Estudio de la viabilidad de RPL y Contiki para un entorno de sensorización en ganadería

Orlando Olaya Trinidad

MÁSTER EN INTERNET DE LAS COSAS

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin Máster en Internet de las Cosas

Curso 2019/2020

Convocatoria: Julio de 2020
Calificación: 8.5 (Notable)

Directores:

Sandra Catalán Pallarés
Francisco Igual Peña

Autorización de difusión

ORLANDO OLAYA TRINIDAD

15 de Julio de 2020

El/la abajo firmante, matriculado/a en el Máster en Internet de las Cosas de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: *"Estudio de la viabilidad de RPL y Contiki para un entorno de sensorización en ganadería"*, realizado durante el curso académico 2019-2020 bajo la dirección de Sandra Catalán Pallarés y Francisco Igual Peña en el Departamento de Arquitectura de Computadores y Automática, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Resumen

El presente Trabajo de Fin de Máster aborda el estudio de la viabilidad en el uso del protocolo para redes con pérdidas RPL en entornos con sensores móviles en el ámbito de la ganadería inteligente o *smart farming*. Su aplicación permitirá la monitorización de parámetros biométricos o ambientales en entornos ganaderos, específicamente en un grupo familiar de vicuñas. Mediante la integración de dicho protocolo con el protocolo 6LowPAN, y el uso del sistema operativo Contiki, se evalúa una estrategia en la cual cada individuo monitoriza, y a la vez transmite mensajes a modo de enrutador, reduciendo el coste y aumentando la fiabilidad de los despliegues.

El desarrollo propuesto se basa en simulaciones intensivas y automatizadas a través del simulador *COOJA* equipado con el plugin *Mobility*. Además, el trabajo incluye un estudio detallado de la dinámica de movimiento de un grupo familiar de vicuñas, así como la generación pseudo-aleatoria de patrones de movimiento que permitan una simulación fiel a la realidad. Los resultados experimentales obtenidos corroboran el uso 6LowPAN, y su protocolo de enrutamiento asociado RPL, aprovechando la reducida distancia individual entre individuos, para permitir encaminar datos inter-individuo y alcanzar un punto común de conexión con una red externa. Desde el punto de vista experimental, se han determinado distintas topologías de red y distribución de nodos sensores y enrutadores de borde, así como el impacto que surge sobre el consumo energético en distintos escenarios reales.

Palabras clave

6LoWPAN, Contiki, ContikiMAC, COOJA, Mobility, RPL, Topologías MESH, *Smart Farming*, Vicuñas.

Abstract

The present Master's work deals with the study of the viability of using the protocol for networks with RPL losses in mobile sensor environments in the field of intelligent livestock or *smart farming*. Its application will allow the monitoring of biometric or environmental parameters in farming environments, specifically in a family group of vicuñas. By integrating this protocol with the 6LowPAN protocol, and the use of the Contiki operating system, a strategy is evaluated in which each individual monitors, and at the same time transmits messages as a router, reducing the cost and increasing the reliability of deployments.

The proposed development is based on intensive and automated simulations through the simulator *COOJA* equipped with the plugin *Mobility*. In addition, the work includes a detailed study of the movement dynamics of a family group of vicuñas, as well as the semialeatorial generation of movement patterns that allow a simulation faithful to reality. The experimental results obtained corroborate the use of 6LowPAN, and its associated RPL routing protocol, taking advantage of the reduced individual distance between individuals, to allow inter-individual data routing and reaching a common point of connection with an external network. From the experimental point of view, different network topologies and distribution of sensor nodes and edge routers have been determined, as well as the impact that arises on energy consumption in different real scenarios.

Keywords

6LoWPAN, Contiki, ContikiMAC, COOJA, MESH Topologies, Mobility, RPL, *Smart Farming*, Vicuñas.

Índice general

Agradecimientos

Quiero agradecer a mis padres y hermano por su apoyo incondicional.

También extender un agradecimiento a mis tutores, los doctores Sandra Catalán Pallarés y Francisco Igual Peña, por su guía invaluable en el desarrollo del presente trabajo.

Dedicatoria

A mi padre por ser mi inspiración, a mi madre por siempre animarme a ser mejor y a mi hermano por ser un amigo y guía.

Capítulo 1

Introducción

1.1. Smart Farming e IoT.

En los últimos tiempos, la elevada precisión que conceden los sensores integrados que miden distintos factores ambientales, ha permitido el desarrollo de una agricultura de precisión, que conlleva una mejora significativa de la productividad, el rendimiento, la rentabilidad [12] y la reducción de la huella ambiental, con técnicas de riego eficiente dirigido y el uso óptimo de fertilizantes y pesticidas en el cultivo [1], o un uso adecuado de alimentos y antibióticos en animales [10, 12].

La agricultura de precisión es una idea tan antigua como la agricultura misma [1] y precede al concepto de agricultura inteligente, que consiste en la recopilación, procesamiento, análisis de datos y automatización de los procesos de cultivo que conlleva a una mejora en la operación y gestión agrícola. También permite que los granjeros tomen decisiones mas informadas, esto debido al impacto que, condiciones ambientales (lluvia, temperatura, humedad, granizo, etc) y eventos impredecibles (enfermedades de animales o plagas) tienen en la agricultura [12].

En 1999 el término Internet de las cosas o IoT es introducido por Kevin Ashton para referirse a la recopilación de información del ambiente e interacción con el mundo físico, sin intervención humana [26]. IoT y su naturaleza interoperable, escalable, generalizada y abierta es perfecta para la agricultura y la ganadería inteligente. De hecho, con el pasar del

tiempo, IoT ha ganado un impulso en el sector agrícola; un ejemplo lo tenemos en Australia, donde los ganaderos están obligados a que cada cabeza de ganado tenga implantada en su oreja un dispositivo RFID pasivo con el fin de informar los movimientos entre granjas conformando así una base de datos nacional en línea [12].

IoT se ha aplicado con éxito en ciudades inteligentes, atención médica y hogares inteligentes, brindando conectividad perfecta por medio de diversos estándares abiertos (6LoWPAN, ZigBee, CoAP, REST, MQTT) y semánticas (RDF, OWL, SWE, etc) entre diversos actores, componentes, dispositivos, procesos y plataformas [12]. Estos brindan la oportunidad de revolucionar las técnicas agrícolas tradicionales.

El uso de IoT para la agricultura y ganadería inteligente conlleva varios beneficios, incluidos la reducción del riesgo de bloqueo de proveedores, la adopción de maquinaria y sistemas de detección/automatización de diferentes compañías, ya que estos podrían volverse fácilmente interoperables en el sistema inteligente de la granja en general. El intercambio de datos es ahora más sencillo entre diferentes componentes heterogéneos (a nivel *hardware* y *software*), permitiendo mayor automatización y despliegues más complejos con menos esfuerzo mediante el uso de estándares de Internet [12].

1.2. Motivación y planteamiento del problema

En Perú, la mayor parte de los trabajos de investigación relacionados con la producción de lana de vicuña tienen un enfoque biológico, es decir, consideran la alimentación y técnicas de reproducción del animal.

Actualmente, la gran mayoría de producción de lana de vicuña es realizada empleando técnicas de crianza ancestrales, con poco o ningún control tecnológico. Los animales realizan sus actividades en un *pseudo cautiverio*, desplazándose en áreas muy extensas y con patrones difícilmente parametrizables, lo que dificulta la recogida de datos a menos que cada animal sea autosuficiente para enviarlos de forma autónoma (por ejemplo, con una conexión satelital propia, o bien con tecnologías LPWAN (*Low-Power, Wide Area Networks*), con el

consiguiente coste energético y de adquisición y despliegue.

El uso de técnicas de ganadería inteligente se plantea con un potencial impacto muy positivo, ya que permitiría la monitorización del grupo familiar de vicuñas en tiempo real. Se podría disponer, así, de manera proactiva de información como tipo de terreno, condiciones climáticas o presencia de depredadores, entre otros.

De entre las tecnologías de transmisión asociadas a IoT, 6LowPAN, y su protocolo de enrutamiento asociado RPL, se plantean como dos alternativas que potencialmente pueden adaptarse correctamente a las características de la aplicación seleccionada (monitorización de parámetros biométricos o ambientales en grupos móviles de vicuñas). Específicamente, estos protocolos:

- Son abiertos y están ampliamente documentados y estandarizados vía RFCs [32, 33].
- Su diseño está fuertemente enfocado a redes con pérdidas y con elevadas restricciones de consumo energético.
- Forman parte integral de diversos sistemas operativos enfocados principalmente a IoT (por ejemplo, Contiki).
- En un entorno geográficamente extenso, permiten el despliegue de una red *mesh*, con la ventaja de que los elementos intermedios pueden actuar a la vez como fuentes de información y como enrutadores, haciendo llegar los paquetes enviados a su destino final.

En entornos con elementos móviles, como es nuestro caso, existen ciertos retos adicionales que se tratarán en el presente trabajo. Estos retos se pueden resumir en:

- Patrones de movimiento en los nodos. Normalmente, las redes RPL se tratan desde un enfoque totalmente estático. Es por tanto conveniente realizar un estudio sobre el comportamiento de este tipo de algoritmos de enrutamiento en entornos móviles. Este

estudio debe incluir parámetros como pérdida de paquetes, velocidad de convergencia, impacto de la velocidad y patrón de movimiento en la calidad de la recepción, etc.

- Impacto en el consumo energético del proceso de enrutamiento con nodos móviles. Diferenciación de consumo en nodos aislados o en contacto con otros nodos vecinos.

- Distribución óptima de los nodos (y routers de borde que dan acceso a redes externas) para minimizar el coste de adquisición y mantenimiento.

1.3. Objetivos

El objetivo general de este trabajo es desarrollar un estudio sobre la viabilidad del protocolo de enrutamiento para redes con pérdidas RPL en entornos con nodos sensores móviles, con aplicación directa en la sensorización de diversos parámetros en entornos ganaderos, específicamente aplicado en grupos familiares de vicuñas.

Del objetivo general se derivan otros objetivos de carácter específico:

- Caracterizar el movimiento de un grupo familiar de vicuñas para su uso en el simulador COOJA con el plugin Mobility.

- Determinar el impacto de las distintas topologías de red (distribución de nodos móviles y routers de borde) en la calidad del despliegue.

- Reducir el impacto sobre la energía consumida por un nodo móvil para su óptimo uso en RPL.

- Diseñar y automatizar escenarios de simulación fieles a escenarios reales.

El estudio realizado en el trabajo se basa intensivamente en simulaciones a través de un entorno de simulación (COOJA). La razón para tal enfoque se justifica principalmente por la dificultad, a día de hoy, de realizar pruebas de campo reales en amplios entornos geográficos. Las conclusiones obtenidas, sin embargo, son directamente aplicables a un entorno real. Sin

embargo, este enfoque de implementación se plantea como trabajo futuro, y no se ha tratado en este TFM.

1.4. Estructura de la memoria

La organización del presente trabajo se basa en 5 capítulos descritos a continuación: El Capítulo 2 contiene un estudio detallado de la dinámica de movimiento de vicuñas, que ha permitido desarrollar una metodología y código para para generar movimientos pseudo aleatorios con una estructura específica para su uso en el simulador COOJA equipado con el plugin *Mobility*. El Capítulo 3 es un estudio teórico de los conceptos necesarios para desplegar los entornos experimentales, planteados en el marco de un entorno de simulación. El Capítulo 4 expone los resultados obtenidos en diferentes escenarios experimentales realistas, que modelan, de forma aproximada, los entornos que podrían encontrarse en un despliegue real. El Capítulo 5 lista las principales conclusiones obtenidas y plantea, a su vez, una serie de trabajos de extensión en forma de trabajo futuro.

Atendiendo a la normativa, los Capítulos 6 y 7 corresponden a la traducción al idioma inglés de los capítulos de introducción y conclusiones.

Capítulo 2

Patrones de movimiento en colonias de animales

La Vicuña (*Vicugna vicugna*) en un animal que pertenece a la familia de los camélidos que derivan de especies prehistóricas originarias de Norteamérica, desaparecidas hace más de 11 millones de años; se trata de una especie silvestre al igual que el guanaco (*Lama guanicoe*) [28]. Existen también algunas especies domésticas como la llama (*Lama glama*) y alpaca (*Vicugna pacos*). En muchas regiones de algunos países sudamericanos como Ecuador, Perú, Bolivia, Argentina o Chile, las fibras son valiosas y constituyen el principal sustento económico de algunos de sus pobladores. Existen dos tipos de fibra: finas y cortas (*down*) y las gruesas y largas (cerda) [28].

El hábitat de estos camélidos sudamericanos se encuentra a una altitud por encima de los 3000 metros sobre el nivel del mar; los ambientes son en su mayoría mesetas (planicies) y laderas cordilleranas con una alta tasa de heladas y disponibilidad de fuentes de agua muy limitada [28].

El Cuadro 2.1 resume la información sobre población y distribución de los camélidos sudamericanos. Perú es el país con el mayor número de estos camélidos, aproximadamente 5 millones de animales, además de ser el país que más alpacas y vicuñas alberga. Bolivia tiene la mayoría de las llamas y Argentina la mayoría de los guanacos [28]. Nótese que, pese a centrarse en las vicuñas como actores principales del presente trabajo, éste es directamen-

Camélido	Perú	Bolivia	Argentina	Chile
Alpaca	3.041.598	269.285	pocas	28.551
Llama	1.462.730	2.237.170	161.402	50.132
Vicuña	147.000	12.047	131.220	27.921
Guanaco	pocos	pocos	636.477	27.150

Cuadro 2.1: *Población de camélidos en Sudamérica [28].*

te aplicable a cualquier animal criado en rebaño y régimen de semilibertad, simplemente adaptando los patrones de movimiento planteados.

La vicuña es una de las especies de camélidos más delgada y pequeña y alcanza un peso que varía entre los 35 y 50 Kg. Debido a que su hábitat es un ambiente frío, su fibra es muy tupida y fina, con una gran retención de temperatura. Esta fibra debe ser obtenida de animales vivos por medio de la esquila. Existen tres formas de crianza [28]:

- Crianza en cautiverio, implementado en Argentina bajo las normas propuestas por el Instituto Nacional de Tecnología Agropecuaria (INTA) de Argentina.
- Aprovechamiento en silvestría, implementada en Perú, Bolivia y Argentina.
- Crianza en semi-cautiverio, o sistemas de cercos cuyos principales promotores son Perú y Chile.

En el presente trabajo, nos centraremos en escenarios basados en las dos últimas modalidades de crianza, por suponer retos mayores desde el punto de vista de las comunicaciones en un despliegue de sensorización o *smart farming*.

Debido a que las fibras presentan un crecimiento lento, su obtención debe ser óptima. Antiguamente, en la época incaica, los *chakus* (captura de vicuñas) se realizaban en intervalos trianuales. Actualmente, tanto los gobiernos de Perú como de Chile implementan una estrategia para que las comunidades campesinas obtengan la fibra de forma óptima; por ejemplo, en Perú se ha implementado la modalidad de "de uso". Posteriormente los derechos cambiaron a propiedad, por lo que es obligación de las comunidades su cuidado y

conservación [28].

A día de hoy, la captura y esquila se realizan una vez por año, con lo cual se logra una mejor vigilancia y solo en animales que tengan fibras de un largo superior a 2 cm [28].

2.1. Dinámica de movimiento de vicuñas

Un grupo familiar es una unidad que se determina dentro de una población de vicuñas. Está constituida por un macho adulto (conocido en Bolivia como *janacho*), de 5 a 6 hembras adultas y sus crías menores de un año (conocidas como *teques*) [3].

Los animales que forman este grupo familiar toman dos tipos de distancias entre ellos [31]:

- *Distancia individual*: Espacio mínimo entre ellos sin hostilidad.
- *Distancia social*: En la cual se pierde cohesión con el grupo familiar.

Las vicuñas son animales territoriales; dentro de la población existen algunas que son sedentarias y otras más móviles [31].

Las hembras están a una distancia media de 2,6 metros; esta distancia no se ve modificada en función de las actividades que realicen, pero en cambio los machos tienen una distancia inter-individual mayor en comparación al de las hembras, siendo la distancia media aproximadamente de 8 metros. Es importante remarcar que esta distancia media puede variar según el tipo de actividad que estén realizando como descansar o alimentarse [31].

Las áreas de pastoreo en época seca abarcan entre 4,2 y 6,2 hectáreas, mientras que durante una estación lluviosa los territorios pueden reducirse entre 3,5 y 5,2 hectáreas. Se observa que estos grupos familiares tienen un territorio de pastoreo en las zonas más bajas y de descanso nocturno en colinas que las protegen del viento [3].

En cuanto a hábitats, se identifican principalmente dos [3]:

- *Pastizal y/o pradera seca*: vegetación muy baja, espesor de entre 1 y 15 cm, predominando gramíneas, hierbas en roseta y rastreras que forman cojines [3].

Promedio de Área de uso y Distancia recorrida durante un semestre.							
Detalle	Dic.	Ene.	Feb.	Mar.	Abr.	May.	Promedio
Distancia (km./día)	5,20	4,25	6,95	7,75	4,85	5,30	5,72
Área de uso (ha/día)	47,50	27,15	35,35	69,26	39,40	63,50	47,03

Cuadro 2.2: *Promedio de área de uso vs. distancia recorrida por un grupo familiar de vicuñas [3].*

- *Bofedal*: vegetación con forma de cojín de gran espesor, con un elevado contenido de materia orgánica y nitrógeno [24].

Estos dos macro-hábitats se encuentran en las áreas planas de la región de Ulla Ulla, descrita en los sectores denominados Cultivo Pampa (zona I) [3] y Pista Pampa (zona II) [3] con pobre vegetación con respecto a la anterior [3].

