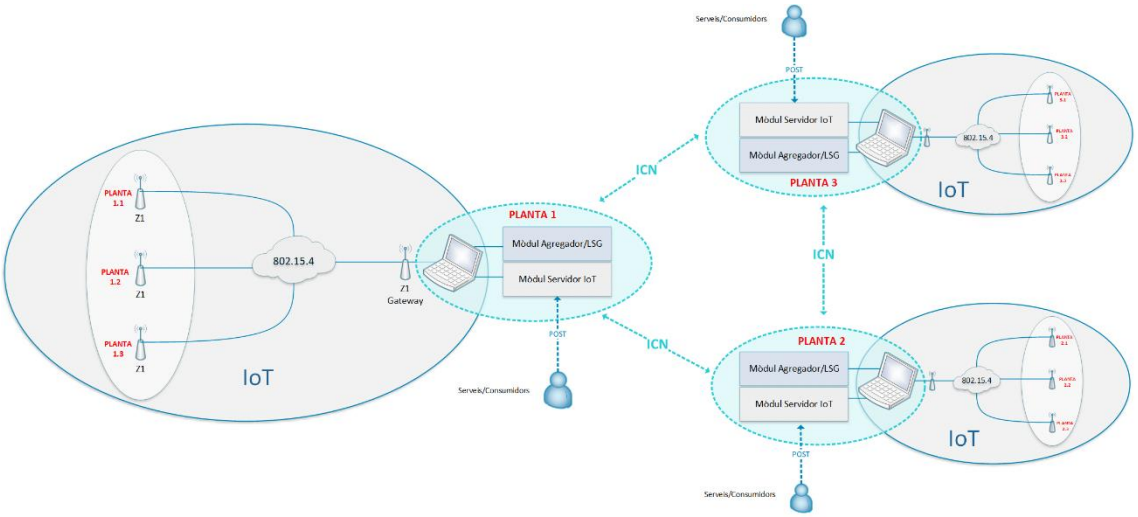

Figura 24. Comunicació ICN de l'arquitectura

Tenint en compte aquest configuració, el procediment de sol·licitud de les dades seria el següent:

1. El consumidor sol·licitarà de quina planta i localització desitja la temperatura amb el seu ID (#Planta.#Localització) a un mòdul servidor IoT.
2. El mòdul servidor IoT transmetrà la petició al mòdul agregador/LSG del seu mòdul:
   2.1. Si la petició és de la seva mateixa planta, buscarà la dada de la localització demanada i li enviarà la resposta al mòdul servidor IoT.
   2.2. Si la petició no es de la seva mateixa planta, primer buscarà que no estigui emmagatzemada d'alguna sol·licitud anterior. En el cas de que no estigués, la

sol·licitarà al mòdul agregador/LSG de la planta on estigui la dada, quan aquest li hagi retornat la dada ell la podrà reenviar al mòdul servidor IoT. A més, s'emmagatzemarà la dada per si hi han posteriors sol·licituds d'aquesta mateixa dada i així no haver-la de tornar a buscar, tot i que la dada només serà valida durant 30 minuts.

L'espai d'emmagatzematge del mòdul agregador és limitat i no cal mantenir dades més temps del necessari, per lo que s'ha hagut de pensar en una política de reemplaçament que s'adaptés a les necessitats d'aquest cas en concret. A l'estat de l'art s'ha parlat de diferents polítiques de reemplaçament, però en aquest cas concret és molt més fàcil que qualsevol d'aquelles on es té en compte l'ús que s'ha fet de les dades (LRU, LFU, etc.).

Ens hem basat simplement en la lògica de les dades recollides, tenint en compte que només interessa la dada més actualitzada de cada localització, quan es rebi una dada nova d'una mota es pot esborrar l'anterior dada d'aquella mateixa mota automàticament.

En el cas de les dades emmagatzemades que s'han sol·licitat d'un altre mòdul agregador/LSG, és a dir d'una altre planta, passaria el mateix i quan es rebés una dada més actualitzada s'esborraria l'antiga. S'ha fixat que el temps màxim per a que una dada tingui una validesa de 30 minuts, per lo tant si es sol·licités una dada d'una localització que ja estigués emmagatzemada durant més temps del fixat s'hauria de tornar a sol·licitar i per lo tant al rebre la nova dada s'esborraria l'antiga.

3.2 Disseny implementació DTN

L'escenari del control climatològic d'un edifici tal i com s'ha plantejat amb la tecnologia ICN, es pot anar un pas més enllà afegint motes mòbils, poden ser un dispositiu tipus "*Roomba*" on s'hagi instal·lat un mota de les fixes, les quals anessin desplaçant-se per l'edifici i poguessin recollir temperatura de qualsevol lloc. Aquest canvi podria ocasionar una problemàtica, ja que al anar-se'n desplaçant contínuament les motes, hi hauria més risc de que la connexió amb el Gateway es perdés i per tant també es perdessin les dades recollides.

La tecnologia DTN donaria la solució a aquesta problemàtica ja que com s'ha explicat a l'estat de l'art, DTN és una arquitectura de xarxa amb tolerància al retard dissenyada per treballar en entorns sense connectivitat extrem a extrem, amb grans retards en la comunicació.

Tal i com s'explica a l'estat de l'art, DTN utilitza mecanismes de *store-carry and forward* que permeten a un node emmagatzemar els missatges mentre no hi ha comunicació i lliurar-los en quan es produeix el contacte amb altres nodes. De manera que el canvi respecte l'anterior disseny seria que les motes emmagatzemen les dades que van recollint per allà on passen, fins que trobin al seu abast un Gateway per poder-li enviar les dades recollides, d'aquesta manera es podria evitar la pèrdua d'aquestes dades que s'han recollit mentrestant no hi havia connexió amb cap node Gateway. Les motes mòbils han de mantenir les dades emmagatzemades fins que

rebin resposta de que el Gateway les ha rebut correctament per evitar que no hi hagin pèrdues d'informació.

Aquesta ampliació en el sistema permet actuar de forma distribuïda i no depenent d'un node central, d'aquesta manera augmenta la resiliència de la solució.

4 Implementació disseny

En aquest apartat s'explica les diferents parts per poder realitzar la implementació del disseny del sistema que s'ha explicat en l'apartat anterior. Tot i que un principi es tenia pensat implementar les dues tecnologies estudiades, ICN i DTN, finalment la part de DTN va quedar fora de l'abast del projecte, ja que l'altre part implicava un volum de feina superior a l'esperat.

4.1 Xarxa WSN de motes Z1

La xarxa WSN de motes Z1 que hi haurà a cada pis tindrà la mateixa configuració. Cada pis tindrà una mota que farà de Gateway, la qual rebrà tota la informació que li enviïn la resta de motes de la xarxa, les quals enviaran cada un cert temps la temperatura que hagin registrat en aquell moment, tal i com es pot veure representat a la Figura 25.

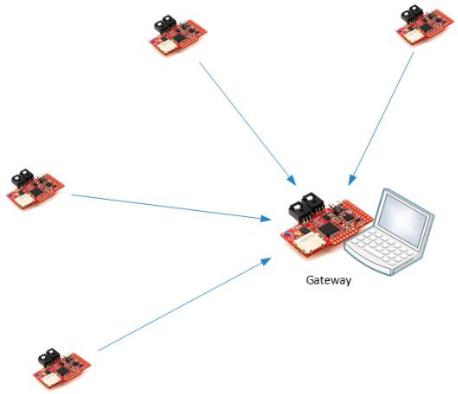

Figura 25. Xarxa WSN de motes Z1

Per a la configuració de les motes s'ha utilitzat el sistema operatiu Contiki, ja que com s'ha explicat a l'estat de l'art és compatible amb les motes de Zolertia Z1 i a més hi ha una gran quantitat d'exemples de codi a aplicar.