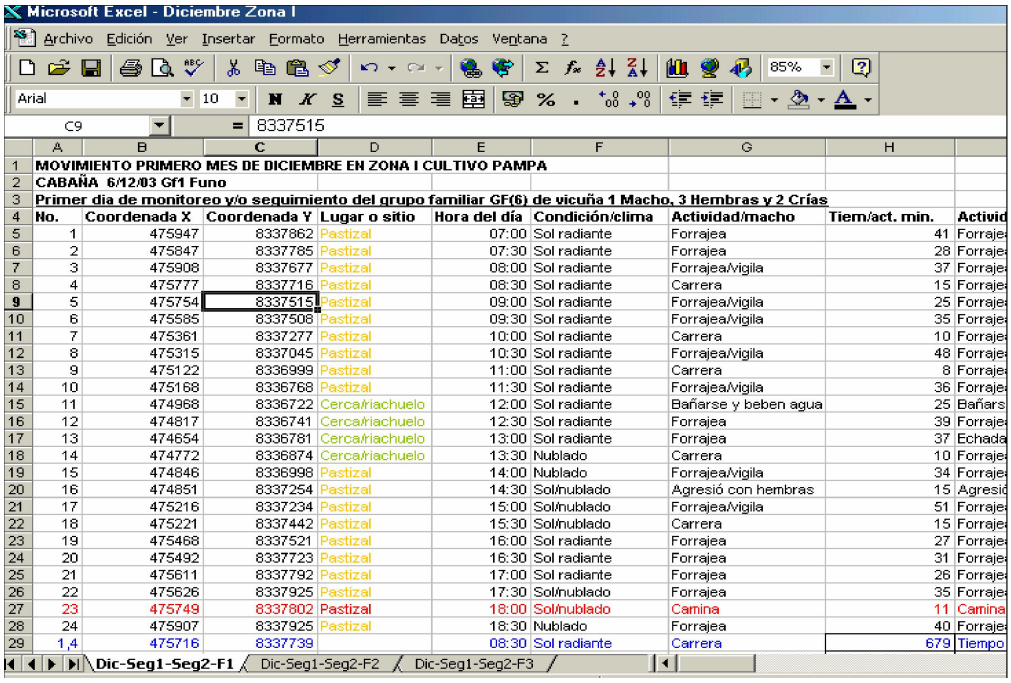
Las vicuñas beben agua todos los días y siempre se encuentran a una distancia máxima de dos kilómetros de un cuerpo de agua. Éste puede ser una laguna, un arroyo, o un manantial. En cuanto a su alimentación, se produce mayormente en los bofedales, ya que estos poseen un forraje verde [3].

En un seguimiento de un grupo familiar por un aproximado de 8 a 10 horas por día, desarrollado por Bernabé Alex Mamani Choque [3], se grabaron sus posiciones con instrumentos GPS y se usó el software Excel y ArcView GIS 3.2 para su análisis.

Las Figura 2.1 y 2.2 [3] corresponden a un archivo Excel, donde se ha etiquetado cada una de las coordenadas obtenidas por GPS guiándose de imágenes satelitales; las etiquetas corresponden a: tipo de lugar, condición climática, actividad realizada por cada miembro del grupo familiar, duración de la actividad, miembro responsable del desplazamiento y tipo de vegetación.

Este grupo familiar estuvo compuesto por unas 20 vicuñas aproximadamente. En el Cuadro 2.2 se muestra un promedio de Área de uso vs. Distancia recorrida [3].

El desplazamiento de este grupo familiar se puede expresar en los siguientes porcentajes debido a: instintos, guiado por macho y guiado por hembras, tal como se muestra en



Microsoft Excel - Diciembre Zona I

Archivo Edición Ver Insertar Formato Herramientas Datos Ventana ?

Arial 10 — C9 = 8337515

	A	B	C	D	E	F	G	H	
1	MOVIMIENTO PRIMERO MES DE DICIEMBRE EN ZONA I CULTIVO PAMPA								
2	CABAÑA 6/12/03 Gf1 Funo								
3	Primer dia de monitoreo y/o seguimiento del grupo familiar GF(6) de vicuña 1 Macho, 3 Hembras y 2 Crías								
4	No.	Coordenada X	Coordenada Y	Lugar o sitio	Hora del día	Condición/clima	Actividad/macho	Tiem/act. min.	Activid
5	1	475947	8337862	Pastizal	07:00	Sol radiante	Forrajea	41	Forraje
6	2	475847	8337785	Pastizal	07:30	Sol radiante	Forrajea	28	Forraje
7	3	475908	8337677	Pastizal	08:00	Sol radiante	Forrajea/vigila	37	Forraje
8	4	475777	8337716	Pastizal	08:30	Sol radiante	Carrera	15	Forraje
9	5	475754	8337515	Pastizal	09:00	Sol radiante	Forrajea/vigila	25	Forraje
10	6	475585	8337508	Pastizal	09:30	Sol radiante	Forrajea/vigila	35	Forraje
11	7	475361	8337277	Pastizal	10:00	Sol radiante	Carrera	10	Forraje
12	8	475315	8337045	Pastizal	10:30	Sol radiante	Forrajea/vigila	48	Forraje
13	9	475122	8336999	Pastizal	11:00	Sol radiante	Carrera	8	Forraje
14	10	475168	8336768	Pastizal	11:30	Sol radiante	Forrajea/vigila	36	Forraje
15	11	474968	8336722	Cerca/riachuelo	12:00	Sol radiante	Bañarse y beben agua	25	Bañars
16	12	474817	8336741	Cerca/riachuelo	12:30	Sol radiante	Forrajea	39	Forraje
17	13	474654	8336781	Cerca/riachuelo	13:00	Sol radiante	Forrajea	37	Echada
18	14	474772	8336874	Cerca/riachuelo	13:30	Nublado	Carrera	10	Forraje
19	15	474846	8336998	Pastizal	14:00	Nublado	Forrajea/vigila	34	Forraje
20	16	474851	8337254	Pastizal	14:30	Sol/nublado	Agresió con hembras	15	Agresió
21	17	475216	8337234	Pastizal	15:00	Sol/nublado	Forrajea/vigila	51	Forraje
22	18	475221	8337442	Pastizal	15:30	Sol/nublado	Carrera	15	Forraje
23	19	475468	8337521	Pastizal	16:00	Sol radiante	Forrajea	27	Forraje
24	20	475492	8337723	Pastizal	16:30	Sol radiante	Forrajea	31	Forraje
25	21	475611	8337792	Pastizal	17:00	Sol radiante	Forrajea	26	Forraje
26	22	475626	8337925	Pastizal	17:30	Sol/nublado	Forrajea	35	Forraje
27	23	475749	8337802	Pastizal	18:00	Sol/nublado	Camina	11	Camina
28	24	475907	8337925	Pastizal	18:30	Nublado	Forrajea	40	Forraje
29	1,4	475716	8337739		08:30	Sol radiante	Carrera	679	Tiempo

Dic-Seg1-Seg2-F1 / Dic-Seg1-Seg2-F2 / Dic-Seg1-Seg2-F3

Figura 2.1: *Etiquetado de coordenadas obtenidas por GPS [3].*

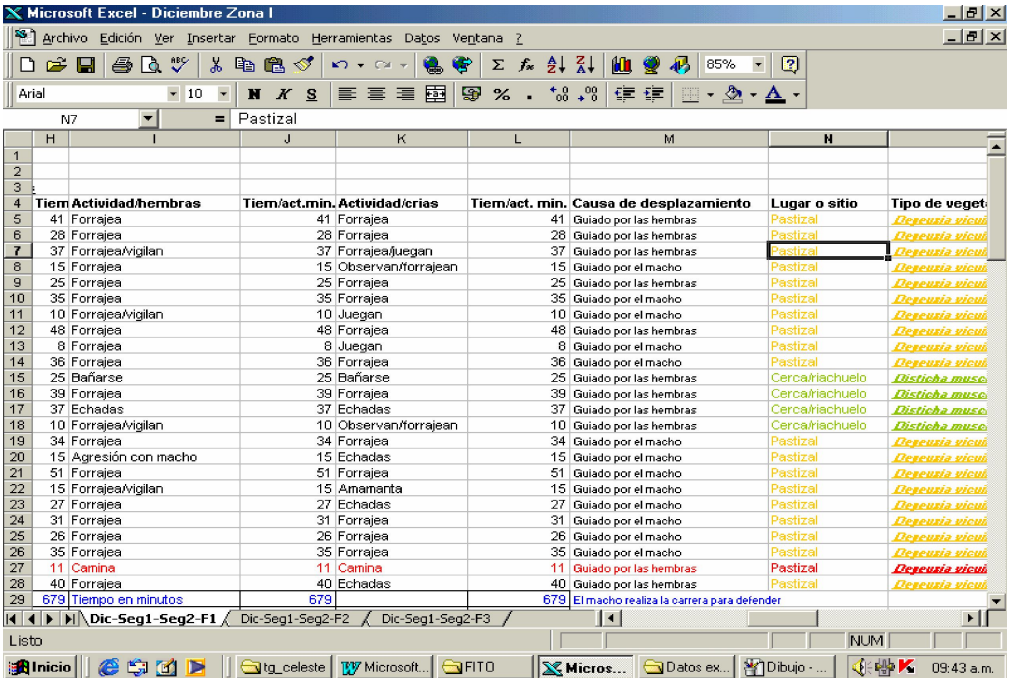
Zonas de estudio	**Zona I Cultivo Pampa**		**Zona I Pista Pampa**	
Actividades	Frecuencia	Porcentaje(%)	Frecuencia	Porcentaje(%)
Avanzan por instinto Guiado	2	8	1	4,16
Guiado por el macho	13	52	12	50
Guiado por las hembras	10	40	11	45,83
Total	25	100	24	100

Cuadro 2.3: *Traslado de los grupo familiares de vicuñas entre localizaciones [3].*

Cuadro 2.3 [3].

En último lugar, el promedio de tiempo de permanencia en diferentes hábitats, como: pastizal, pastizal de transición, cerca y/o en el río y bofedal, se muestra en el Cuadro 2.4.

En general, un estudio detallado de los anteriores patrones de movimiento, sugiere que el uso de tecnologías *mesh* con radio de corto/medio alcance, podría resultar interesante, aprovechando la reducida distancia individual entre individuos, que ayudaría al enrutamiento de datos inter-individuo, encaminado a alcanzar un punto común de conexión con una red externa (en adelante, router de borde).



H	I	J	K	L	M	N	
1							
2							
3							
4	Tiem	Actividad/hembras	Tiem/act.min.	Actividad/crias	Tiem/act. min.	Causa de desplazamiento	Lugar o sitio	Tipo de veget
5	41	Forrajea	41	Forrajea	41	Guiado por las hembras	Pastizal	Depresia vicui
6	28	Forrajea	28	Forrajea	28	Guiado por las hembras	Pastizal	Depresia vicui
7	37	Forrajea/vigilan	37	Forrajea/juegan	37	Guiado por las hembras	Pastizal	Depresia vicui
8	15	Forrajea	15	Observan/forrajean	15	Guiado por el macho	Pastizal	Depresia vicui
9	25	Forrajea	25	Forrajea	25	Guiado por las hembras	Pastizal	Depresia vicui
10	35	Forrajea	35	Forrajea	35	Guiado por el macho	Pastizal	Depresia vicui
11	10	Forrajea/vigilan	10	Juegan	10	Guiado por el macho	Pastizal	Depresia vicui
12	48	Forrajea	48	Forrajea	48	Guiado por las hembras	Pastizal	Depresia vicui
13	8	Forrajea	8	Juegan	8	Guiado por el macho	Pastizal	Depresia vicui
14	36	Forrajea	36	Forrajea	36	Guiado por el macho	Pastizal	Depresia vicui
15	25	Bañarse	25	Bañarse	25	Guiado por las hembras	Cerca/riachuelo	Distichia musc
16	39	Forrajea	39	Forrajea	39	Guiado por las hembras	Cerca/riachuelo	Distichia musc
17	37	Echadas	37	Echadas	37	Guiado por las hembras	Cerca/riachuelo	Distichia musc
18	10	Forrajea/vigilan	10	Observan/forrajean	10	Guiado por las hembras	Cerca/riachuelo	Distichia musc
19	34	Forrajea	34	Forrajea	34	Guiado por el macho	Pastizal	Depresia vicui
20	15	Agresión con macho	15	Echadas	15	Guiado por el macho	Pastizal	Depresia vicui
21	51	Forrajea	51	Forrajea	51	Guiado por el macho	Pastizal	Depresia vicui
22	15	Forrajea/vigilan	15	Amamanta	15	Guiado por el macho	Pastizal	Depresia vicui
23	27	Forrajea	27	Echadas	27	Guiado por el macho	Pastizal	Depresia vicui
24	31	Forrajea	31	Forrajea	31	Guiado por el macho	Pastizal	Depresia vicui
25	26	Forrajea	26	Forrajea	26	Guiado por el macho	Pastizal	Depresia vicui
26	35	Forrajea	35	Forrajea	35	Guiado por el macho	Pastizal	Depresia vicui
27	11	Camina	11	Camina	11	Guiado por las hembras	Pastizal	Depresia vicui
28	40	Forrajea	40	Echadas	40	Guiado por las hembras	Pastizal	Depresia vicui
29	679	Tiempo en minutos	679		679	El macho realiza la carrera para defender		

Figura 2.2: *Etiquetado de coordenadas obtenidas por GPS (continuación) [3]*

Promedio de tiempo (horas/día) de permanencia en diferentes hábitats			
Pastizal	**Cerca y/o en el río**	**Pastizal en transición**	**Bofedal**
7,79	0,87	0,51	0,84

Cuadro 2.4: *Permanencia de un grupo familiar de vicuñas en diferentes hábitats [3].*

2.2. Script para generación de movimiento

Con la dinámica de movimiento de vicuñas descrita en la Sección 2.1, se dan a continuación las directrices para la construcción de un *script* que automatice la emulación del movimiento de un grupo familiar.

Este *script*, programado en lenguaje Python, generará un archivo con extensión `.dat`, que listará las posiciones de cada miembro del grupo familiar junto a una marca de tiempo asociada, en un formato adecuado para su uso con el plugin Mobility, cuyo funcionamiento se detalla en la Sección 3.2.2. Se toma como base el código escrito por Marcus Lunden, que se encuentra a disposición de la comunidad en Github [19].

Los componentes son los siguientes:

`WSNnodelist.py`

Encargado de generar los atributos de cada nodo.

`genConfFile.py`

Genera el archivo de configuración con el nombre `generate-mobility.txt` en el directorio de trabajo, los parámetros que se pueden configurar son los siguientes (a partir de ahora, se utilizará indistintamente el término mota o nodo para referirse a cada uno de los elementos sensorizados en la simulación):

- *Number of nodes*: Número de nodos en movimiento en la simulación.
- *Time for simulation*: Tiempo de simulación. Cuando se llegue al final de esta marca de tiempo, el plugin Mobility volverá a situar las motas en la posición con marca de tiempo cero. Este tiempo no debe entenderse como el tiempo que dura la simulación; se trata del tiempo que toman todas las marcas de posición.
- *Time resolution*: Tiempo que existe entre cada marca de posición.
- *Minimum pause time*: Tiempo mínimo que una mota puede permanecer sin realizar ningún cambio de posición.
- *Maximum pause time*: Tiempo máximo que una mota puede permanecer sin realizar ningún cambio de posición.
- *Maximum x-coordinate*: Máxima posición en el eje X en la que se puede situar una mota en el archivo `.dat`.
- *Maximum y-coordinate*: Máxima posición en el eje Y en la que se puede situar una mota en el archivo `.dat`.
- *Minimum speed*: Velocidad mínima que puede tener una mota.
- *Maximum speed*: Velocidad máxima que puede tener una mota.

- *How long time to disregard from the start of the simulation*: Cuando empieza una simulación no es obligatorio que las motas cambien de posición, este valor permite indicar a partir de qué instante de tiempo (en segundos) se empezarán a mover las motas.

- *Radio Interference range*: Distancia de interferencia.

- *Radio reception range*: Distancia de recepción de la radio.

`generate-mobility.py`

Encargado de generar la posición de cada mota, con los parámetros que contenga el archivo `generate-mobility.txt`. Las funciones más importantes, aplicadas en orden, son la siguientes:

Paso 1. Calcular la nueva posición de cada mota:

$$\begin{aligned} x_{nueva} &= \begin{cases} x_{actual} + paso_{seg} * xdot_{vector\_velocidad} & ; \text{si } x_{actual} < x_{nueva} \\ x_{actual} - paso_{seg} * xdot_{vector\_velocidad} & ; \text{si } x_{actual} > x_{nueva} \end{cases} \\ y_{nueva} &= \begin{cases} y_{actual} + paso_{seg} * ydot_{vector\_velocidad} & ; \text{si } y_{actual} < y_{nueva} \\ y_{actual} - paso_{seg} * ydot_{vector\_velocidad} & ; \text{si } y_{actual} > y_{nueva} \end{cases} \end{aligned} \tag{2.1}$$

Paso 2. Calcular la distancia entre el nodo y la posición que desea alcanzar:

$$\begin{aligned} dx &= x_{actual} - x_{alcanzar} \\ dy &= y_{actual} - y_{alcanzar} \\ d_{nodo} &= \sqrt{dx^2 + dy^2} \end{aligned} \tag{2.2}$$

Paso 3. Calcular una velocidad de forma aleatoria entre los límites máximo y mínimo de una mota:

$$s = random(V_{min}, V_{max}) \tag{2.3}$$

Paso 4. Calcular el vector de velocidad para alcanzar la posición deseada:

$$\begin{aligned} dx &= |x_{actual} - x_{alcanzar}| \\ dy &= |y_{actual} - y_{alcanzar}| \\ xdot &= s * \sqrt{\frac{dx^2 * dy^2}{dy^4 + dx^2 * dy^2}} \\ ydot &= s * \sqrt{1 - \frac{dx^2 * dy^2}{dy^4 + dx^2 * dy^2}} \end{aligned} \tag{2.4}$$

Paso 5. Proceder a actualizar la posición de cada mota en cada marca de tiempo

2.2.1. Ejemplo de generación de archivo de posiciones

Al ejecutar el *script* de generación de movimientos con los siguientes parámetros:

- Number of nodes: 17
- Time for simulation: 28800
- Time resolution: 6
- Minimum pause time: 60
- Maximum pause time: 120
- Maximum x-coordinate: 4000
- Maximum y-coordinate: 2000
- Minimum speed: 1
- Maximum speed: 2
- How long time to disregard from the start of the simulation: 0
- Radio Interference range: 200

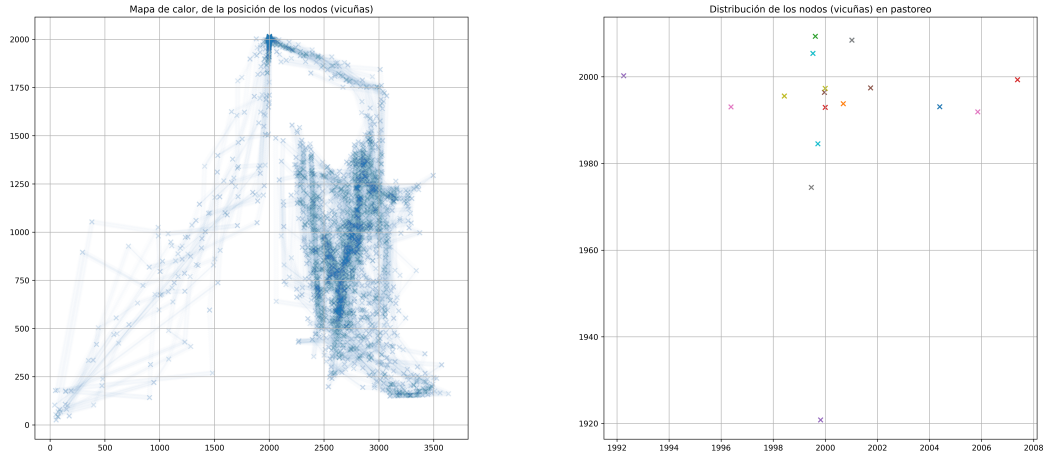


Figura 2.3: *Distribución de movimiento (izquierda) y en zona de pastoreo (derecha).*

- Radio reception range: 150

Se aprecia una distribución en su movimiento como el que se muestra en la Figura 2.3 (izquierda). Las zonas oscuras corresponden a los lugares donde las vicuñas pasan más tiempo, en contraparte a las zonas claras. En zonas de pastoreo, los nodos presentan una distribución como se muestra en la Figura 2.3 (derecha).

Los resultados obtenidos se aproximan a la dinámica de movimiento de vicuñas descritos en la Sección 2.1, lo que corrobora la validez del enfoque propuesto en el diseño e implementación del patrón de movimientos.

Capítulo 3

Entorno experimental

3.1. Contiki

3.1.1. Descripción General

Contiki es un sistema operativo de código abierto diseñado para su ejecución en nodos sensores. Fue desarrollado en el Instituto Sueco de Ciencias de la Computación y presentado en Dunkels et al. [8]. Sus características principales son la carga y descarga dinámica de código en tiempo de ejecución y la posibilidad de ejecutar múltiples subprocesos sobre un núcleo común usando un sistema controlado por eventos [29, 8].

Contiki está implementado en lenguaje C y soporta distintas arquitecturas de microcontroladores, como Texas Instruments MSP430 o Atmel AVR; para dar soporte a IPv6, Contiki cuenta con una pila de red conocida como μIP. Contiki presenta la siguiente estructura de directorios, directamente relacionada con la funcionalidad que proporciona [4]:

- `apps`: Aplicaciones independientes, de la arquitectura del sistema.
- `core`: Código fuente de ContikiOS.
- `cpu`: Código fuente específico para distintas arquitecturas.
- `doc`: Documentación.
- `examples`: Ejemplos de aplicaciones para distintas plataformas.

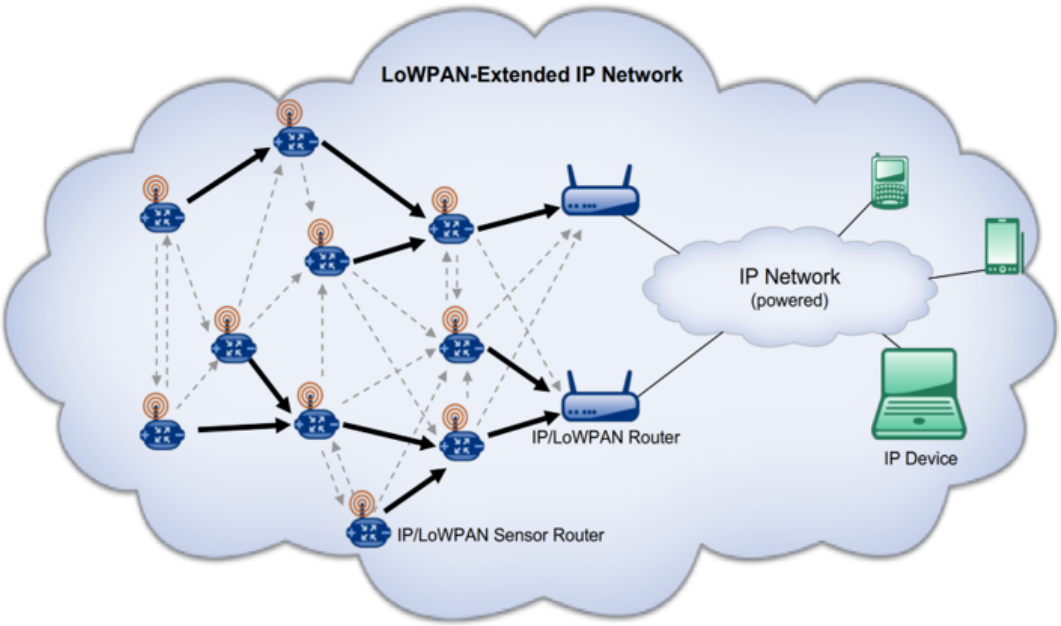


Figura 3.1: *Red de nodos basados en 6LoWPAN [5].*

- `platforms`: Código específico para distintas plataformas.
- `tools`: Aplicaciones que extienden la funcionalidad de ContikiOS.

3.1.2. RPL

El estándar 6LoWPAN [32], permite que se transmitan paquetes IP (en su versión 6) en redes basadas en el estándar de capa de enlace IEEE 802.15.4 [20]. Esto se consigue realizando una adaptación entre la capa de enlace (*link-layer*) y la capa de red (*network-layer*) y reestructurando las cabeceras IPv6. Algunas de sus características son: bajo ancho de banda requerido, tamaño de paquete pequeño, topología tipo *malla* o *estrella* y capacidad de readaptación debido a que los nodos pueden entrar en reposo por largo tiempo. En la Figura 3.1 se aprecia una red de nodos basados en el protocolo 6LowPAN [11, 25].

Los estándares mencionados anteriormente no definen un protocolo de enrutamiento; sin embargo, en cuanto a protocolos de enrutamiento, el *Routing Protocol for Low power and Lossy Networks* (RPL) proporciona caminos de enrutamiento eficientes para los tres tipos

de tráfico de una red de sensores en 6LowPAN: multi punto a punto (MP2P), punto a punto (P2P) y punto a multipunto (P2MP) [18].

La topología formada por RPL es similar a un árbol (DAG o grafo acíclico dirigido), donde cada nodo dentro del árbol posee un rango asignado (*rank*), de tal manera que los nodos seleccionan su ruta con aquellos rangos más bajos [9].

Un DAG enraizado en un solo destino, es decir, en una sola raíz DAG, se le conoce como DODAG [33] (*Destination-Oriented Directed Acyclic Graph*).

Para mantener e identificar una topología, RPL usa cuatro valores [18]:

- *RPLInstanceID*: Responsable de identificar uno o más DODAG. Si hay varios *RPLInstanceID* en la misma red, define un conjunto de DODAG optimizados independientemente para diferentes funciones objetivo (OF). La OF define cómo los nodos seleccionan y optimizan rutas dentro de una instancia RPL, de igual forma explora todos los vecinos candidatos para verificar si pueden actuar como enrutadores para un DODAG. Multiples vecinos podrían actuar como enrutadores, y un vecino candidato podría necesitar pasar algunas pruebas de validación antes de que pueda utilizarse como siguiente salto [33]. Este conjunto de DODAG constituye una instancia RPL en la que todos los DODAG usan la misma OF.

- *DODAGID*: Único para un DODAG; su combinación con *RPLInstanceID* puede generar de forma única un DODAG único en la red.

- *DODAGVersionNumber*: Se incrementa cuando una raíz DODAG reconstruye un DODAG. Se puede utilizar para identificar una versión de DODAG cuando se combina con RPLInstanceID y DODAGID.

- *Rank*: Identifica la posición de un nodo individual de acuerdo a su posición con respecto a una raíz DODAG.

En caso de existir más de una raíz DODAG, como se ilustra en la Figura 3.2, que pueden

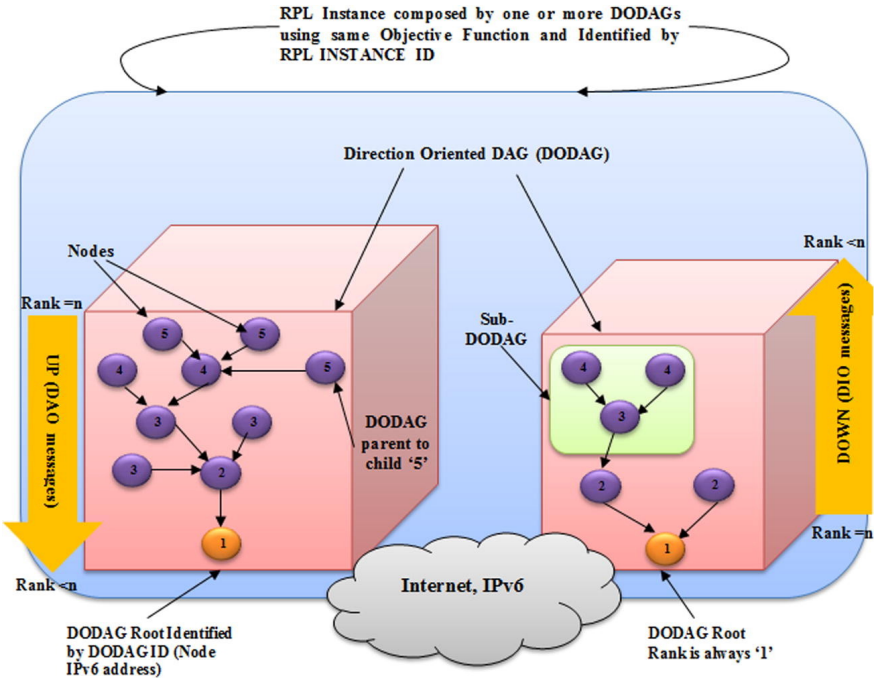


Figura 3.2: *Ilustración de las diferentes operaciones del mecanismo RPL. [18].*

coordinarse a través de una red u operar independientemente, la instancia de RPL tiene como regla proporcionar una ruta a un destino accesible a través de una raíz DODAG [18].

Un árbol (DAG) se crea, mantiene y actualiza mediante diferentes tipos de paquetes ICMPv6, a saber [2]:

- `DIO` (*DODAG Information Object*): Permite publicar información sobre el DODAG; de igual forma permite conocer a sus vecinos y sus rangos.
- `DIS` (*DODAG Information Solicitation Message*): Permite descubrir DODAGs cercanos, también para solicitar mensajes DIOs de los nodos de la red.
- `DAO` (*Destination Advertisement Object*): Sirve para propagar información de rutas y prefijos de una red RPL.

Contiki implementa RPL en dos versiones:

- *RPL Classic*: Implementación original de RPL en su etapa de desarrollo, conocida como ContikiRPL. Soporta gran cantidad de funcionalidades y borradores del protocolo,

lo cual la hace compleja y al desplegarla en un nodo IoT, tiene un gran consumo de memoria ROM.

- *RPL Lite*: Implementación predeterminada en Contiki, presenta un consumo menor de memoria ROM, esto se logra eliminando ciertas características presentes en RPL, como por ejemplo, eliminando el almacenamiento de tablas de enrutamiento. Esta es la opción elegida en el presente trabajo.

3.1.3. ContikiMAC

Debido a que los nodos sensores tienen restricción de recursos y energía, éstos deben de mantener sus radios apagadas el mayor tiempo posible, pero deben de despertar con la debida frecuencia para poder enviar y recibir datos de sus vecinos [7]. Por este motivo, en redes 802.15.4 se usa ampliamente ContikiMAC ya que su alta eficiencia energética y fácil implementación, lo hacen muy conveniente para su uso en nodos con restricciones [14].

ContikiMAC es un protocolo de ciclo de trabajo de radio *radio duty cycle* que utiliza activaciones periódicas para escuchar las transmisiones de paquetes de los vecinos. Si se detecta una transmisión de paquetes durante una activación, el receptor se mantiene encendido para poder recibir el paquete. Cuando el paquete se recibe con éxito, el receptor envía un acuse de recibo a nivel de capa de enlace. Para transmitir un paquete, un remitente envía repetidamente su paquete hasta que recibe un acuse de recibo de la capa de enlace del receptor. Los paquetes que se envían en difusión, no dan lugar a acuses de recibo de la capa de enlace. En cambio, el remitente envía repetidamente el paquete durante el intervalo de activación completo para asegurarse de que todos los vecinos lo hayan recibido. El principio de ContikiMAC se muestra en las Figuras 3.3 y 3.4 [7].

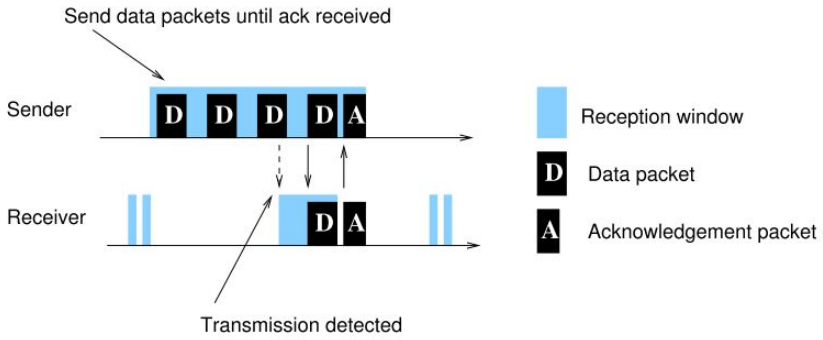


Figura 3.3: *Transmisiones de datos en unicast [7].*

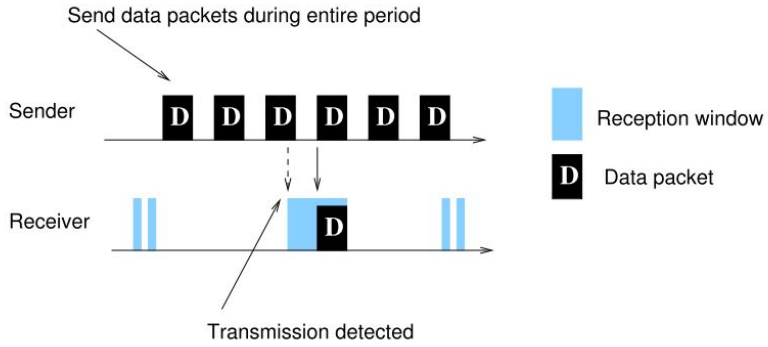


Figura 3.4: *Transmisiones de datos en difusión [7].*

3.2. Cooja

3.2.1. Descripción

Los simuladores para redes de sensores inalámbricos son valiosas herramientas para el desarrollo de sistemas IoT. Sin embargo, los simuladores actuales solo pueden simular un único nivel de un sistema simultáneamente. Esto dificulta el desarrollo y la evolución del sistema, ya que los desarrolladores no pueden usar el mismo simulador tanto para el desarrollo de algoritmos de alto nivel como para el desarrollo de bajo nivel [23].

COOJA es un simulador basado en Java, diseñado para simular redes de sensores que ejecuten el sistema operativo Contiki [21]. Permite la simulación simultánea a nivel de la red, a nivel del sistema operativo y a nivel del conjunto de instrucciones del código de máquina

tal como se muestra en la Figura 3.5. También permite la simulación de niveles cruzados: simulación simultánea a distintos niveles del sistema, combinando la simulación de bajo nivel (hardware del nodo sensor) y la simulación del comportamiento de alto nivel, en una sola simulación. COOJA es flexible y extensible ya que todos los niveles del sistema pueden cambiarse o reemplazarse: plataformas de nodos de sensores, software del sistema operativo, transceptores de radio y modelos de transmisión de radio [23].

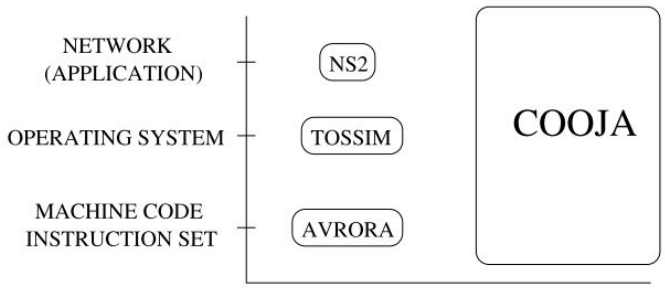


Figura 3.5: *Niveles que permiten simular COOJA [23].*

COOJA permite simular redes de nodos sensores, donde cada nodo puede ser de un tipo diferente; diferenciándose no solo en el software integrado, sino también en el hardware simulado. COOJA es flexible, ya que muchas partes del simulador se pueden reemplazar o extender fácilmente con funcionalidades adicionales. Las partes que se pueden extender incluyen el medio de radio simulado, el hardware de nodo simulado y los complementos para entrada/salida simulada [8, 23].