El codi de les motes està realitzat amb programació en C, i està basat en el que hi ha d'exemples a la plataforma de Contiki amb petites modificacions per poder-ho adaptar a les nostres necessitats, el qual es troba detallat a l'Annex 1. Per a les motes que fan de Gateway s'ha utilitzat el codi unicast-receiver.c [32], el qual permet rebre les dades que envien les altres motes a través del protocol d'enrutament RPL i al protocol a nivell de transport UDP. I en per a les motes de la xarxa que envien les dades al Gateway s'ha utilitzat el codi unicast-sender.c [31], el qual permet enviar cada cert temps la temperatura que s'hagi mesurat a la mota que fa de Gateway a través del protocol de nivell de transport UDP.

Amb aquesta configuració, el Gateway rebrà cada cert temps el següent missatge:

```
"Data received from aaaa::c30c:0:0:66 on port # from port # with length
#: Temp =  21.4375”
```

A partir del qual es podrà saber de quina mota prové i la seva teva temperatura, tot i que el tractament d'aquestes dades es realitzarà en el mòdul agregador.

4.2 Mòdul agregador/LSG

Tal i com s'ha explicat a l'apartat de disseny, el mòdul agregador/LSG es l'encarregat de recollir les dades de la xarxa de motes z1 a través del Gateway i després assignar un nom ICN, és a dir un ID per a cada una d'elles. A més serà l'encarregat de gestionar les peticions que rep el mòdul servidor de part del consumidors.

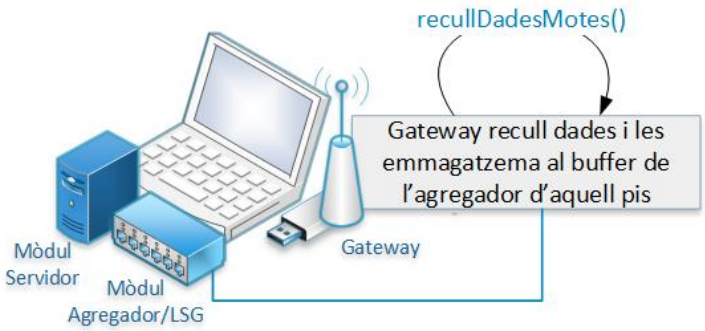

Figura 26. Diagrama de recull de dades

A la Figura 26 es pot veure representat com la mota que fa de Gateway recull les dades de forma periòdica de la resta de motes. A cada mota que envia dades se li assigna un ID en funció del ip origen. Per exemple per a les motes del pis 1 s'assigna el seguen ID, el qual da referencia al #Pis.#Localització, d'aquesta manera es podrà sol·licitar dades només amb el nom ICN i prescindint de la ip:

```
mota1=Mota('aaaa::c30c:0:0:66','1.1','','')
mota2=Mota('aaaa::c30c:0:0:67','1.2','','')
```

Les dades rebudes s'emmagatzemen al buffer del mòdul agregador, que es un fitxer txt, al qual s'afegeix l'ID de la mota, l'hora en que s'ha rebut la dada i la seva temperatura, amb el següent format:

```
ID 1.1 (Sat Apr 28 19:36:54 2018): 11.8125
ID 1.2 (Sat Apr 28 19:36:51 2018): 12.6250
```

Només es guarda una temperatura de cada mota, per lo que quan es rep una dada més recent s'actualitza el fitxer amb la nova dada i la nova hora.

El procediment per a la gestió de les peticions provinents del servidor es pot veure detallat a la Figura 27. Al rebre una petició que li envia el servidor, primer mira si es una dada del seu pis o no, si es del seu pis li envia la resposta. En el cas de no ser una dada del seu pis es posa a buscar la dada sol·licitada entre les que te emmagatzemades, en el cas de tenir-la primer s'assegura de que la dada té menys de 30 minuts i li envia la resposta al servidor. En el cas de que l'agregador no disposi de la dada o bé aquesta fos més antiga de 30 minuts, aquest sol·licitarà la petició a un altre node de la xarxa ICN, concretament al del pis corresponent a la dada sol·licitada, el mòdul agregador del altre node busca la dada i quan la ha trobat se la envia al mòdul agregador que l'ha sol·licitat, aquest agregador l'emmagatzema per si més endavant se la tornen a demanar i li envia la resposta al servidor.

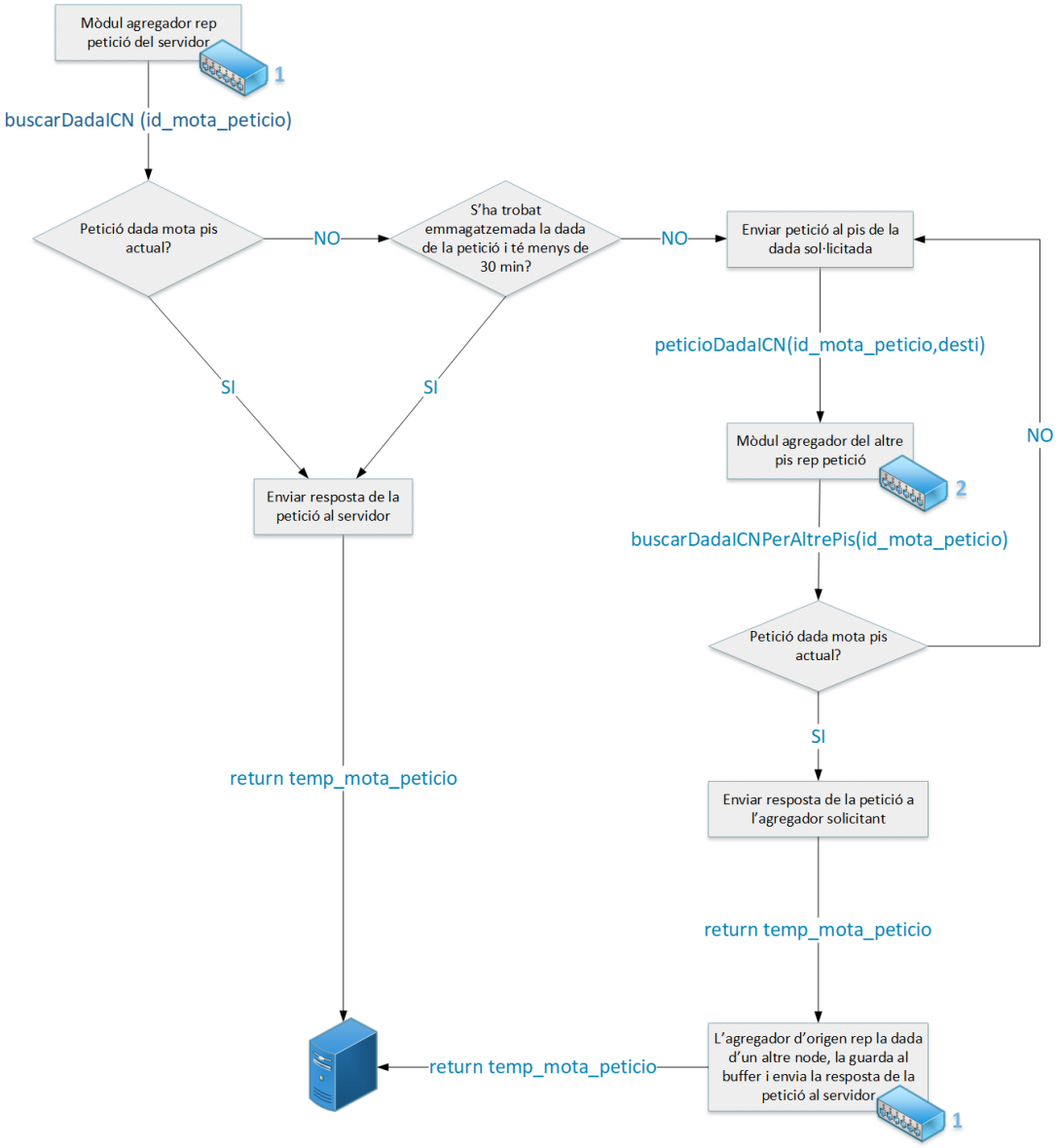

Figura 27. Diagrama d'estats agregador

Les diferents funcions que realitza el mòdul agregador/LSG s'han aconseguit mitjançant la creació de scripts de Python, algun dels quals han d'estar en continua execució, com podria ser la recollida de dades de les motes o el servidor que permet a altres nodes establir comunicació, i d'altres només s'executaran quan hi hagi una petició del servidor. A la Taula 3 es poden veure detallades totes les interfícies de les funcions que s'han creat amb les seves respectives accions. A més cada mòdul agregador tindrà la seva pròpia configuració en funció del pis en el que estigui. El codi complert es pot veure a l'Annex 2.