Al ejecutar programas Contiki, COOJA ejecuta el código de dos maneras [23]:

- *Nativo*: Compilado directamente en la CPU del host.
- *Emulado*: Compilado en un emulador TI MSP430 a nivel de instrucción.

Existen nodos llamados *no Contiki* que son implementados en Java conocidos como nodos *cooja*, que permiten una simulación más rápida, COOJA incluso permite que se ejecuten nodos con un sistema operativo distinto a Contiki. Cada uno de estos enfoques tiene sus

ventajas y desventajas, como por ejemplo que los nodos *cooja* presentan problemas con código desplegable (con esto nos referimos a código que se emplea para una arquitectura específica) [23].

Al emular nodos se proporciona una ejecución más precisa en comparación a nodos *cooja* o nodos que ejecuten código nativo. Sin embargo, los nodos nativos, debido a su estilo de compilación, son más eficientes. Una ventaja de combinar varios niveles de abstracción, es que permite simular algunos nodos a nivel de hardware mientras que el resto se implementan como Java puro, simulándose simplemente su funcionalidad. El uso de este enfoque combina las ventajas de los diferentes niveles [23].

A lo largo de esta sección, se indica que COOJA permite simulaciones simultáneas en tres niveles diferentes [23]:

- *Nivel de red*: Permitir al usuario simular protocolos de enrutamiento, controladores de dispositivos o módulos de medios de radio. Además, los nuevos medios e interfaces de radio, pueden desarrollarse fácilmente en Java y agregarse al entorno de simulación COOJA [23].

- *Nivel de sistema operativo*: Podría denominarse "emulado de alto nivel", ejecuta todo el sistema operativo Contiki, incluidos los procesos del usuario; también es posible alterar la funcionalidad principal de Contiki. Esto es útil, por ejemplo, para probar y evaluar cambios en las bibliotecas Contiki [23].

- *Nivel del conjunto de instrucciones* del código máquina: El uso de un emulador de microcontrolador basado en Java en lugar de un sistema Contiki compilado, permite representar un nodo ESB (*Embedded Sensor Board*), emulado a nivel de bit [13, 23].

Para el cálculo de la calidad de la señal de radio entre nodos, existen distintos métodos basados en distintos indicadores de calidad [30]:

- *Estimadores por software*: Usan el indicador por objetivo o conteo de saltos, con lo que se trata de disminuir los saltos desde el nodo origen al nodo destino. ETX es un

ejemplo de indicador software un poco más complejo, en el se que combina información de múltiples capas OSI para calcular las métricas de enlace [30].

- *Estimadores por hardware*: Usan los indicadores de intensidad de señal recibida (RSSI) y calidad de enlace (LQI) para el cálculo de la métrica de enlace [30].

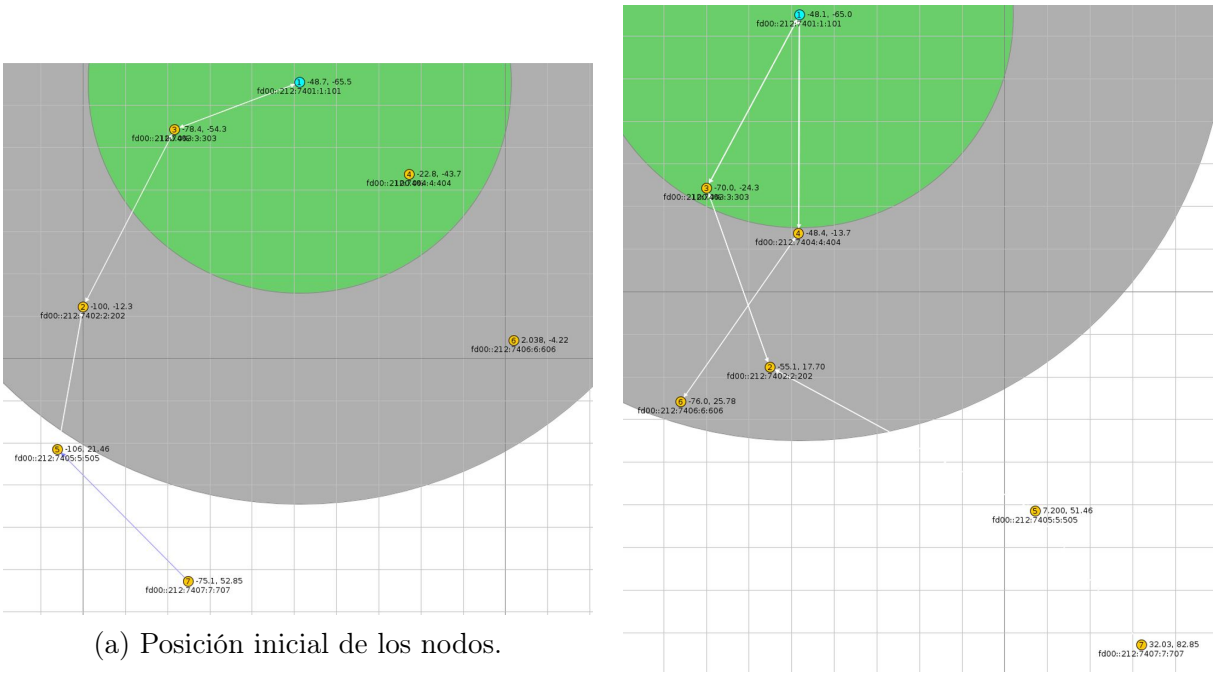
En sus simulaciones, COOJA no considera interferencias de radio externas y usa por defecto el modelo *Unit Disk Graph Medium* (UDGM), que calcula el valor RSSI en función de la distancia entre los nodos y utiliza 37 como valor predeterminado para todos los valores LQI, degradando la calidad de señal de radio en función de la distancia entre cada nodo [30].

3.2.2. El plugin Mobility

En las secciones anteriores se ha visto la potencialidad de Contiki como sistema operativo para nodos IoT, RPL como protocolo de enrutamiento en redes con pérdida y COOJA como entorno de simulación. Muchas de estas simulaciones o implementaciones se basan en nodos estáticos, con posiciones definidas e invariables en el tiempo. Dadas las características de este trabajo, es indispensable poder realizar simulaciones de nodos móviles, por lo que se ha recurrido a Mobility para tal fin. Mobility es un plugin desarrollado por Fredrik Osterlind, que se puede descargar mediante el siguiente link [22], el cual nos permite dotar a los nodos estáticos de movilidad.

El plugin Mobility permite cambiar de posición cada nodo con respecto al tiempo de simulación basándose en un archivo con extensión `.dat`, el cual debe presentar la siguiente estructura:

- Primera columna: Identificador del nodo.
- Segunda columna: Marca de tiempo en la simulación.
- Tercera columna: Posición del nodo en el eje X.
- Cuarta columna: Posición del nodo en el eje Y.



(a) Posición inicial de los nodos.

(b) Posición en el instante $t = 1s$ de simulación.

Figura 3.6: *Movimiento de nodos IoT.*

La Figura 3.6 muestra el comportamiento de un grupo de 6 nodos móviles, que corresponde a las posiciones marcadas en un archivo con extensión .dat con la estructura requerida.

1	0	-100	-12.3
2	0	-78.4	-54.3
3	0	-22.8	-43.7
4	0	-106	21.46
5	0	2.038	-4.22
6	0	-75.1	52.85
1	1	-55.1	17.7
2	1	-70	-24.3
3	1	-48.4	-13.7
4	1	7.2	51.46
5	1	-76	25.78
6	1	32.038	82.85

3.3. Entorno de simulación

En la Figura 3.7 se aprecia el modelo de arquitectura para los distintos escenarios de simulación, explicados con más detalle en la Sección 4. En los experimentos se hace uso del simulador COOJA junto el sistema operativo ContikiOS en su versión 2.7.

Cada nodo cliente emulado envía paquetes de información a un servidor UDP, en tiempos regulares. Uno o más routers de borde tienen como tarea servir de puente entre el simulador COOJA y la red externa.

El servidor UDP envía cada uno de los paquetes recibidos de cada nodo cliente a un servidor de base de datos para su almacenamiento, de igual forma estos paquetes recibidos son tratados por un servidor NODE-RED para su visualización en un *dashboad*.

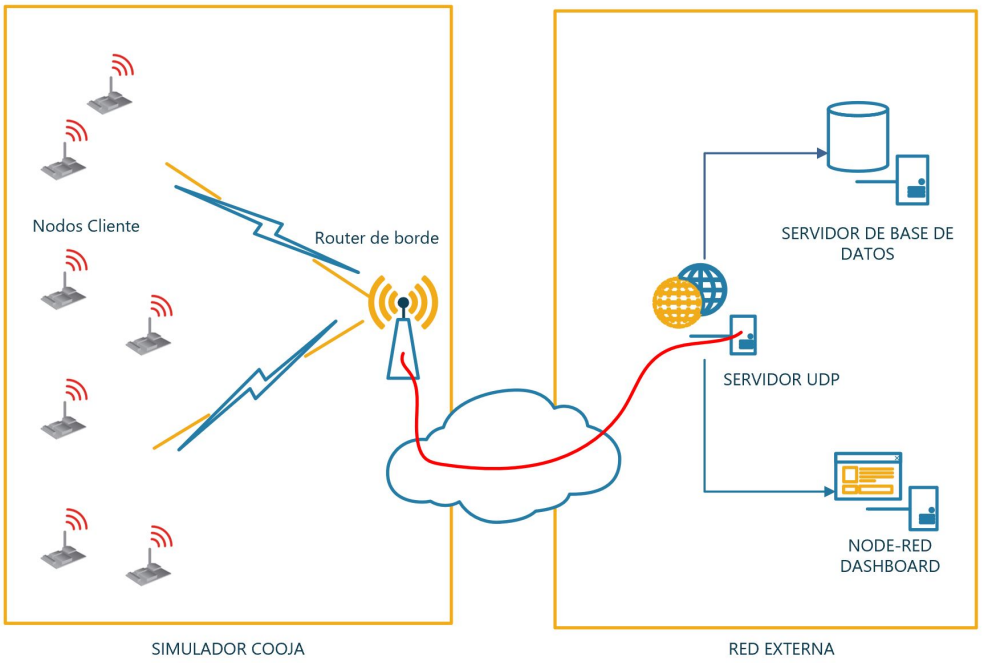


Figura 3.7: *Modelo de arquitectura planteada*

3.3.1. Motas cliente

Tmote Sky es un nodo IoT desarrollado originalmente por la Universidad de California, Berkeley, con un consumo ultra bajo de potencia, con $10kB$ de memoria RAM y $48kB$ de

memoria flash. La Figura 3.8 muestra en detalle un nodo Sky de la empresa Moteiv, el cual utiliza para sus comunicaciones una radio Chipcon CC2420 que tiene un ámbito de recepción de 50 metros en ambientes cerrados y 150 metros en ambiente exteriores. Según los patrones de distancia individual y social vistos, estos parámetros son más que suficientes para el escenario objetivo.

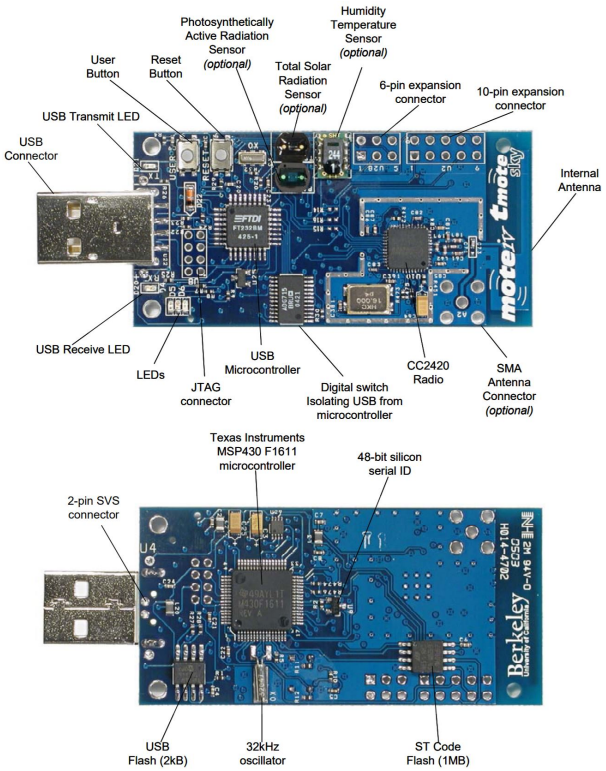
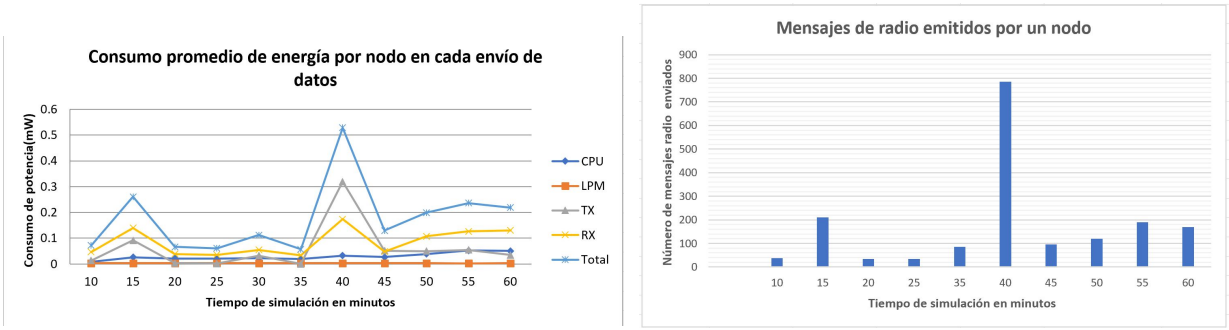


Figura 3.8: *Tmote Sky [27].*

Para un nodo Sky, a pesar de tener un ultra bajo consumo de potencia, se recomienda mantener su uso al mínimo. Para tal fin, se puede apagar y encender la radio en un intervalo preestablecido, ésto se logra con el uso de la biblioteca ContikiMAC (fichero de cabecera `contikimac.h`) y las instrucciones `mac->off(0)` y `mac->on()`.

ContikiOS permite medir el consumo aproximado de energía de un nodo, mediante la biblioteca Powertrace (fichero de cabecera `powertrace.h`). La instrucción `powertrace_start` examina el estado de potencia en un intervalo de tiempo definido, siendo los estados a



(a) Consumo de energía de un nodo.

(b) Número de mensajes de radio emitidos por un nodo.

Figura 3.9: *Gráficas comparativas entre el consumo de energía de un nodo y el número de mensajes de radio emitidos por éste.*

examinar [16]:

- CPU: Consumo de energía de la CPU.
- LPM: Consumo de energía en modo de baja potencia.
- TRANSMIT: Consumo de energía de transmisión de datos.
- LISTEN: Consumo de energía de recepción de datos.
- IDLE_TRANSMIT: Consumo de energía de transmisión de datos en modo de reposo.
- IDLE_LISTEN: Consumo de energía de recepción de datos en modo de reposo.

La Figura 3.9 (a), corresponde al consumo de energía en cada envío de datos del nodo 1 que enciende/apaga su radio en intervalos predefinidos en la simulación realizada en la Sección 4.2, esta medición se realiza con Powertrace. Se aprecia un pico de energía transcurridos 45 minutos de simulación, esto guarda correlación con la cantidad de mensajes de radio emitidos por el nodo tal como se aprecia en la Figura 3.9 (b), y que se debe a que el nodo se encuentra aislado de otros nodos.

3.3.2. Motas router de borde

Un router de borde permite realizar la adaptación de la capa de 6LowPAN para su enrutamiento en redes IPv6 convencionales; el router de borde no realiza traducción de tecnología, solo cambia algunas capas como se aprecia en la Figura 3.10. Los paquetes que se han transmitido en la red de sensores 6LowPAN mantendrán inmutable la información de las capas superiores.

Un router de borde también se encargará de autoconfigurarse como la raíz de un DODAG y de suministrar un prefijo IPv6 a todos los nodos clientes que forman parte del DODAG.

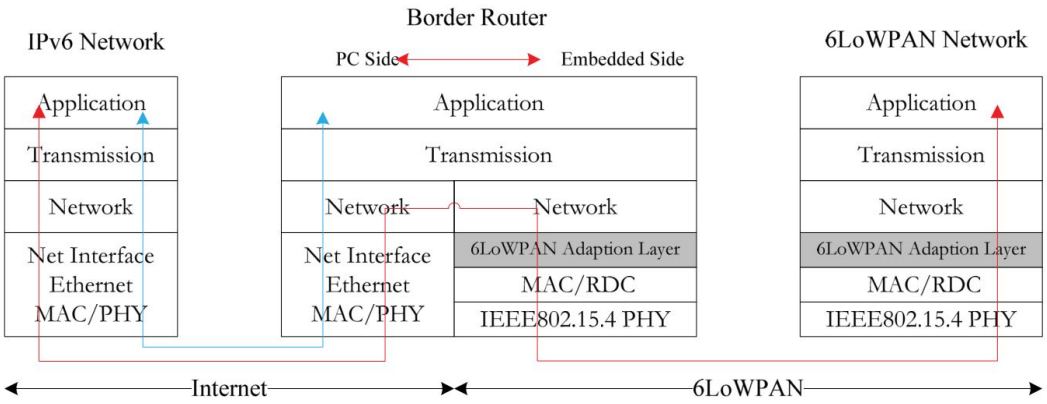


Figura 3.10: *Router de borde en el stack de protocolo [11].*

3.3.3. Servidor

En una primera instancia se utiliza un servidor UDP implementado en Python 2.7 que escucha peticiones de clientes por el puerto 5678, las cuales son mostradas por consola, como se aprecia en la Figura 3.11. Una mejor opción, implementada en segunda instancia, consiste en complementar la funcionalidad de un servidor UDP con un servidor Node-Red, el cual permite implementar un *dashboard* para tener una forma más amigable de visualizar la información por parte del usuario, como se aprecia en la Figura 3.12.