MÒDUL AGREGADOR/LSG	
Funcions	Explicació
recullDadesMotes ( )	Amb aquesta funció en continua execució, el mòdul agragador pot recollir les dades que rep la mota Gateway de la resta de motes de la xarxa, i les emmagatzema al buffer del seu propi pis, que es un fitxer txt. En funció de la IP de l'origen, se li associa un ID amb el número del pis i número de la seva localització (#.#) per així poder-les identificar.
recollirDades (nom_buffer)	Amb aquest funció el mòdul agragador processa les dades que ha recollit la mota del gateway al buffer. Cada dada tindrà un id de mota on indica el pis i la ubicació, la seva temperatura rebuda i la hora en que s'ha rebut.
buscaDadaICN (id_mota_peticio)	Aquesta és la funció principal del mòdul agregador, busca la dada sol·licitada per la petició del servidor, en cas de no trobar-la o de ser molt antiga, ha de trucar a la funció peticioDadaICN per buscar-la en un altre node. Quan la rep la resposta de l'altre mòdul agregador la guarda, en el cas de tenir una dada antiga aquesta s'actualitza.
peticioDadaICN (id_mota_peticio)	Si el mòdul agregador no ha trobat la dada que buscava, realitza una petició a un altre mòdul agregador per obtenir-la, mitjançant la comunicació per socket client-servidor.
buscarDadaICNPerAltrePis (id_mota_peticio)	El mòdul agregador que ha rebut una sol·licitud d'un altre node busca la dada sol·licitada per enviar-se-la al mòdul agregador que se l'ha sol·licitat.
serverSocket ( )	Amb aquesta funció en constant execució el mòdul agregador manté oberta la connexió del seu servidor per a que qualsevol node pugui sol·licitar alguna dada a traves de la comunicació client-servidor .

Taula 3. Interfícies funcions mòdul agregador

4.3 Mòdul Servidor

El mòdul servidor és el que permet que els consumidors puguin sol·licitar una dada i aquest els hi retorni la resposta del mòdul agregador.

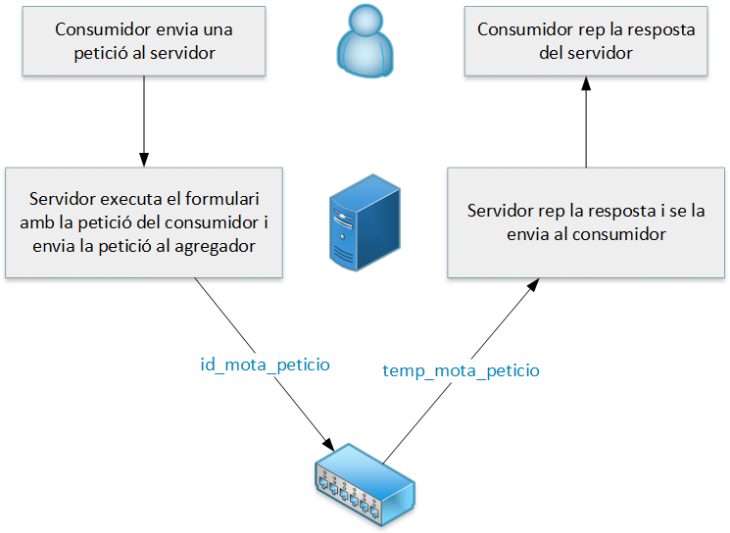

Figura 28. Diagrama d'estats servidor

A la Figura 28 es pot observar quins són els diferents estats per els que passa el mòdul servidor. Quan rep una petició per part d'un consumidor s'executa el formulari i s'envia la petició al mòdul agregador, un cop el mòdul agregador li envia la resposta, el servidor la reenviarà al consumidor com a resposta de la seva petició.

La realització d'aquest mòdul ha sigut mitjançant un servidor web Apache, ja que després d'estudiar a l'estat de l'art les diferents plataformes IoT s'ha considerat que per aquest cas era molt més senzill fer-ho amb un petit servidor.

S'ha realitzat un formulari HTML amb el que el consumidor pot seleccionar la dada que vol i enviar la sol·licitud, tal i com es pot veure a la Figura 29.

Figura 29. Servidor web Apache, formulari HTML

Quan es selecciona la petició i se li dona al botó s'executa un codi php que envia la petició al mòdul agregador i aquest executant la petició al seu sistema li envia la resposta, tal i com es pot veure a la Figura 30.

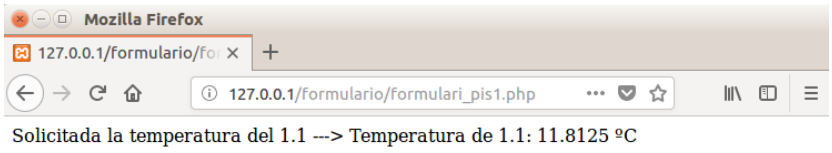

Figura 30. Resposta del servidor a la petició

Cada pis té el seu propi servidor el qual enviarà la petició al seu respectiu mòdul agregador. El codi es pot veure detallat a l'Annex 3.

4.4 Integració mòduls

Una vegada integrat el mòdul agregador i el mòdul servidor, s'ha aconseguit que totes les funcionalitats funcionessin conjuntament.

Per una banda, per recollir les dades que es van reben de la xarxa de sensors s'ha d'executar de manera continua el script de python "script_port_serie_pis#", el qual llegeix els missatges que es reben pel USB on es tingui connectada la mota Gateway, se li assigna el ID de la mota segons la IP d'on provingui i s'emmagatzema la nova dada al buffer del pis que li correspongui. A la Figura 31 es pot veure una captura del procediment d'execució del script i de com s'emmagatzema la dada més nova de cada localització al buffer amb el ID corresponent, l'hora i la temperatura.

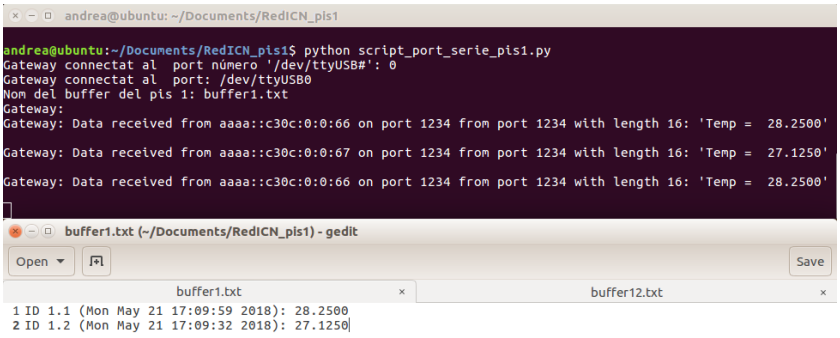

Figura 31. Script per recollir dades Gateway

D'altra banda mitjançant el servidor que hi ha a cada pis, es podrà sol·licitar la temperatura tant de les motes d'aquell mateix pis com d'un altre. A la Figura 32 es pot veure com al sol·licitar la dada del la mota 1.1 des de el servidor del pis 1, aquest retorna la temperatura més actualitzada que tingui emmagatzemada en el buffer en aquell moment.