La información recolectada de cada nodo cliente debe ser almacenada en un medio persistente, como puede ser un archivo de texto, archivo de valores separados por comas (`csv`)

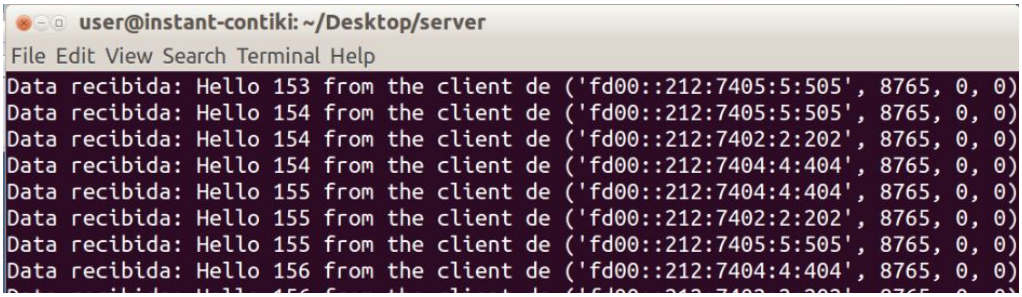


Figura 3.11: *Visualización de los datos por consola recibidos por el servidor UDP.*

o, como opción más conveniente, almacenada en una base de datos.

3.3.4. Recogida de datos

Los datos recibidos por cada nodo cliente se almacenan en una base de datos no relacional, implementada en un servidor MongoDB v4.2.6. Como herramienta de gestión de dicha base de datos se usa el programa MongoDB Compas v1.21.2 con el cual se realiza la descarga de los datos, previamente almacenados, para su posterior análisis. La Figura 3.13 nos muestra la lectura de datos de algunos nodos.

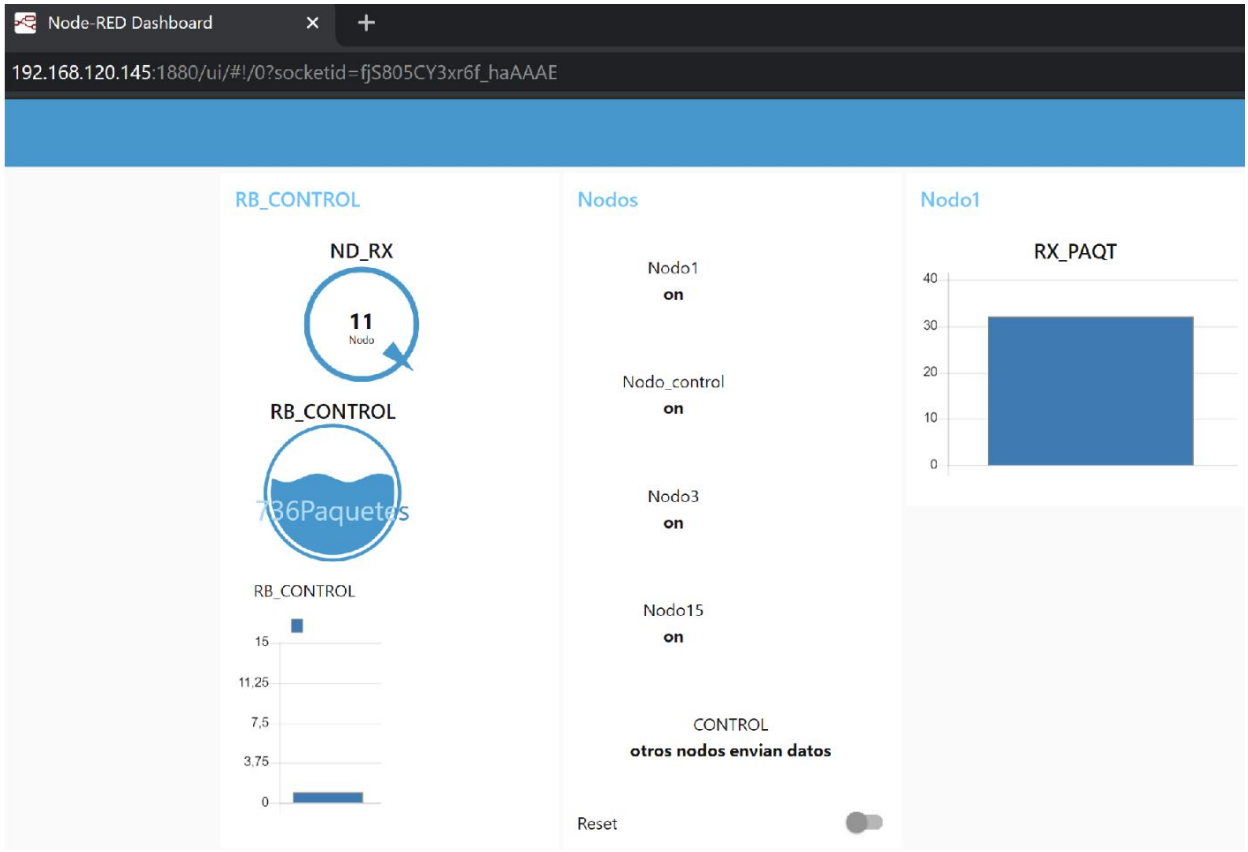


Figura 3.12: *Dashboard implementado en Node-Red.*

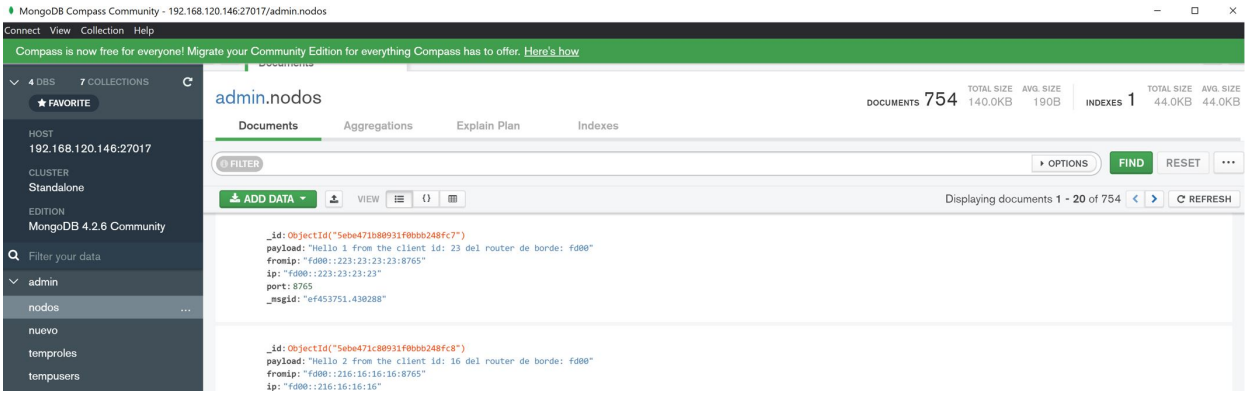


Figura 3.13: *Lectura de los datos utilizando el programa MongoDB Compas.*

Capítulo 4

Resultados experimentales

En el presente capítulo se expone los resultados de cuatro escenarios diferentes, considerados representativos de los escenarios típicos e ilustrativos de la aplicación planteada:

- **Escenario 1:** Impacto de nodos móviles, con movimientos predefinidos sobre RPL.
- **Escenario 2:** Uso de múltiples routers de borde para la sensorización con nodos móviles que siguen el movimiento de un grupo familiar de vicuñas.
- **Escenario 3:** Uso de nodos estáticos para aumentar el alcance del escenario 2.
- **Escenario 4:** Uso de un router de borde con múltiples nodos estáticos que sirven como extensión de su alcance.

Los códigos y configuración de cada uno de los escenarios se detallan en el apéndice B.

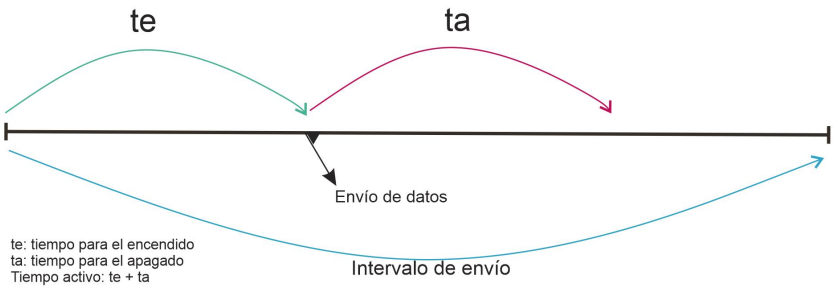


Figura 4.1: *Tiempo de encendido/apagado de radio, en cada envío de dato.*

4.1. RPL con nodos móviles

4.1.1. Uso de ContikiMAC para minimizar el consumo de energía.

En esta sección se evalúa el impacto que tiene la acción de dormir (apagar radio) y despertar (encender radio) un nodo, sobre el consumo de energía y la tasa de recepción de paquetes enviados por las motas.

El escenario plantea la medición de este impacto en una simulación con seis nodos clientes y un router de borde, sin dinámica de movimiento; su distribución espacial se aprecia en la Figura 4.2, y los parámetros utilizados se indican el Cuadro 4.1. La acción de dormir/-despertar un nodo, se obtendrá con el uso de la biblioteca ContikiMAC (fichero de cabecera `contikimac.h`) y las instrucciones `mac->off(0)` y `mac->on()`. El encendido de la radio se realizará en cada intervalo de envío de datos, 5 segundos después se realizará el envío de datos y el apagado de radio posterior transcurridos 25 segundos; la Figura 4.1 ilustra estos tiempos.

El Cuadro 4.2 muestra los resultados de una simulación sin encendido/apagado de radio, con una tasa de recepción del 100 % de paquetes y un promedio de consumo de energía de $0,27mW$ por nodo en cada envío de datos; la Figura 4.3 muestra el histórico de consumo de energía de cada nodo. En el Cuadro 4.3 se evidencia que con la acción de encendido/apagado de radio, la tasa de recepción se mantiene en 100 % pero con un consumo de energía de $0{,}061mW$ por nodo, que corresponde a una reducción promedio del 77 %; la Figura 4.4 muestra el histórico de consumo de energía de cada nodo.

Parámetro	Valor
Simulador	COOJA sobre ContikiOS 2.7
Ambiente de radio	Unit disk graph medium (UDGM)
Tiempo de simulación	20 minutos
Nodos clientes	6
Router de borde	1
Tipo de mota	Sky
Ciclo de trabajo de radio	ContikiMAC
Radio RX	50 metros
Intervalo de envío de datos	5 minutos
Tiempo de encendido de radio	5 segundos
Tiempo para el apagado de radio	25 segundos

Cuadro 4.1: *Parámetros de simulación.*

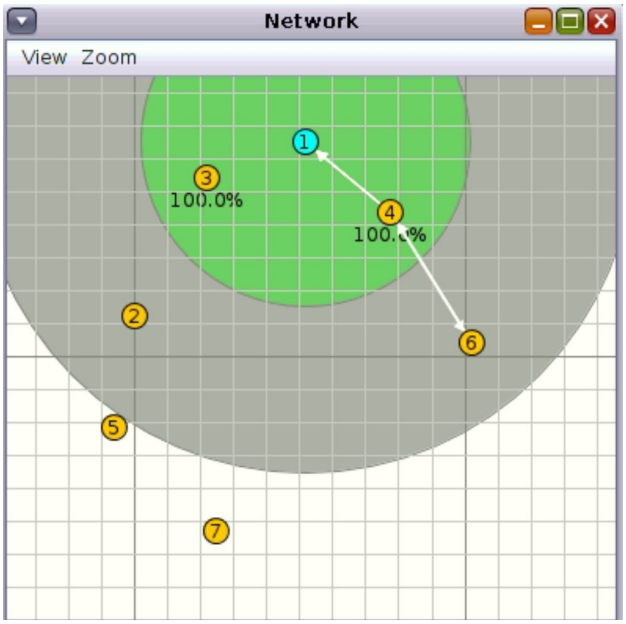


Figura 4.2: *Distribución espacial de nodos estáticos.*

4.1.2. Nodos móviles.

En esta sección se evalúa el impacto sobre la tasa de recepción de paquetes, enviados por nodos *móviles* usando RPL como protocolo de enrutamiento. Para dotar de movilidad a cada nodo cliente se hará uso del plugin *Mobility* y su correspondiente archivo de posición *escenario1.dat*. La Figura 4.5 muestra las tres distintas posiciones de los nodos clientes en

Nodo	Paquetes enviados	Tasa de recepción	Consumo de energía (mW)
2	4	100 %	0,40
3	4	100 %	0,38
4	4	100 %	0,39
5	4	100 %	0,08
6	4	100 %	0,20
7	4	100 %	0,19

Cuadro 4.2: *Resultados de la simulación sin encendido/apagado de radio.*

Nodo	Paquetes enviados	Tasa de recepción	Consumo de energía (mW)
2	4	100 %	0,072
3	4	100 %	0,067
4	4	100 %	0,058
5	4	100 %	0,065
6	4	100 %	0,055
7	4	100 %	0,55

Cuadro 4.3: *Resultados de la simulación con encendido/apagado de radio.*

la simulación; el Cuadro 4.4 indica los parámetros de simulación.

El Cuadro 4.5 muestra los resultados de una simulación, con una tasa de recepción promedio del $48,15\,\%$ de paquetes y un promedio de consumo de energía por nodo en cada ciclo de envío de $0,092mW$.

La baja tasa de recepción obedece a que en la tercera posición espacial de los nodos clientes, no se forman correctamente su DODAG. La Figura 4.6 (a) muestra que el nodo 4 trata de enviar su información directamente al router de borde; debido a que está fuera de rango de recepción de la radio, el paquete se pierde. En la Figura 4.6 (b) se muestra el comportamiento deseado, donde el nodo 4 enviará su información a través del nodo 3.

Una solución para lograr dicho comportamiento es usar `rpl_repair_root` en el router de borde, esto forzará en cada ciclo de envío de datos la reconstrucción del DODAG. Lo que incrementará el DODAGVersionNumber. Esto inicia una nueva versión de DODAG. Los nodos clientes en la nueva versión de DODAG pueden elegir una nueva posición cuyo rango no sea restringido por su rango dentro de la versión anterior de DODAG [33].

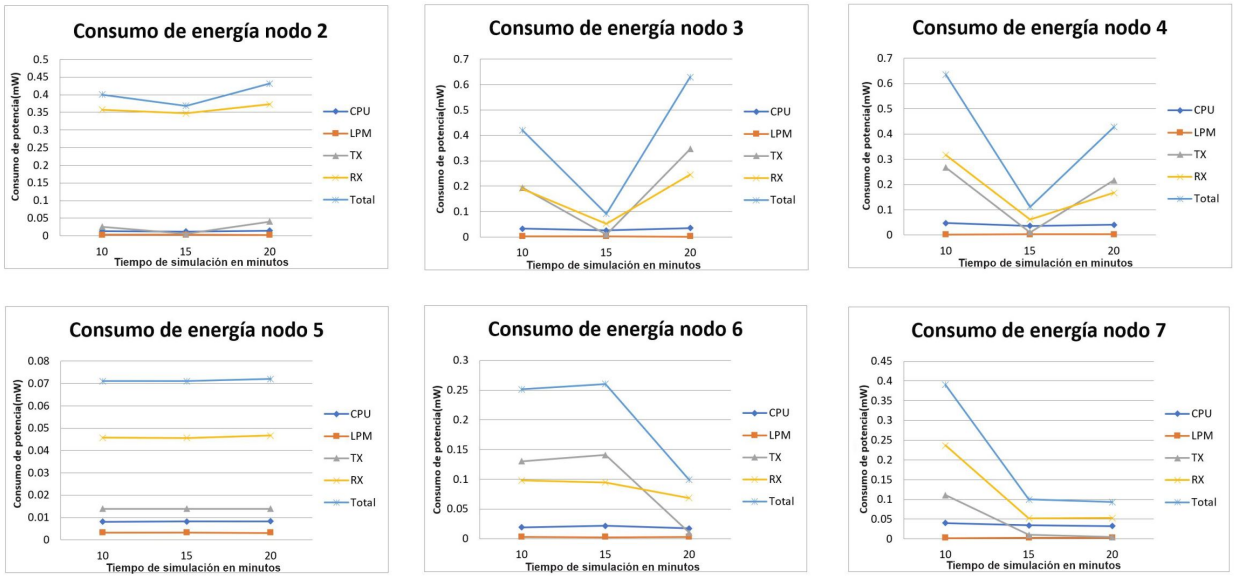


Figura 4.3: *Histórico de consumo de energía sin encendido/apagado.*

El Cuadro 4.6 muestra los resultados de la simulación, con una tasa de recepción promedio del $88,89\,\%$ de paquetes y un promedio de consumo de energía por nodo en cada envío de datos de $0,21mW$ por nodo, se observa un aumento del $40{,}14\,\%$ de tasa de recepción, se aprecia que el uso de reparación global tiene un impacto en el consumo de energía, con un aumento promedio por nodo en cada envío de datos de $0{,}118mW$.

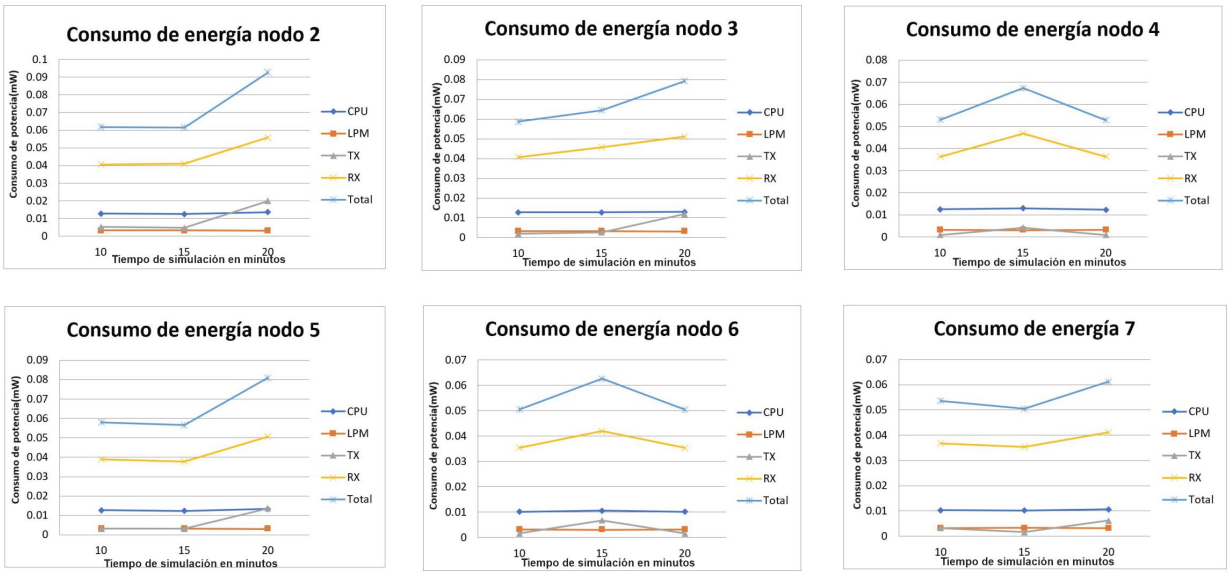


Figura 4.4: *Histórico de consumo de energía con encendido/apagado.*

Parámetro	Valor
Simulador	COOJA sobre ContikiOS 2.7
Ambiente de radio	Unit disk graph medium (UDGM)
Tiempo de simulación	45 minutos
Nodos clientes	6
Router de borde	1
Tipo de mota	Sky
Ciclo de trabajo de radio	ContikiMAC
Archivo de posición	escenario1.dat
Radio RX	50 metros
Intervalo de envío de datos	5 minutos
Tiempo de encendido de radio	15 segundos
Tiempo para el apagado de radio	25 segundos

Cuadro 4.4: *Parámetros de simulación, nodos móviles.*

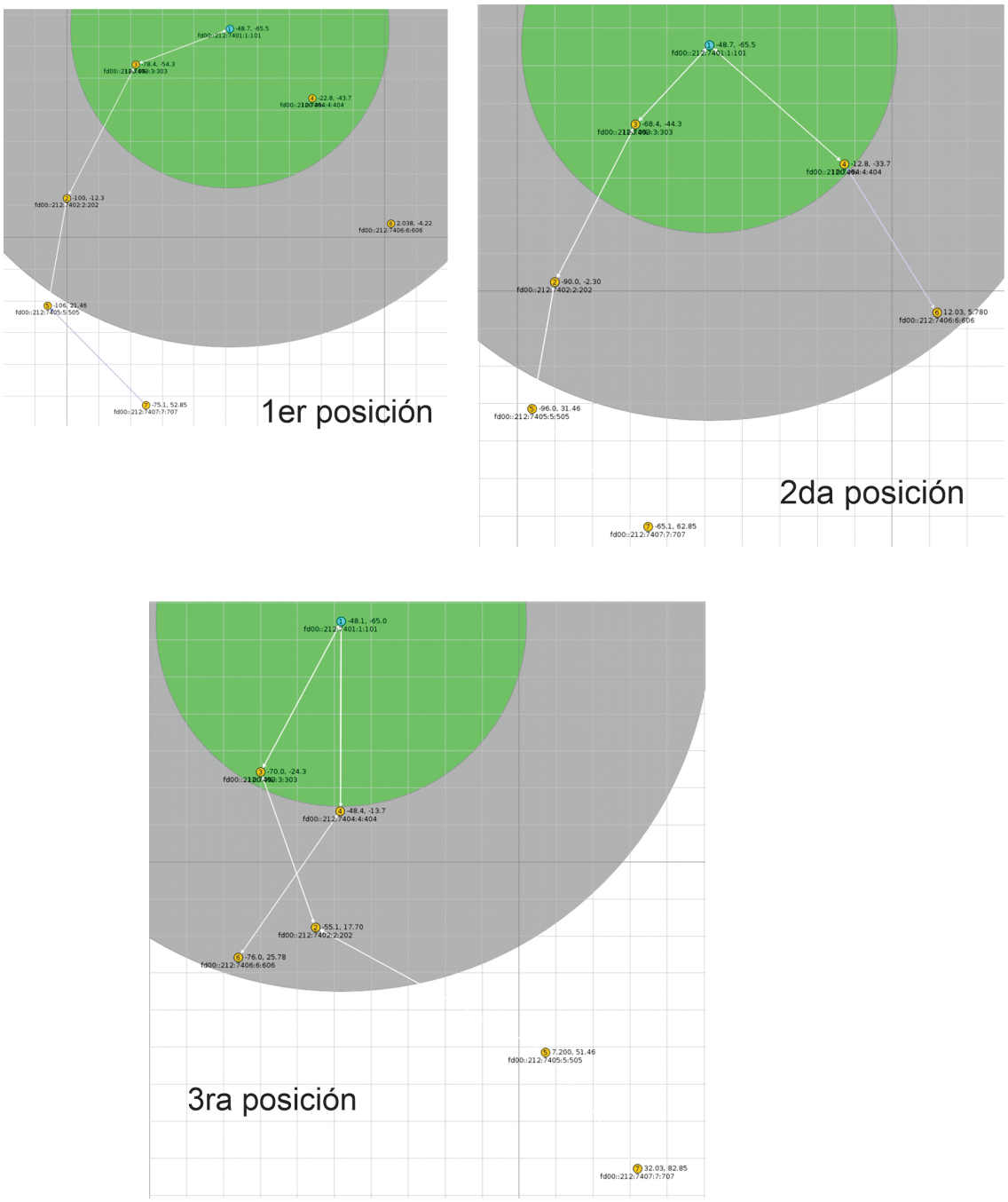


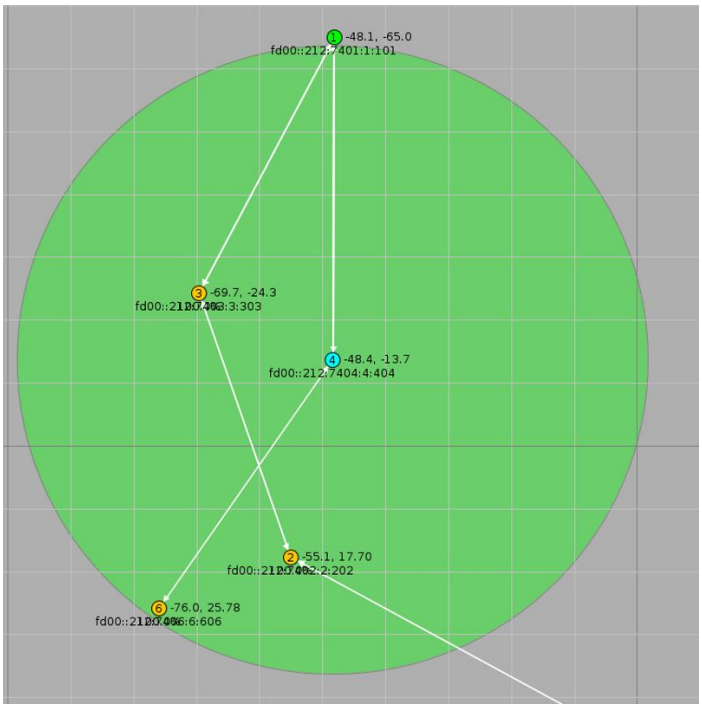
Figura 4.5: *Posiciones de los nodos en la simulación.*

Nodo	Paquetes enviados	Ratio de recepción	Consumo de energía (mW)
2	9	100 %	0,076
3	9	100 %	0,076
4	9	11,11 %	0,068
5	9	33,33 %	0,167
6	9	11,11 %	0,066
7	9	33,33 %	0,103

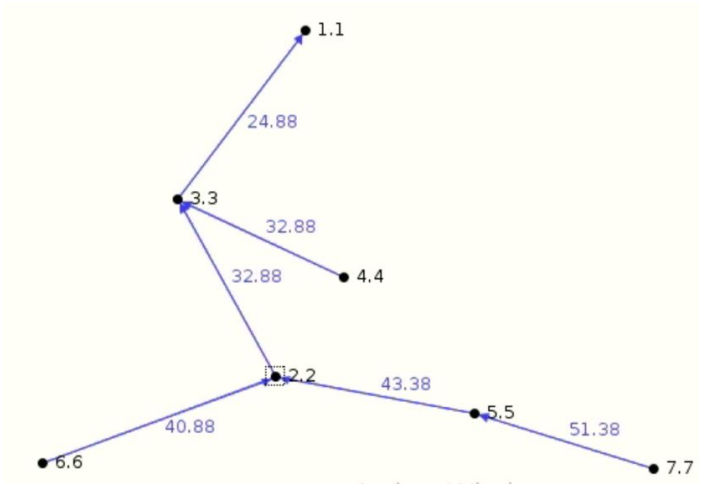
Cuadro 4.5: *Resultados de la simulación sin uso de reparación DODAG.*

Nodo	Paquetes enviados	Ratio de recepción	Consumo de energía (mW)
2	9	100 %	0,26
3	9	100 %	0,18
4	9	100 %	0,29
5	9	66,67 %	0,21
6	9	100 %	0,21
7	9	66,67 %	0,11

Cuadro 4.6: *Resultados de la simulación con uso de reparación DODAG.*



(a) DODAG incorrecto.



(b) DODAG correcto.

Figura 4.6: *Comportamiento en la formación del DODAG en nodos móviles.*

4.2. Múltiples routers de borde

En la sección 4.1 se muestra el impacto sobre la tasa de recepción de paquetes y el consumo de energía, al dormir/despertar un nodo usando ContikiMAC. Usar reparación global de un DODAG en cada envío de datos, también presenta un impacto. En esta sección se enfatiza en cada uno de esos enfoques sobre un entorno de ganadería, más específicamente en un entorno familiar de vicuñas, con el objetivo de obtener el mejor rendimiento sobre la recepción de datos y consumo energético.

Para la recepción de paquetes enviados por cada sensor instalado en una vicuña se emplearon múltiples routers de borde. La posición de cada router de borde se basó en el movimiento que sigue un grupo familiar tal como se aprecia en la Figura 4.7. A través de una imagen satelital obtenida mediante Google Earth (Figura 4.8) se muestra el área de estudio de la dinámica de movimiento de vicuñas descrito en la Sección 2.1. Ambas imágenes no coinciden y ello se debe a que en COOJA las coordenadas Y se encuentran invertidas.

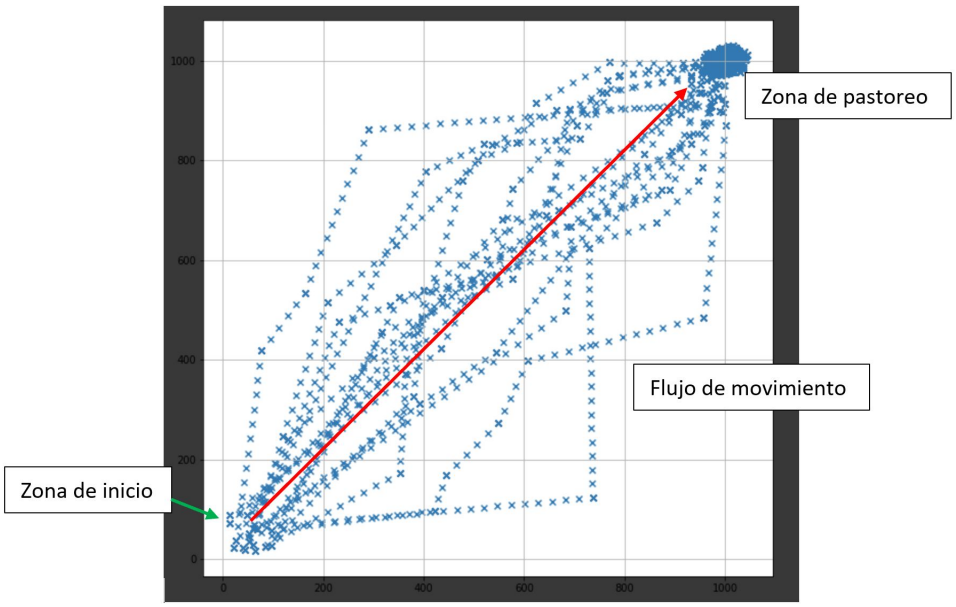


Figura 4.7: *Movimiento pseudo aleatorio hasta llegar a una zona de pastoreo.*

El Cuadro 4.7 indica los parámetros de simulación. Los nodos cliente representan a un grupo familiar de vicuñas.

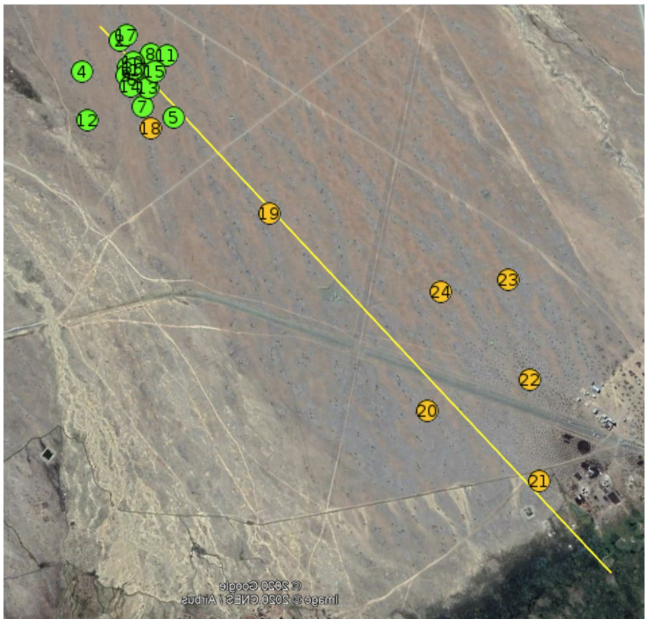


Figura 4.8: *Ubicación de los routers de borde en la simulación.*

Se han realizado cuatro simulaciones por cada valor de tiempo de encendido de radio (Te); los resultados presentados serán un promedio de cada simulación. Debido al número de nodos usados y repeticiones de simulación, es necesario automatizar la ejecución. El Apéndice A muestra en detalle como realizarlo.

Los Cuadros 4.8 y 4.9 muestran los resultados obtenidos. La Figura 4.9 muestra el comportamiento de la tasa de recepción de paquetes, en caso de no usar reparación de DODAG, al aumentar el valor de tiempo de encendido de radio, se observa un aumento en promedio del 13 % en la tasa de recepción de datos, que guarda una correlación con el consumo promedio de energía por nodo en cada envío de datos. En contraparte, al usar reparación de DODAG esta tasa de aumento es menor y a partir de 30 segundos como tiempo de encendido de radio se aprecia una disminución de la tasa, pero en el caso del consumo de energía promedio por nodo en cada envío de datos no ocurre lo mismo, manteniéndose la tendencia de aumento en cada valor. La Figura 4.10 muestra el porcentaje de tiempo activo en cada envío y tiempo activo de los nodos cliente durante la simulación.

El uso de reparación global de DODAG tiene un impacto significativo en la tasa de

Parámetro	Valor
Simulador	COOJA sobre ContikiOS 2.7
Ambiente de radio	Unit disk graph medium (UDGM)
Tiempo de simulación	120 minutos
Nodos clientes	17
Router de borde	7
Tipo de mota	Sky
Ciclo de trabajo de radio	ContikiMAC
Archivo de posición	position.dat
Radio RX	150 metros
Intervalo de envío de datos	5 minutos
Tiempo de encendido de radio (Te)	5, 15, 30, 60 segundos
Tiempo para el apagado de radio	25 segundos

Cuadro 4.7: *Parámetros de simulación, múltiples routers de borde.*

Te	Paquetes enviados	Tasa de recepción	Consumo de energía (mW)
5s	391	27,62 %	0,172
15s	391	39,64 %	0,193
30s	391	59,08 %	0,213
60s	391	66,75 %	0,271

Cuadro 4.8: *Resultados de la simulación para múltiples routers de borde sin reparación DODAG.*

recepción de paquetes y sobre el consumo de energía. Los mejores resultados en este escenario se obtienen con un tiempo de encendido de radio de 15 segundos, que ofrece una tasa de recepción de 75,70 % y un consumo de energía promedio por nodo en cada envío de datos de 0,384mW, cabe resaltar que, con un tiempo de 60 segundos, se obtiene una tasa mayor de recepción, pero a cambio presenta un aumento en el consumo energético promedio por nodo del 18 %, lo cual no es recomendable teniendo en cuenta que el aumento en la tasa de recepción en comparación al anterior es de solo un 4,1 %.