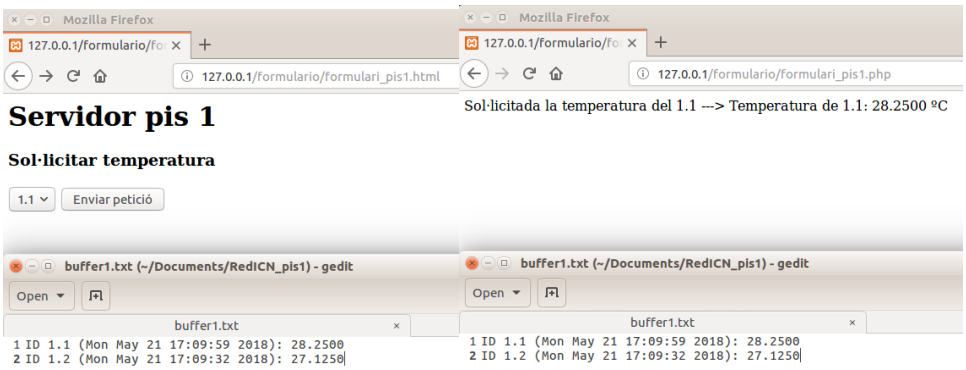

Figura 32. Petició dada mota propi pis

En canvi, si es vol sol·licitar la dada d'un altre pis poden passar dos coses: que la tingui emmagatzemada al buffer amb una antiguitat menor de 30 minuts i el servidor retorni aquella dada, o bé que no estigui o sigui més antiga de 30 minuts i s'hagi de sol·licitar al mòdul agregador d'aquell pis per enviar-la i emmagatzemar-la al buffer per si hi hagués alguna petició posteriorment. A la Figura 33 es pot observar com al sol·licitar la dada de la mota 2.2 des de el servidor del pis 1 s'ha hagut de sol·licitar al pis 2, i amb el que aquets ha retornat s'emmagatzema al buffer i es respon al servidor.

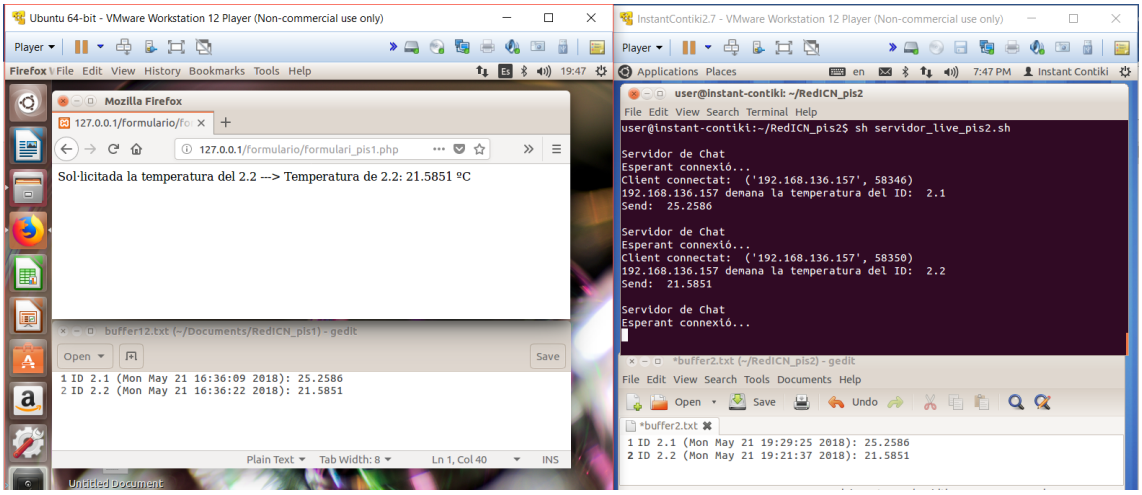

Figura 33. Petició dada d'un altre pis

5 Conclusions

El projecte s'ha centrat un repte tecnològic de recol·lecció d'informació mitjançant un sistema IoT i amb WSN, utilitzant l'aplicació d'arquitectures de xarxa ICN i DTN en aquest entorn IoT, per l'intercanvi d'informació entre els diferents objectes de la xarxa. L'objectiu ha sigut l'estudi del paradigma ICN com a capa d'adreçament, enrutament i servei (com a cerca semàntica d'informació) i DTN com a extensió de la xarxa IoT en el cas de xarxes de curt abast.

Amb l'estudi de les tecnologies ICN i DTN s'ha realitzat el disseny per al sistema de control intel·ligent de la temperatura d'un edifici. Els sistema està format per una xarxa mallada de sensors de temperatura, la qual està interconnectada per poder orquestrar de manera autònoma els diferents punts, sense necessitat d'una gestió centralitzada des d'un centre de control.

S'ha implementat el disseny mitjançant el desenvolupament de diversos mòduls els quals conjuntament permeten que el sistema de control intel·ligent de temperatura sigui possible. Per una banda, s'ha desenvolupat el funcionament de les motes que contenen el sensors de temperatura i s'ha realitzat el sistema que recull aquestes dades. I per altre banda, s'ha realitzat el mòdul servidor que permet sol·licitar la informació de qualsevol punt i el mòdul agregador que és el que permet gestionar la petició i retorna la seva resposta.

Finalment s'han aconseguit resultats satisfactoris amb la implementació del sistema amb la tecnologia ICN, tot i que la part que s'havia dissenyat per d'implantar la tecnologia DTN ha quedat fora de l'abast d'aquest projecte, ja que a mesura que avançava el l'estudi ens vam adonar de l'excessiva complexitat d'afegir els dos temes que es tenien planejats, i es va decidir que el millor era centrar-se en implementar només la part de ICN i deixar com línies futures la implementació de DTN.

Una altre de les propostes per a línies futures seria dissenyar i desplegar un sistema d'emmagatzematge distribuït, de l'estil "Fog/Edge Computing/Storage", que doni suport al desplegament del sistema que s'ha creat.

A nivell personal aquest projecte ha sigut tot un repte, no només per treballar temes d'un àmbit diferent al que havia treballat mai, sinó també perquè aquestes tecnologies encara no estan gaire investigades i per lo tant ha sigut complicat buscar com implementar-ho de manera pràctica i que fos útil. Aquest estudi m'ha aportat una visió sobre temes molt actuals i que poden ser una solució en futur, tot i que considero que encara queda molt per investigar, el que si que crec que pot ser una solució a problemes que tenim a dia d'avui com a que sigui possible la comunicació sense tenir en compte una ubicació especifica o bé que pugui haver-hi connexions inestables i no es perdi informació.

6 Referencies

[1] M. Amadeo, C. Campolo, A. Iera i A. Molinaro, «Information Centric Networking in IoT scenarios: the Case of a Smart Home,» IEEE, 2015.

[2] F. Z. Benhamida, A. Bouabdellah i Y. Challal, «Using Delay Tolerant Network for the Internet of Things: Opportunities and challenges,» IEEE, 2017.

[3] E. Monticelli, M. Arumaithurai i ed al, «Combining Opportunistic and Information Centric Networks in Real World Applications,» 2015.

[4] E. Monticelli, B. M. Schubert i et al., «An Information Centric Approach for Communications in Disaster Situations,» IEEE, 2014.

[5] D. Trossen, A. Sathianseelan i J. Ott, «Towards an Information Centric Network Architecture for Universal Internet Access,» SIGCOMM, 2016.

[6] S. Kaveh, Y. Lin i ed al., «Less Pain, Most of the Gain: Incrementally Deployable ICN,» SIGCOMM, 2013.