El Cuadro 4.10 muestra la tasa de recepción obtenida por cada router de borde. El router de borde con ID 21 es el responsable de recabar la mayor parte de paquetes enviados. La Figura 4.8 muestra su ubicación que corresponde a zona de pastoreo.

Te	Paquetes enviados	Tasa de recepción	Consumo de energía (mW)
5s	391	65,22 %	0,301
15s	391	74,42 %	0,318
30s	391	76,98 %	0,369
60s	391	68,80 %	0,404

Cuadro 4.9: *Resultados de la simulación para múltiples routers de borde con reparación DODAG.*

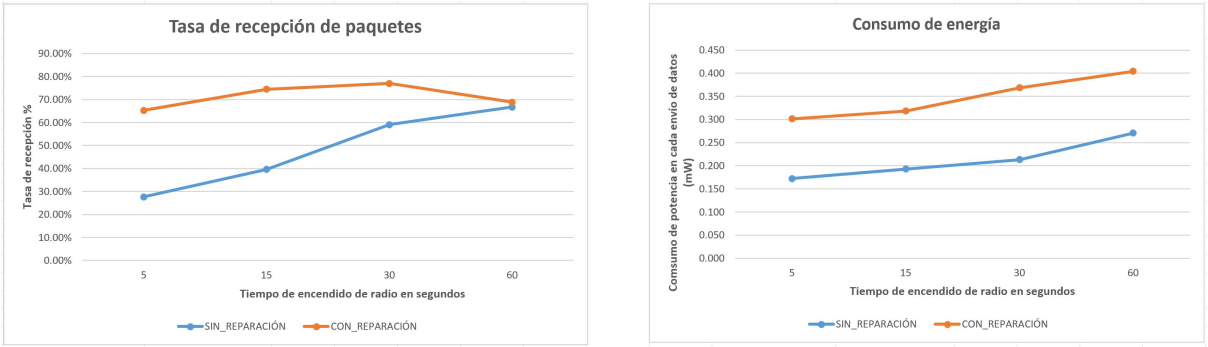


Figura 4.9: *Comportamiento de tasa de recepción y consumo de energía, usando múltiples routers de borde.*

ID Router	Tasa de recepción
18	2,41 %
19	2,75 %
20	1,03 %
21	82,82 %
22	5,15 %
23	0,34 %
24	5,50 %

Cuadro 4.10: *Tasa de recepción de datos por router de borde.*

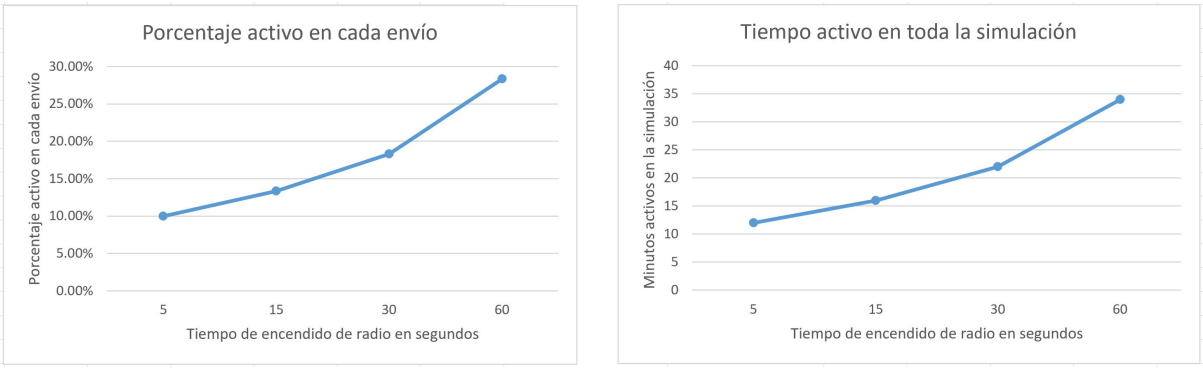


Figura 4.10: *Porcentaje de tiempo activo en cada envío y tiempo activo en toda la simulación.*

4.3. Múltiples routers de borde en conjunto con nodos estáticos

En la sección 4.2 se emplearon múltiples routers de borde obteniendo un buen resultado, con un tiempo para el encendido de radio de 15 segundos y reparación global de un DODAG. El Cuadro 4.10 indica que el router de borde con ID 21 recaba la mayor parte de paquetes enviados.

En esta sección se hará el uso de nodos estáticos para aumentar el alcance de cada router de borde, tal como se aprecia en la Figura 4.11. El objetivo buscado es no solo aumentar la tasa de recepción general, sino también el de cada router de borde. El Cuadro 4.11 indica los parámetros de simulación.

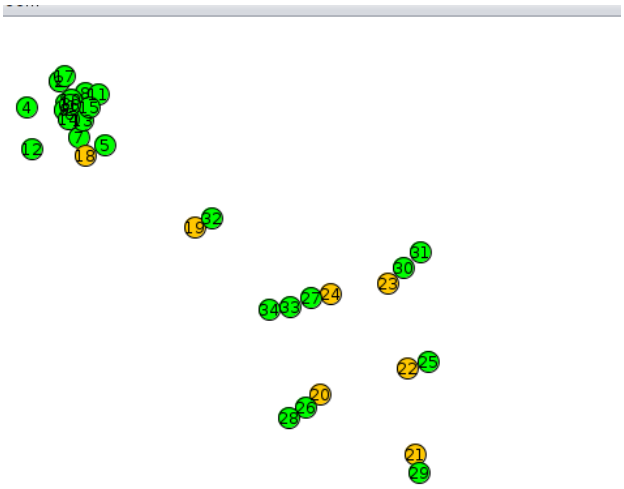


Figura 4.11: *Distribución espacial de nodos estáticos en la simulación.*

Los Cuadros 4.12 y 4.13 muestran los resultados obtenidos. La Figura 4.12 muestra el comportamiento de la tasa de recepción de paquetes con respecto al tiempo para el encendido de radio, en caso de no usar reparación de DODAG, al aumentar el valor de tiempo de encendido de radio, se observa un aumento en promedio del $7,67\,\%$ en la tasa de recepción de datos, que guarda una correlación con el consumo promedio de energía por nodo en cada envío de datos. Sin embargo, al usar reparación de DODAG esta tasa de aumento es menor

Parámetro	Valor
Simulador	COOJA sobre ContikiOS 2.7
Ambiente de radio	Unit disk graph medium (UDGM)
Tiempo de simulación	120 minutos
Nodos clientes	17
Nodos estáticos de reenvío	10
Router de borde	7
Tipo de mota	Sky
Ciclo de trabajo de radio	ContikiMAC
Archivo de posición	position.dat
Radio RX	150 metros
Intervalo de envío de datos	5 minutos
Tiempo de encendido de radio (Te)	5, 15, 30, 60 segundos
Tiempo para el apagado de radio	25 segundos

Cuadro 4.11: *Parámetros de simulación, múltiples routers de borde en conjunto con nodos estáticos.*

a partir de 15 segundos como tiempo de encendido de radio, pero en el caso del consumo de energía promedio por nodo en cada envío de datos no sucede lo mismo, la tasa de aumento es más pronunciada.

Te	Paquetes enviados	Tasa de recepción	Consumo de energía (mW)
5s	391	41,18 %	0,182
15s	391	50,13 %	0,209
30s	391	51,92 %	0,223
60s	391	64,19 %	0,291

Cuadro 4.12: *Resultados de la simulación para múltiples routers de borde con nodos estáticos sin reparación DODAG.*

El Cuadro 4.14 muestra la tasa de recepción obtenida por cada router de borde. El router de borde con ID 21 es el responsable de recabar la mayor parte de paquetes enviados. La Figura 4.8 muestra su ubicación que corresponde a zona de pastoreo.

Te	Paquetes enviados	Tasa de recepción	Consumo de energía (mW)
5s	391	62,66 %	0,308
15s	391	75,70 %	0,374
30s	391	78,01 %	0,384
60s	391	79,8 %	0,460

Cuadro 4.13: *Resultados de la simulación para múltiples routers de borde con nodos estáticos con reparación DODAG.*

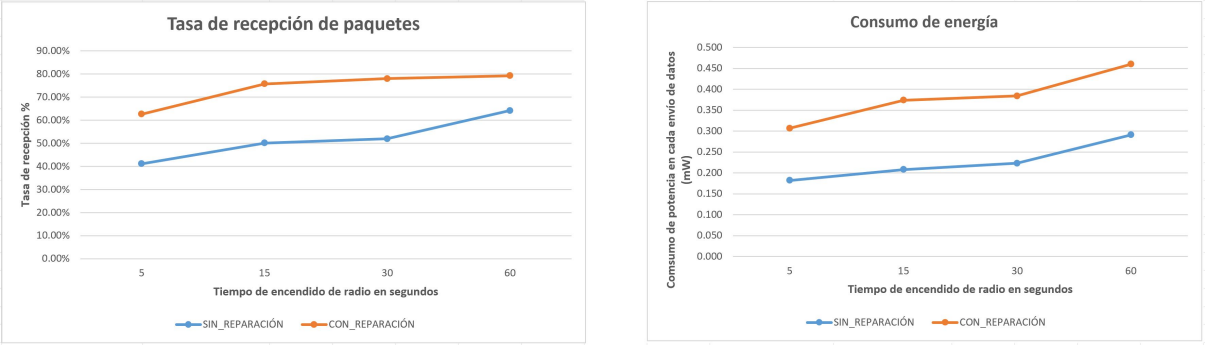


Figura 4.12: *Comportamiento de tasa de recepción y consumo de energía, usando múltiples routers de borde en conjunto con nodos estáticos.*

ID Router	Tasa de recepción
18	2,36 %
19	3,38 %
20	1,35 %
21	81,08 %
22	4,39 %
23	0,34 %
24	7,09 %

Cuadro 4.14: *Tasa de recepción de datos por router de borde usando nodos estáticos.*

4.4. Router de borde con nodos estáticos

Los resultados obtenidos en la Sección 4.2 y 4.3, nos indican que el router de borde con ID 21 es el responsable de recabar aproximadamente el 81 % de todos los paquetes enviados. Esta alta tasa de recepción nos plantea un escenario con un único router de borde en conjunto con múltiples nodos estáticos para aumentar su radio de recepción. La Figura 4.13 muestra la distribución espacial de los nodos. El Cuadro 4.15 indica los parámetros de simulación.

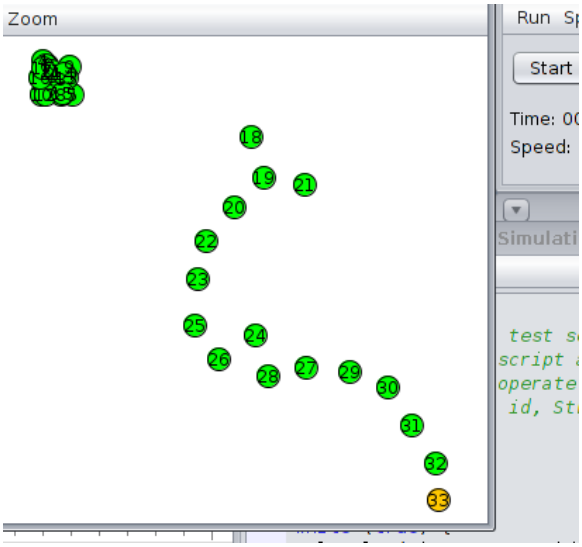


Figura 4.13: *Un router de borde en conjunto con nodos estáticos.*

Los Cuadros 4.16 y 4.17 muestran los resultados obtenidos. La Figura 4.14 muestra el comportamiento de la tasa de recepción de paquetes con respecto al tiempo para el encendido de radio, en caso de no usar reparación de DODAG, se observa un caso atípico en comparación a los anteriores escenarios, en donde, al aumentar el valor de tiempo de encendido de radio, se aprecia una disminución en la tasa de recepción ha partir de 15 segundos como tiempo de encendido de radio, pero en el caso del consumo de energía promedio por nodo en cada envío de datos no sucede los mismo, se mantiene la tendencia de aumento en cada valor.

Parámetro	Valor
Simulador	COOJA sobre ContikiOS 2.7
Ambiente de radio	Unit disk graph medium (UDGM)
Tiempo de simulación	120 minutos
Nodos clientes	17
Nodos estáticos de reenvío	15
Router de borde	1
Tipo de mota	Sky
Ciclo de trabajo de radio	ContikiMAC
Archivo de posición	position.dat
Radio RX	150 metros
Intervalo de envío de datos	5 minutos
Tiempo de encendido de radio (Te)	5, 15, 30, 60 segundos
Tiempo para el apagado de radio	25 segundos

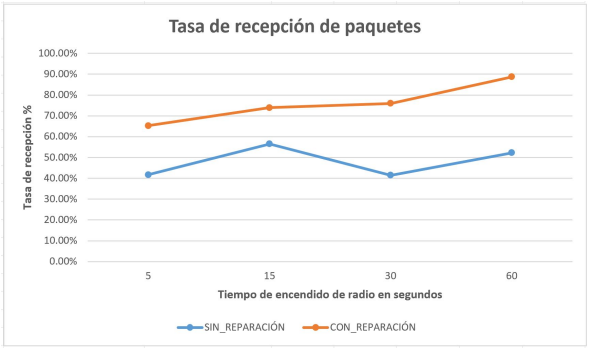
Cuadro 4.15: *Parámetros de simulación, un router de borde en conjunto con nodos estáticos.*

Te	Paquetes enviados	Tasa de recepción	Consumo de energía (mW)
5s	391	41,69 %	0,185
15s	391	56,52 %	0,167
30s	391	41,43 %	0,221
60s	391	52,24 %	0,295

Cuadro 4.16: *Resultados de la simulación para un router de borde con nodos estáticos sin reparación DODAG.*

Te	Paquetes enviados	Tasa de recepción	Consumo de energía (mW)
5s	391	65,22 %	0,207
15s	391	73,91 %	0,306
30s	391	75,96 %	0,346
60s	391	85,75 %	0,437

Cuadro 4.17: *Resultados de la simulación para un router de borde con nodos estáticos con reparación DODAG.*



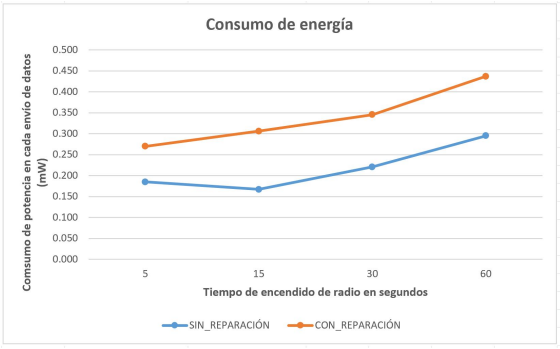


Figura 4.14: *Comportamiento de tasa de recepción y consumo de energía usando un router de borde en conjunto con nodos estáticos.*

Te	Escenario 2	Escenario 3	Escenario 4
5s	65,22 %	62,66 %	65,22 %
15s	74,42 %	75,70 %	73,91 %
30s	76,98 %	78,01 %	75,96 %
60s	68.80 %	79,28 %	88,75 %

Cuadro 4.18: *Tasa de recepción de los distintos escenarios.*

Te	Escenario 2	Escenario 3	Escenario 4
5s	0.301mW	0.307mW	0.270mW
15s	0.318mW	0.374mW	0.306mW
30s	0.369mW	0.384mW	0.346mW
60s	0.404mW	0.460mW	0.437mW

Cuadro 4.19: *Consumo promedio de energía por nodo de los distintos escenarios.*

4.5. Evaluación del mejor escenario

Tras la evaluación de los resultados experimentales extraídos en las secciones 4.2, 4.3 y 4.4, el uso de `rpl_repair_root` nos ofrece mejores tasas de recepción. El Cuadro 4.18 indica la comparación de la tasa de recepción de los 3 escenarios. En el Cuadro 4.19 se puede observar la comparación de consumo energético promedio en cada envío de datos por nodo; a su vez, los costos promedios de algunos dispositivos que pueden usarse se indican en el Cuadro 4.20. La Figura 4.15 ofrece una comparativa de costo de cada escenario.

El escenario 4 con un tiempo para el encendido de radio de 15 segundos ofrece los mejores resultados, presenta un menor costo de despliegue y, al usar solo un router de borde, se simplifica el soporte y administración en comparación a lo otros escenarios.

Dispositivo	Función	Costo(euros)
Kit Raspberry Pi4 [15]	Router de borde	89,90
HAT L-Tek IoT [17]	HAT 6LowPAN para Raspberry Pi	76
SensorTag CC2650STK [6]	Nodo sensor	63,23

Cuadro 4.20: *Costo promedio de cada dispositivo a usarse.*

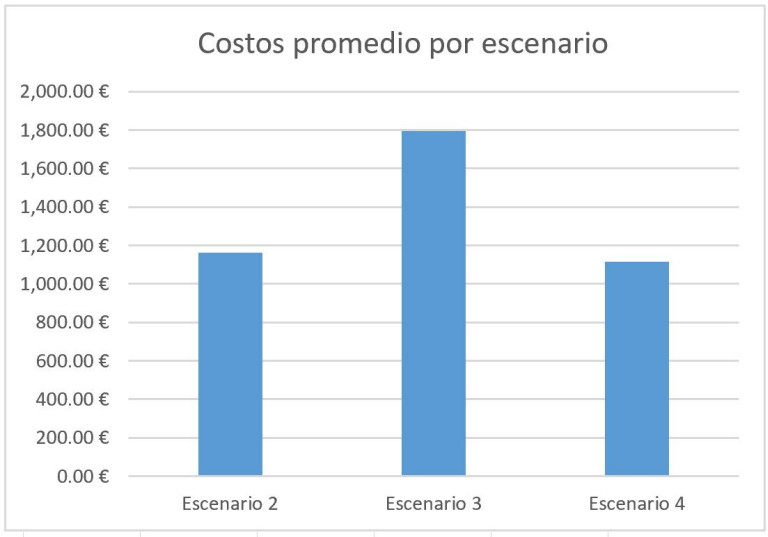


Figura 4.15: *Comparativa de costos entre escenarios.*

Capítulo 5

Conclusiones y trabajo futuro

Consecuencia del trabajo realizado y los resultados obtenidos de los distintos escenarios propuestos, se concluye viable el uso del protocolo de enrutamiento para redes con pérdidas RPL en entornos con nodos sensores móviles, con aplicación directa en la sensorización de diversos parámetros en entornos ganaderos, específicamente aplicado a grupos familiares de vicuñas.