[7] B. Wissingh, . C. Wood, A. Afanasyev, L. Zhang, D. Oran i C. Tschudin, «Information-Centric Networking (ICN): Terminology,» Internet-Draft, 2017.

[8] F. Zhang, Y. Zhang i D. Raychaudhuri, «Edge Caching and Nearest Replica Routing in Information-Centric Networking,» IEEE, 2016.

[9] D. Ezquerra, À. Fabregas, M. de Toro i J. Borrell, «Un Enfoque Tolerante a Interrupciones para la Seguridad de la Internet de las Cosas,» RECSI, 2014.

[10] L. Romero Amondaray i H. R. Sánchez Paz, «Las redes tolerantes al retardo (DTN). Una solución a las comunicaciones rurales en Cuba,» TELEMATIQUE, 2010.

[11] Y. Zhang, D. Raychadhuri i et al., «ICN based Arquitecture for IoT,» Internet-Draft, 2017.

[12] «ERP Software blog,» 22 September 2016. [En línia]. Available: http://www.erpsoftwareblog.com/2016/09/hosting-microsoft-dynamics-cloud-evaluating-iaas-paas-saas/. [Últim accés: Diciembre 2017].

[13] V. Ventura, «Polaridad,» Noviembre 2015. [En línia]. Available: https://polaridad.es/almacenar-datos-internet-cosas-iot/#almacenar-datos-internet-cosas-iot.

[14] «Apache Friends,» Decembre 2017. [En línia]. Available: https://www.apachefriends.org/.

[15] E. Rico, «Ermesh,» Abril 2017. [En línia]. Available: http://www.ermesh.com/plataformas-iot-internet-de-las-cosas/.

[16] «Interoute,» [En línia]. Available: https://www.interoute.es/what-iaas.

[17] B. Brandon, «Gartner: Amazon's cloud is 10x bigger than its next 14 competitors, combined,» 21 Maig 2015.

[18] «Docs Amazon Web Services,» [En línia]. Available: https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html. [Últim accés: Decembre 2017].

[19] «Amazon Web Services,» [En línia]. Available: https://aws.amazon.com/es/iot/?ft=n. [Últim accés: Diciembre 2017].

[20] J. Fernandez, «¿Qué es AWS Greengrass?,» *Un poco de Java y +,* 28 Juny 2017.

[21] «Microsoft Azure,» [En línia]. Available: https://azure.microsoft.com/es-es/. [Últim accés: Decembre 2017].

[22] «Docs Microsoft,» [En línia]. Available: https://docs.microsoft.com/es-es/azure/#pivot=products&panel=iot. [Últim accés: Diciembre 2017].

[23] P. Patierno, «An IoT Platforms Match: Microsoft Azure IoT vs Amazon AWS IoT,» *Paolo Patierno's Blog,* 13 Octubre 2015.

[24] K. Weins, «Cloud Computing Trends: 2018 State of the Cloud Survey,» *RIGHT SCALE: CLOUD MANAGEMENT BLOG,* 13 Febrer 2018.

[25] «Z1 Datasheet,» Zolertia.

[26] «Zolertia,» [En línia]. Available: https://zolertia.io/. [Últim accés: Decembre 2017].

[27] «Zolertia Source Forge,» Zolertia, [En línia]. Available: http://zolertia.sourceforge.net/wiki/index.php/Main_Page. [Últim accés: Decembre 2017].

[28] O. Vadillo Gutiérrez, «Provisión de servicios de la Internet de las Cosas sobre redes de sensores basadas en 6LoWPAN,» 2014.

[29] R. Romero Jotel, «Demostrador d'un sistema de calefacció de la llar basat en OpenMote, OpenWSN (IEEE 802.15.4e) i thethings.iO,» UOC-URL, 2015.

[30] W. Thomas, «Open WSN,» 9 Novembre 2012. [En línia]. Available: https://openwsn.atlassian.net/wiki/spaces/OW/pages/688151/RPL.

[31] «Slideshare,» Maig 2014. [En línia]. Available: https://www.slideshare.net/asobimat/rpl-dodag. [Últim accés: Decembre 2017].

[32] «GitHub,» Contiki Unicast-Receiver, [En línia]. Available: https://github.com/contiki-os/contiki/blob/master/examples/ipv6/simple-udp-rpl/unicast-receiver.c. [Últim accés: Març 2018].

[33] «GitHub,» Zolertia, [En línia]. Available: https://github.com/Zolertia/Resources/wiki/The-Z1-mote. [Últim accés: Decembre 2017].

[34] «Z1 Brochure,» Zolertia.

[35] T. Winter, P. Thubert i A. Brandt, «RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks,» IETF, 2012.

[36] «Contiki,» [En línia]. Available: http://www.contiki-os.org/index.html. [Últim accés: Decembre 2017].

[37] «GitHub,» Contiki Unicast Sender, [En línia]. Available: https://github.com/contiki-os/contiki/blob/master/examples/ipv6/simple-udp-rpl/unicast-sender.c. [Últim accés: Març 2018].

Annex 1 Codi motes z1

Codi per a les motes Gateway:

```
/*
 * Copyright (c) 2011, Swedish Institute of Computer Science.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above
copyright
 *    notice, this list of conditions and the following disclaimer in
the
 *    documentation and/or other materials provided with the
distribution.
 * 3. Neither the name of the Institute nor the names of its
contributors
 *    may be used to endorse or promote products derived from this
software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE INSTITUTE AND CONTRIBUTORS ``AS
IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE INSTITUTE OR CONTRIBUTORS BE
LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
OF
 * SUCH DAMAGE.
 *
 * This file is part of the Contiki operating system.
 *
 */

#include "contiki.h"
#include "lib/random.h"
#include "sys/ctimer.h"
#include "sys/etimer.h"
#include "net/uip.h"
#include "net/uip-ds6.h"
#include "net/uip-debug.h"

#include "simple-udp.h"
```

```
#include "servreg-hack.h"

#include "net/rpl/rpl.h"

#include <stdio.h>
#include <string.h>

#define UDP_PORT 1234
#define SERVICE_ID 190

#define SEND_INTERVAL         (10 * CLOCK_SECOND)
#define SEND_TIME       (random_rand() % (SEND_INTERVAL))

static struct simple_udp_connection unicast_connection;

/*---------------------------------------------------------------------
-------*/
PROCESS(unicast_receiver_process, "Unicast receiver example process");
AUTOSTART_PROCESSES(&unicast_receiver_process);
/*---------------------------------------------------------------------
-------*/
static void
receiver(struct simple_udp_connection *c,
         const uip_ipaddr_t *sender_addr,
         uint16_t sender_port,
         const uip_ipaddr_t *receiver_addr,
         uint16_t receiver_port,
         const uint8_t *data,
         uint16_t datalen)
{
  printf("Data received from ");
  uip_debug_ipaddr_print(sender_addr);
  printf(" on port %d from port %d with length %d: '%s'\n",
         receiver_port, sender_port, datalen, data);
}
/*---------------------------------------------------------------------
-------*/
static uip_ipaddr_t *
set_global_address(void)
{
  static uip_ipaddr_t ipaddr;
  int i;
  uint8_t state;

  uip_ip6addr(&ipaddr, 0xaaaa, 0, 0, 0, 0, 0, 0, 0);
  uip_ds6_set_addr_iid(&ipaddr, &uip_lladdr);
  uip_ds6_addr_add(&ipaddr, 0, ADDR_AUTOCONF);

  printf("IPv6 addresses: \n");
  for(i = 0; i < UIP_DS6_ADDR_NB; i++) {
    state = uip_ds6_if.addr_list[i].state;
    if(uip_ds6_if.addr_list[i].isused &&
       (state == ADDR_TENTATIVE || state == ADDR_PREFERRED)) {
      uip_debug_ipaddr_print(&uip_ds6_if.addr_list[i].ipaddr);
      printf("\n");
    }
  }

  return &ipaddr;
}
```

```
/*---------------------------------------------------------------------
-------*/
static void
create_rpl_dag(uip_ipaddr_t *ipaddr)
{
  struct uip_ds6_addr *root_if;

  root_if = uip_ds6_addr_lookup(ipaddr);
  if(root_if != NULL) {
    rpl_dag_t *dag;
    uip_ipaddr_t prefix;

    rpl_set_root(RPL_DEFAULT_INSTANCE, ipaddr);
    dag = rpl_get_any_dag();
    uip_ip6addr(&prefix, 0xaaaa, 0, 0, 0, 0, 0, 0, 0);
    rpl_set_prefix(dag, &prefix, 64);
    PRINTF("created a new RPL dag\n");
  } else {
    PRINTF("failed to create a new RPL DAG\n");
  }
}
/*---------------------------------------------------------------------
-------*/
PROCESS_THREAD(unicast_receiver_process, ev, data)
{
  uip_ipaddr_t *ipaddr;

  PROCESS_BEGIN();

  servreg_hack_init();

  ipaddr = set_global_address();

  create_rpl_dag(ipaddr);

  servreg_hack_register(SERVICE_ID, ipaddr);

  simple_udp_register(&unicast_connection, UDP_PORT,
                      NULL, UDP_PORT, receiver);

  while(1) {
    PROCESS_WAIT_EVENT();
  }
  PROCESS_END();
}
/*---------------------------------------------------------------------
-------*/
```

Codi per a la resta de motes de la xarxa, les quals envien les dades al Gateway:

```
/*
 * Copyright (c) 2011, Swedish Institute of Computer Science.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above
copyright
 *    notice, this list of conditions and the following disclaimer in
the
 *    documentation and/or other materials provided with the
distribution.
 * 3. Neither the name of the Institute nor the names of its
contributors
 *    may be used to endorse or promote products derived from this
software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE INSTITUTE AND CONTRIBUTORS ``AS
IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE INSTITUTE OR CONTRIBUTORS BE
LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
OF
 * SUCH DAMAGE.
 *
 * This file is part of the Contiki operating system.
 *
 */

#include "contiki.h"
#include "lib/random.h"
#include "sys/ctimer.h"
#include "sys/etimer.h"
#include "net/uip.h"
#include "net/uip-ds6.h"
#include "net/uip-debug.h"

#include "dev/i2cmaster.h"
#include "dev/tmp102.h"

#include "sys/node-id.h"
```

```
#include "simple-udp.h"
#include "servreg-hack.h"

#include <stdio.h>
#include <string.h>

#define UDP_PORT 1234
#define SERVICE_ID 190

#define SEND_INTERVAL         (60 * CLOCK_SECOND)
#define SEND_TIME       (random_rand() % (SEND_INTERVAL))

static struct simple_udp_connection unicast_connection;

/*---------------------------------------------------------------------
-------*/
PROCESS(unicast_sender_process, "Unicast sender example process");
AUTOSTART_PROCESSES(&unicast_sender_process);
/*---------------------------------------------------------------------
-------*/
static void
receiver(struct simple_udp_connection *c,
         const uip_ipaddr_t *sender_addr,
         uint16_t sender_port,
         const uip_ipaddr_t *receiver_addr,
         uint16_t receiver_port,
         const uint8_t *data,
         uint16_t datalen)
{
  printf("Data received on port %d from port %d with length %d\n",
         receiver_port, sender_port, datalen);
}
/*---------------------------------------------------------------------
-------*/
static void
set_global_address(void)
{
  uip_ipaddr_t ipaddr;
  int i;
  uint8_t state;

  uip_ip6addr(&ipaddr, 0xaaaa, 0, 0, 0, 0, 0, 0, 0);
  uip_ds6_set_addr_iid(&ipaddr, &uip_lladdr);
  uip_ds6_addr_add(&ipaddr, 0, ADDR_AUTOCONF);

  printf("IPv6 addresses: \n");
  for(i = 0; i < UIP_DS6_ADDR_NB; i++) {
    state = uip_ds6_if.addr_list[i].state;
    if(uip_ds6_if.addr_list[i].isused &&
       (state == ADDR_TENTATIVE || state == ADDR_PREFERRED)) {
      uip_debug_ipaddr_print(&uip_ds6_if.addr_list[i].ipaddr);
      printf("\n");
    }
  }
}
/*---------------------------------------------------------------------
-------*/
static const char * getTemperature() {
 int16_t tempint;
 uint16_t tempfrac;
```

```
 int16_t raw;
 uint16_t absraw;
 int16_t sign;
 char minus = ' ';
 char cadena[32]="";

 sign = 1;
 raw = tmp102_read_temp_raw();
 absraw = raw;
 if(raw < 0) {
 absraw = (raw ^ 0xFFFF) + 1;
 sign = -1;
 }
 tempint = (absraw >> 8) * sign;
 tempfrac = ((absraw >> 4) % 16) * 625;
 minus = ((tempint == 0) & (sign == -1))? '-' : ' ';
 sprintf(cadena,"Temp = %c%d.%04d", minus, tempint, tempfrac);
 return &cadena;
}
/*---------------------------------------------------------------------*/
PROCESS_THREAD(unicast_sender_process, ev, data)
{
  static struct etimer periodic_timer;
  static struct etimer send_timer;
  uip_ipaddr_t *addr;

  PROCESS_BEGIN();

  servreg_hack_init();

  set_global_address();

  simple_udp_register(&unicast_connection, UDP_PORT,
                      NULL, UDP_PORT, receiver);

  etimer_set(&periodic_timer, SEND_INTERVAL);

  tmp102_init();


  while(1) {

    PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&periodic_timer));
    etimer_reset(&periodic_timer);
    etimer_set(&send_timer, SEND_TIME);

    PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&send_timer));
    addr = servreg_hack_lookup(SERVICE_ID);
    if(addr != NULL) {
      static unsigned int message_number;
      char buf[20];

      printf("Sending unicast to ");
      uip_debug_ipaddr_print(addr);
      printf("\n");
      char result[128] ="";
      strcat(result,getTemperature ());
      printf(result);
      printf("\n");
      simple_udp_sendto(&unicast_connection, result, strlen(result) +
1, addr);
```

```
    } else {
      printf("Service %d not found\n", SERVICE_ID);
    }
  }

  PROCESS_END();
}
/*---------------------------------------------------------------------------
--------*/
```

Annex 2 Codi Mòdul Agregrador/LSG

Tot el codi següent és el que li correspon al pis 1, els altres pisos tenen el mateix codi només canviat algun detalls d'ubicació del fitxers, noms del buffers, configuració sockets, etc.

Codi del script que s'executa per emmagatzemar les dades rebudes a través del Gateway, el qual ha d'estar en constantment funcionant per poder rebre les dades al moment:

```
#!/usr/bin/env python
# -*- coding: 850 -*-

#""" --------------------SCRIP_PORT_SERIE_PIS1-------------------- """

import serial
import time
import datetime

class Mota:
      def __init__(self, ip_mota, id_mota, temp, hora):
            self.ip_mota = ip_mota
            self.id_mota = id_mota
            self.temp = temp
            self.hora = hora

""" -------------recullDadesMotes és la funció encarregada
d'emmagatzemar les dades rebudes per el gateway------------ """

def recullDadesMotes():
      nport = '/dev/ttyUSB'
      port_name = nport + raw_input("Gateway connectat al  port número
'/dev/ttyUSB#': ")
      ser = serial.Serial(
                        port = port_name,\
                        baudrate = 115200,\
                        parity = serial.PARITY_NONE,\
                        stopbits = serial.STOPBITS_ONE,\
                        bytesize = serial.EIGHTBITS,\
                        timeout = 30)
      print("Gateway connectat al  port: " + ser.portstr)

      data_file = "buffer1.txt"
      print("Nom del buffer del pis 1: "+data_file)

      while True:
            #Llegim totes les línies que entren per el gateway
            line = str(ser.readline())
            print("Gateway: "+line)
            if len(line) >= 1:
                  #De la línea rebuda es treu el ID i la temperatura
                  ip=line[19:36]
                  temp=line[90:97]
                  #Es guarda la temperatura a la variable de la classe
mota
                  if ip == mota1.ip_mota:
                        mota1.temp=temp
                        mota1.hora=time.strftime("%c")
                  elif ip == mota2.ip_mota:
```

```
                        mota2.temp=temp
                        mota2.hora=time.strftime("%c")
                  #S'actualitza el txt del buffer amb la dada rebuda
                  if mota1.temp=='' and mota2.temp!='':
                        reescriuredada(mota2.id_mota, mota2.temp,
mota2.hora, data_file)

                  elif mota2.temp=='' and mota1.temp!='':
                        reescriuredada(mota1.id_mota, mota1.temp,
mota1.hora, data_file)

                  elif mota1.temp!='' and mota2.temp!='':
                        file = open(data_file, "w+")
                        file.write("ID "+mota1.id_mota+'
('+mota1.hora+'): '+mota1.temp+'\n'+"ID "+mota2.id_mota+'
('+mota2.hora+'): '+mota2.temp+'\n')
                        file.close()

            time.sleep(1)
      ser.close()

""" ------------reescriuredada és la funció que actualitza la dada
que li han enviat d'un altre pis ----------- """
def reescriuredada(id_mota, temp, hora, buffer_file):
      nova_linea="ID "+id_mota+" ("+hora+"): "+temp+"\n"
      with open(buffer_file,'r') as fh:
            #Guardem les dades que hi han al fitxer
            lines = fh.readlines()
      with open(buffer_file,'w+') as fh:
            for line in lines:
                  id_mota_txt=line[3:6]
                  if id_mota == id_mota_txt:
                        #Reescrivim la dada a actualitzar
                        fh.write(nova_linea)
                  else:
                        #Escrivim les dades que no s'han d'actualitzar
                        fh.write(line)


"""--------------------EXECUCIÓ LA FUNCIÓ PRINCIPAL----------------"""
#Creació motes pis 1
mota1=Mota('aaaa::c30c:0:0:66','1.1','','')
mota2=Mota('aaaa::c30c:0:0:67','1.2','','')
recullDadesMotes()
```

Codi que llegeix el fitxer buffer que realitza l'script anterior i emmagatzema les dades rebudes de la xarxa de motes z1.

```
#!/usr/bin/env python
# -*- coding: 850 -*-

""" ------------------RECOLLIR_DADES_PIS1-------------------- """


#Direcció dels buffers amb les dades del pis 1
buffer1="/home/andrea/Documents/RedICN_pis1/buffer1.txt"


class Mota:
```

```
      def __init__(self, id_mota, temp, hora):
            self.id_mota = id_mota
            self.temp = temp
            self.hora = hora

""" -------------RecollirDades és la funció encarregada d'emmagatzemar
les dades al buffer del propi pis------------ """

def RecollirDades(file_name):
      with open(file_name,"r") as fh:
            line = fh.readline()

            while line:
                  #Llegir linees buffer pis 1
                  id_mota=line[3:6]
                  hora=line[19:27]
                  temp=line[35:42]
                  #Es guarda la temperatura i la hora a les variables
de la classe mota
                  if id_mota == mota1.id_mota:
                        mota1.temp=temp
                        mota1.hora=hora
                  elif id_mota == mota2.id_mota:
                        mota2.temp=temp
                        mota2.hora=hora

                  line = fh.readline()

"""--------------------EXECUCIÓ LA FUNCIÓ PRINCIPAL-------------------
"""
#Motes registrades al pis 1
mota1=Mota('1.1','','')
mota2=Mota('1.2','','')
RecollirDades(buffer1)
```

Codi de la funció principal, que es l'encarregada de buscar la dada de la petició que li ha enviat el servidor.

```
#!/usr/bin/env python
# -*- coding: 850 -*-

""" --------------------BUSCAR_DADA_ICN_PIS1--------------------- """

#Importem el mòdul que recull les dades del pis1
import recollirdades_pis1 as dades
import time
import datetime
import sys
import os

#Ubicació buffers de les dades d'altres pisos
buffer12="/home/andrea/Documents/RedICN_pis1/buffer12.txt"
buffer13="/home/andrea/Documents/RedICN_pis1/buffer13.txt"


""" -------------buscarDadaICN és la funció principal que retorna la
dada sol·licitada------------ """
```

```
def buscarDadaICN(id_mota_peticio):
      #Importem el mòdul que sol·licita dades a altres pisos
      import peticiodadaicn_pis1

      if id_mota_peticio == dades.mota1.id_mota:
            print(dades.mota1.temp)
            return dades.mota1.temp

      elif id_mota_peticio == dades.mota2.id_mota:
            print(dades.mota2.temp)
            return dades.mota2.temp
      else:
            hora=time.strftime("%c")

      return_comprovar_dada=comprovarDadesICNAltrePis(id_mota_peticio)
            if return_comprovar_dada==0:
                  #La dada de la mota no està emmagatzemada, s'ha de
sol·licitar
                  desti=id_mota_peticio[0]
                  result=getattr(peticiodadaicn_pis1,'peticioDadaICN')
                  temp_mota=result(id_mota_peticio,desti)
                  print(temp_mota)
                  guardardadesaltrepis(id_mota_peticio, temp_mota,
hora)
                  return temp_mota

            elif return_comprovar_dada==1:
                  #La dada de la mota que hi ha emmagatzemada té més de
30 minuts, s'ha de sol·licitar
                  desti=id_mota_peticio[0]
                  result=getattr(peticiodadaicn_pis1,'peticioDadaICN')
                  temp_mota=result(id_mota_peticio,desti)
                  print(temp_mota)
                  reescriuredadesaltrepis(id_mota_peticio, temp_mota,
hora)
                  return temp_mota

            else:
                  #La dada si que està emmagatzemada i té menys de 30
minuts
                  print(return_comprovar_dada)
                  return return_comprovar_dada


"""-------------------------FUNCIONS AUXILIARS---------------------"""

""" ------------comprovarDadesICNAltrePis és la funció que comprova
si té emmagatzemada la dada sol·licitada d'un altre pis------------
"""

def comprovarDadesICNAltrePis(id_mota_peticio):

      if id_mota_peticio[0] == '2':
            #Si al pis 1 li demanen una dada del pis 2
            with open(buffer12,"r") as fh:
                  line = fh.readline()
                  while line:
                        id_mota=line[3:6]
                        if id_mota == id_mota_peticio:
```

```
                                    #Llegir buffer del pis 1 amb dades del
pis 2
                                    datatime_str=line[8:32]
                                    datatime_new_str=time.strftime("%c")
                                    datatime=time.strptime(datatime_str,"%c")

      datatime_new=time.strptime(datatime_new_str,"%c")
                                    #Comprovar que la dada té menys de 30
minuts
                                    tdelta=
datetime.datetime.fromtimestamp(time.mktime(datatime_new)) -
datetime.datetime.fromtimestamp(time.mktime(datatime))
                                    if tdelta.total_seconds()>1800.0:
                                          #La dada té més de 30 minuts, s'ha
de sol·licitar dada nova
                                          return 1
                                    else:
                                          #La dada que hi ha al buffer es
actual
                                          return line[35:42]
                              line = fh.readline()
                        #No esta emmagatzemada aquesta dada del pis 2
                        return 0


      elif id_mota_peticio[0] == '3':
            #Si al pis 2 li demanen una dada del pis 3
            with open(buffer13,"r") as fh:
                  line = fh.readline()
                  while line:
                        id_mota=line[3:6]
                        if id_mota == id_mota_peticio:
                              #Llegir buffer del pis 1 amb dades del
pis 3
                              datatime_str=line[8:32]
                              datatime_new_str=time.strftime("%c")
                              datatime=time.strptime(datatime_str,"%c")

      datatime_new=time.strptime(datatime_new_str,"%c")
                              #Comprovar que la dada té menys de 30
minuts
                              tdelta=
datetime.datetime.fromtimestamp(time.mktime(datatime_new)) -
datetime.datetime.fromtimestamp(time.mktime(datatime))
                              if tdelta.total_seconds()>1800.0:
                                    #La dada té més de 30 minuts, s'ha
de sol·licitar dada nova
                                    return 1
                              else:
                                    #La dada que hi ha al buffer es
actual
                                    return line[35:42]
                              line = fh.readline()
                        #No esta emmagatzemada aquesta dada del pis 3
                        return 0


""" -------------guardardadesaltrepis és la funció que guarda la dada
que li han enviat d'un altre pis si no estava------------ """

def guardardadesaltrepis(id_mota, temp, hora):
```

```
      nova_linea="ID "+id_mota+" ("+hora+"): "+temp+"\n"
      if id_mota[0] == '2':
            with open(buffer12, 'a+') as fh:
                  #Escribim la nova dada
                  fh.write(nova_linea)

      elif id_mota[0] =='3':
            with open(buffer13, 'a+') as fh:
                  #Escribim la nova dada
                  fh.write(nova_linea)


""" ------------reescriuredadesaltrepis és la funció que actualitza
la dada que li han enviat d'un altre pis ------------ """

def reescriuredadesaltrepis(id_mota, temp, hora):
      nova_linea="ID "+id_mota+" ("+hora+"): "+temp+"\n"
      if id_mota[0] == '2':
            with open(buffer12,'r') as fh:
                  #Guardem les dades que hi han al fitxer
                  lines = fh.readlines()
            with open(buffer12,'w+') as fh:
                  for line in lines:
                        id_mota_txt=line[3:6]
                        if id_mota == id_mota_txt:
                              #Reescribim la dada a actualitzar
                              fh.write(nova_linea)
                        else:
                              #Escribim les dades que no s'han
d'actualitzar
                              fh.write(line)

      elif id_mota[0] =='3':
            with open(buffer13, 'r') as fh:
                  #Guardem les dades que hi han al fitxer
                  lines= fh.readlines()
            with open(buffer13,'w+') as fh:
                  for line in lines:
                        id_mota_txt=line[3:6]
                        if id_mota == id_mota_txt:
                              #Reescribim la dada a actualitzar
                              fh.write(nova_linea)
                        else:
                              #Escribim les dades que no s'han
d'actualitzar
                              fh.write(line)


"""--------------------EXECUCIÓ LA FUNCIÓ PRINCIPAL----------------"""
buscarDadaICN(sys.argv[1])
```

Codi de la funció que sol·licita la dada a un altre pis:

```
#!/usr/bin/env python
# -*- coding: 850 -*-

""" --------------------PETICIO_DADA_ICN_PIS1--------------------- """


import socket
```

```
#Direcció pis 2
IP_2="192.168.136.159"
PORT_2=8000

#Direcció pis 3
IP_3="192.168.136.159"
PORT_3=9000
""" -------------peticioDadaICN és la funció encarregada de
sol·licitar dades a altres pisos------------ """

def peticioDadaICN(id_mota_peticio,desti):

      if desti == "2":
            #Sol·licitem dada al pis 2
            temp=clientSocket(IP_2,PORT_2, id_mota_peticio)
            return temp

      elif desti == "3":
            #Sol·licitem dada al pis 3
            temp=clientSocket(IP_3,PORT_3, id_mota_peticio)
            return temp



""" -------------clientSocket és la funció encarregada de crear
comunicació amb el servidor de l'altre pis------------ """

def clientSocket(ip, port, id_mota_peticio):
      socket_cliente = socket.socket(socket.AF_INET,
socket.SOCK_STREAM,0)
      #Es connecta al servidor i envia petició
      socket_cliente.connect((ip,port))
      socket_cliente.send(id_mota_peticio)
      #Reb la respota del servidor i tanca la connexió
      temp=socket_cliente.recv(1024)
      socket_cliente.close()
      return temp
```

Codi de la funció que retorna una dada que ha sol·licitat un altre pis:

```
#!/usr/bin/env python
# -*- coding: 850 -*-

""" --------------BUSCAR_DADA_ICN_PER_ALTRE_PIS_PIS1-------------- """


#Importem el mòdul que recull les dades del pis 1
import recollirdades_pis1 as dades


""" -------------buscarDadaICNPerAltrePis és la funció que busca la
dada que li sol·licita un altre pis i se la envia------------ """

def buscarDadaICNPerAltrePis(id_mota_peticio):
      if id_mota_peticio == dades.mota1.id_mota:
            return dades.mota1.temp

      elif id_mota_peticio == dades.mota2.id_mota:
```

```
            return dades.mota2.temp
```

Codi del servidor del socket per poder establir connexió amb altres pisos:

```
#!/usr/bin/env python
# -*- coding: 850 -*-

""" -------------------------SERVIDOR_PIS1---------------------- """



import socket
import time

#Direcció pis 1
IP=""
PORT=7000

""" -------------serverSocket és la funció encarregada de rebre les
peticions del altres pisos------------ """

def serverSocket():

      s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
      s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
      s.bind((IP,PORT))
      s.listen(1)

      print ("\nServidor de Chat")

      print ("Esperant connexió...")
      sc, addr = s.accept()
      print "Client connectat: ", addr
      while True:

            #Es rep el missatge, amb el mètode recv rebem les dades
            id_mota_peticio = sc.recv(1024)
            import buscardadaicnperaltrepis_pis1

      result=getattr(buscardadaicnperaltrepis_pis1,'buscarDadaICNPerAl
trePis')
            temp=result(id_mota_peticio)

            #Si es reben dades es mostra la IP i la sol·licitud rebuda
            print str(addr[0]) + " demana la temperatura del ID: ",
id_mota_peticio

            #S'envia la resposta al client
            sc.send(temp)
            print "Send: ", temp
            sc.shutdown(0)
            sc.close()
            s.shutdown(0)
            s.close()
            break


serverSocket()
```

Script que executa de manera constant el codi del servidor del socket per a que la connexió amb altres pisos sigui possible en qualsevol moment.

```
#! /bin/bash

#""" ---------------------SERVIDOR_LIVE_PIS1---------------------- """


while true
do
      python servidor_pis1.py
done
```

Annex 3 Codi Mòdul Servidor

Codi del formulari HTML del servidor web Apache:

```
<html>
<meta charset="utf-8" />
    <body>
    <h1>Servidor pis 1</h1>
    <h3>Sol·licitar temperatura</h3>
    <form action="formulari_pis1.php" method="post">
    <select name="place">
          <option value="1.1"> 1.1</option>
          <option value="1.2"> 1.2</option>
          <option value="2.1"> 2.1</option>
          <option value="2.2"> 2.2</option>
          <option value="3.1"> 3.1</option>
          <option value="3.2"> 3.2</option>
        </select>
    <input type="submit" value="Enviar petició" />
    </form>
    </body>
    </html>
```

Codi php que s'executa al donar-li al botó d'enviar petició, el qual mana a la funció de buscardadicn ( ) del mòdul agregador.

```
<?php
/* Muestra los datos enviados a través de un formulario HTML
a través del estado actual de PHP */
echo "Sol·licitada la temperatura del ";

while(list($nombre,$id)=each($_POST)){
      echo "$id ---> ";
      $salida = exec("python /home/andrea/Documents/RedICN_
_pis1/buscardadaicn_pis1.py $id");
      echo "Temperatura de $id: $salida °C"."</br>";
}
?>
```