De igual forma, el presente trabajo ha tenido como finalidad servir de guía para estudiar la viabilidad de RPL en entornos móviles, por lo que también se pueden concluir lo siguiente:

1. **Caracterización de movimiento de un grupo familiar de vicuñas:** Esta caracterización puede ser usada como base, no solo en un grupo familiar de vicuñas, sino en cualquier tipo de animal u objeto que presente una dinámica de movimiento para ser usada en el simulador COOJA con el plugin Mobility.

2. **Topologías de red:** Cada tipo de topología usada tiene un impacto en la tasa de recepción de datos enviados por nodos móviles. Las topologías deben ser concordantes con la dinámica de movimiento, para un óptimo uso de los recursos, como el número de routers de borde y nodos estáticos de reenvío a usar.

3. **Impacto de reducir el consumo de energía por un nodo móvil:** Se debe de poner especial atención al minimizar el consumo de energía, ya que presenta un impacto

en RPL, por lo que las tasas de recepción de datos pueden disminuir considerablemente y volver inviable su uso.

4. **Reparación global en RPL:** Su uso permite tener mejores tasas de recepción de datos, pero tiene un impacto en el consumo de energía de un nodo móvil, lo que conlleva a disminuir la vida útil de la fuente de alimentación usada por el nodo sensor.

5. **Automatizar los escenarios de simulación:** Al simular nodos móviles y obtener métricas óptimas para corroborar la viabilidad de RPL se nos presentan dificultades no solo en los escenarios usados, sino en el consumo computacional del host que ejecuta la simulación. Automatizar las simulaciones reduce el consumo computacional y permite realizar múltiples simulaciones una detrás de otra, sin la necesidad de iniciarlas manualmente.

Las conclusiones obtenidas corroboran la aplicabilidad directa del trabajo desarrollado a un entorno real. Esto se debe a que no solo se ha centrado el trabajo en el análisis de datos recogidos, sino también en su almacenamiento y la implementación de un *dashboard* para su visualización.

Por lo tanto, como trabajo futuro se propone instalar instrumentos GPS en ciertos miembros de un grupo familiar de vicuñas para mejorar la caracterización de su dinámica de movimiento. También llevar a cabo un análisis detallado de su hábitat mediante imágenes satelitales, lo que nos permitirá tener un mejor enfoque a la hora de ubicar los routers de borde y nodos de reenvío; esto permitirá no solo tener altas tasas de recepción de datos, sino también de minimizar los costos del despliegue.

Capítulo 6

Introduction

6.1. Smart Farming and IoT.

In recent times, the high precision provided by integrated sensors that measure different environmental factors has allowed the development of precision agriculture, which leads to a significant improvement in productivity, yield, profitability [12] and the reduction of the environmental footprint, with efficient directed irrigation techniques and the optimal use of fertilizers and pesticides in the crop [1], or an appropriate use of food and antibiotics in animals [10, 12].

Precision agriculture is an idea as old as agriculture itself [1], and precedes the concept of intelligent agriculture, which consists of the collection, processing, analysis of data and automation of cultivation processes leading to an improvement in agricultural operation and management. It also allows farmers to make more informed decisions due to the impact that environmental conditions (rain, temperature, humidity, hail, etc.) and unpredictable events (animal diseases or pests) have on agriculture. [12].

In 1999 the term Internet of Things or IoT was introduced by Kevin Ashton to refer to the collection of information from the environment and interaction with the physical world, without human intervention [26], IoT and its interoperable, scalable, widespread and open nature is perfect for agriculture and smart farming. In fact, over time, IoT has gained momentum in the agricultural sector - one example is in Australia, where farmers

are required to have a passive RFID device implanted in the ear of each head of cattle in order to report movements between farms, thus forming a national online database [12].

IoT has been successfully applied in smart cities, healthcare and smart homes, providing seamless connectivity through various open standards (6LoWPAN, ZigBee, CoAP, REST, MQTT) and semantics (RDF, OWL, SWE, etc) among various actors, components, devices, processes and platforms [12]. These provide the opportunity to revolutionize traditional agricultural techniques.

The use of IoT for intelligent agriculture and livestock has several benefits, including reducing the risk of supplier lock-in, the adoption of machinery and detection/automation systems from different companies, as these could easily become interoperable in the overall farm intelligence system. Data exchange is now easier between different heterogeneous components (at the hardware and software level), allowing greater automation and more complex deployments with less effort through the use of Internet standards [12].

6.2. Motivation and approach to the problem

In Peru, most of the research work related to vicuña wool production has a biological approach, that is, it considers the animal's diet and reproduction techniques.

Currently, the vast majority of vicuña wool production is carried out using ancestral breeding techniques, with little or no control.

The animals carry out their activities in *pseudo-captivity*, moving over very large areas and with patterns that are difficult to parameterize, which makes it difficult to collect data unless each animal is self-sufficient to send the data autonomously (for example, with its own satellite connection, or with LPWAN technologies (*Low-Power, Wide Area Networks*), with the consequent energy and acquisition and deployment costs.

The use of intelligent livestock techniques is proposed with a very positive potential impact, since it would allow the monitoring of the family group of vicuñas in real time. Information such as type of terrain, climate conditions, and presence of predators, among

others, could be proactively available.

Among the transmission technologies associated with IoT, 6LowPAN and its associated routing protocol RPL are proposed as two alternatives that can potentially be properly adapted to the characteristics of the selected application (monitoring of biometric or environmental parameters in mobile groups of vicuñas). Specifically, these protocols:

- They are open and widely documented and standardized via RFCs [32, 33].
- Its design is strongly focused on networks with losses and high energy consumption restrictions.
- They are an integral part of several operating systems focused mainly on IoT (e.g. Contiki).
- In a geographically extensive environment, they allow the deployment of a network *mesh*, with the advantage that the intermediate elements can act both as sources of information and as routers, making the sent packets arrive at their final destination.

In environments with moving elements, as is our case, there are certain additional challenges that will be addressed in this work. These challenges can be summarized as follows:

- Movement patterns in the nodes. Normally, RPL networks are treated from a totally static approach. It is therefore convenient to carry out a study on the behaviour of this type of routing algorithms in mobile environments. This study should include parameters such as packet loss, convergence speed, impact of speed and motion pattern on reception quality, etc.
- Impact on the energy consumption of the routing process with mobile nodes. Differentiation of consumption in isolated nodes or in contact with other neighbouring nodes.
- Optimal distribution of nodes (and edge routers that give access to external networks) to minimize the cost of acquisition and maintenance.

6.3. Objectives

The general objective of this work is to develop a study on the viability of the routing protocol for networks with RPL losses in environments with mobile sensor nodes, with direct application in the sensorization of diverse parameters in livestock environments, specifically applied to family groups of vicuñas.

Other objectives of a specific nature derive from the general objective:

- Characterize the movement of a family group of vicuñas to use it in the COOJA simulator with the Mobility plugin.
- Determine the impact of different network topologies (distribution of mobile nodes and edge routers) on the quality of the deployment.
- Reduce the impact on the energy consumed by a mobile node for optimal use in RPL.
- Design and automate simulation scenarios that are faithful to real-life scenarios.

The study carried out in the work is based intensively on simulations through a simulation environment (COOJA). The reason for such an approach is mainly justified by the difficulty, at present, of carrying out real field tests in large geographical environments. The conclusions obtained, however, are directly applicable to a real environment. This implementation approach is, however, intended as future work, and has not been addressed in this TFM.

6.4. Memory structure

The organization of this work is based on 5 chapters described below:

Chapter 2 contains a detailed study of the movement dynamics of vicuñas, which has allowed the development of a methodology and code to generate pseduo random movements with a specific structure for use in the COOJA simulator equipped with the *Mobility* plugin. Chapter 3 is a theoretical study of the concepts needed to deploy the experimental

environments, posed within the framework of a simulation environment. Chapter 4 presents the results obtained from different realistic experimental scenarios, which roughly model the environments that could be found in a real deployment. Chapter 5 lists the main conclusions of the work and, at the same time, poses a set of extensions as future work.

In accordance with the regulations, the Chapters 6 and 7 correspond to the English translation of the introductory and concluding chapters.

Capítulo 7

Conclusions

As a result of the work carried out and the results obtained from the different scenarios proposed, the use of the routing protocol for networks with RPL losses in environments with mobile sensor nodes is concluded to be viable, with direct application in the sensorization of diverse parameters in livestock environments, specifically applied to family groups of vicuñas.

Likewise, the present work has been intended as a guide to study the viability of RPL in mobile environments, so the following conclusions can also be drawn:

1. **Movement characterization of a family group of vicuñas:** This characterization can be used as a basis, not only in a family group of vicuñas, but in any type of animal or object that presents a movement dynamic to be used in the COOJA simulator with the Mobility plugin.

2. **Network topologies:** Each type of topology used has an impact on the data reception rate, sent by mobile nodes, the topologies must be consistent with the movement dynamics, for an optimal use of resources, such as the number of edge routers and static forwarding nodes to use.

3. **Of reducing the power consumption by a mobile node:** Special attention should be paid to minimizing power consumption, as it has an impact on RPL, so that data reception rates can be considerably reduced and make its use unfeasible.

4. **Global repair in RPL:** Its use allows better data reception rates, but has an impact on the power consumption of a mobile node, which leads to reduce the life of the power supply used by the sensor node.

5. **Automate simulation scenarios:** By simulating mobile nodes and obtaining optimal metrics to corroborate the viability of RPL. We are presented with difficulties not only in the scenarios used, but also in the computational consumption of the host running the simulation. Automating the simulations reduces the computational consumption and allows to perform multiple simulations one after the other, without the need to start them manually.

The conclusions obtained corroborate the direct applicability of the work developed to a real environment. This is because the work has not only focused on the analysis of the data collected, but also on its storage and the implementation of a *dashboard* for its visualization.

Therefore, as future work it is proposed to install GPS instruments on certain members of a family group of vicuñas to improve the characterization of their movement dynamics. Also to carry out a detailed analysis of their habitat by means of satellite images, which will allow us to have a better approach when locating the edge routers and forwarding nodes; this will allow not only to have high rates of data reception, but also to minimize the costs of the deployment.

Bibliografía

[1] R. Bongiovanni and J. Lowenberg-Deboer. Precision agriculture and sustainability. *Precision Agriculture*, 5(4):359–387, 2004.

[2] Alejandro Cama and Dora Cama. Las redes de sensores inalámbricos y el internet de las cosas. *Inge-Cuc*, 8(1):162–172, 2012.

[3] Bernabé Alex Mamani Choque. Estudio de la distribución dinámica del movimiento espacial de las vicuñas (vicugna vicugna) y su interrelación con el medio físico y social, 2006.

[4] ContikiOS. Contikios. URL: https://github.com/contiki-os/contiki (Accedido el 10-02-2020).

[5] David E Culler. Wireless embedded internetworking. *University of California, Berkeley*, 2008.

[6] Digikey. Cc2650stk. URL: https://www.digikey.es/product-detail/es/texas-instruments/CC2650STK/296-38831-ND/5130740 (Accedido el 23-06-2020).

[7] Adam Dunkels. The ContikiMAC Radio Duty Cycling Protocol. *SICS Technical Report T2011:13 , ISSN 1100-3154*, pages 1–11, 2011.

[8] Adam Dunkels, Björn Grönvall, and Thiemo Voigt. Contiki - A lightweight and flexible operating system for tiny networked sensors. *Proceedings - Conference on Local Computer Networks, LCN*, pages 455–462, 2004.

[9] Marc Fàbregas Bachs. Diseño e implementación de una estación meteorológica de bajo coste con conectividad a Internet (Tesis de pregrado). 2016.

[10] Emiliano García and Fernando Flego. 2009 Agricultura de precision. *Tecnologia Agropecuaria*, pages 24–31, 2007.

[11] Wei Ge, Lin Zheng, Peng Luo, and Zhenghong Liu. Implementation of multiple border routers for 6LoWPAN with ContikiOS. *IET Conference Publications*, 2015(CP664), 2015.

[12] Andreas Kamilaris, Feng Gao, Francesc X. Prenafeta-Boldu, and Muhammad Intizar Ali. Agri-IoT: A semantic framework for Internet of Things-enabled smart farming applications. *2016 IEEE 3rd World Forum on Internet of Things, WF-IoT 2016*, pages 442–447, 2017.

[13] Inès El Korbi, Mohamed Ben Brahim, Cedric Adjih, and Leila Azouz Saidane. Mobility enhanced RPL for wireless sensor networks. *2012 3rd International Conference on the Network of the Future, NOF 2012*, pages 63–70, 2012.

[14] Konrad-felix Krentz, Christoph Meinel, and Hendrik Graupner. Countering Three Denial-of-Sleep Attacks on ContikiMAC. *Ewsn*, pages 108–119, 2017.

[15] Kubii. Kit raspberry pi4. URL: https://www.kubii.es/raspberry-pi-4-modelo-b/3011-starter-kit-raspberry-pi4-8gb-3272496302112.html (Accedido el 23-06-2020).

[16] Mohan KumarJ, PR Venkateswaran, and N GopalakrishnaKini. Software-based energy estimation for localization algorithms in wireless sensor networks using Contiki-COOJA 1. 119(12):12655–12663, 2018.

[17] L-tek. Hat l-tek iot. URL: https://www.l-tek.com/web-shop/arm-mbed-6lowpan-border-router-hat-ltek-iot/ (Accedido el 23-06-2020).

[18] Hanane Lamaazi, Nabil Benamar, and Antonio J. Jara. RPL-based networks in static and mobile environment: A performance assessment analysis. *Journal of King Saud University - Computer and Information Sciences*, 30(3):320–333, 2018.

[19] Marcus Lunden. Rwmmsim. URL: https://github.com/msloth/RWMMSim (Accedido el 10-02-2020).

[20] G. Montenegro, N. Kushalnagar, and J. Hui. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, Network Working Group, September 2007.

[21] Mona Nasseri, Hussein Al-Olimat, Mansoor Alam, Junghwan Kim, Robert Green, and Wei Cheng. Contiki cooja simulation for time bounded localization in wireless sensor network. *Simulation Series*, 47(3):1–7, 2015.

[22] Fredrik Osterlind. Mobility of nodes in cooja. URL: https://sourceforge.net/p/contikiprojects/code/HEAD/tree/sics.se/mobility/ (Accedido el 10-02-2020).

[23] Fredrik Österlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. Cross-level sensor network simulation with COOJA. *Proceedings - Conference on Local Computer Networks, LCN*, pages 641–648, 2006.

[24] Lafuente Pereyra. Convención internacional sobre camélidos sudamericanos. *Convención Internacional sobre Camélidos Sudamericanos*, 1988.

[25] Julio Tomas Martínez Pérez. Diseño, simulación y despliegue de redes inalámbricas de sensores con contiki. 2014.

[26] Manuel R Pérez, ; Mendoza, Miguel A, & Suarez, and Marco J. Paradigma IoT: desde su conceptualización hacia su aplicación en la agricultura IoT . Technical Report 18.

[27] Dorel Picovici, A. CONTIU, A. TOPA, and John Nelson. Virtual lab for wireless sensor networks. *Advances in Electrical and Computer Engineering*, 8, 06 2008.

[28] E.C. Quispe, T.C. Rodríguez, L.R. Iñiguez, and J.P. Mueller. Producción de fibra de alpaca, llama, vicuña y guanaco en sudamérica. *Animal Genetic Resources Information*, 45:1–14, 2009.

[29] Tobias Reusing. Comparison of operating systems tinyos and contiki. *Proceedings of the Seminar Sensor Nodes – Operation, Network and Application (SN)*, pages 7–13, 2012.

[30] Peter Ruckebusch, Jens Devloo, David Carels, Eli De Poorter, and Ingrid Moerman. An evaluation of link estimation algorithms for rpl in dynamic wireless sensor networks. In *Internet of Things. IoT Infrastructures*, pages 349–361, Cham, 2016. Springer International Publishing.

[31] O Skewes, F González, M Maldonado, C Ovalle, and L Rubilar. *Desarrollo y evaluación de técnicas de cosecha y captura de guanacos para su aprovechamiento comercial y sustentable en Tierra del Fuego*. 2000.

[32] P. Thubert, C. Bormann, L. Toutain, and R. Cragie. IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing Header . RFC 8138, Internet Engineering Task Force, April 2017.

[33] P. Thubert, T. Winter, A. Brandt, and J. Hui. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, Internet Engineering Task Force, March 2012.

[34] Orlando Olaya Trinidad. Repositorio de códigos y configuraciones. URL: https://github.com/oolaya1815/TFM_ARCHIVOS (Accedido el 23-06-2020).

Apéndice A

Configuración del entorno de simulación

A.1. Conexión de COOJA con una red externa

En una simulación en COOJA, la información que se transmite entre los distintos nodos no podrá interactuar con servicios de red externos, como servidores UDP, bases de datos, etc.

Para que un nodo presente una conectividad a una red externa, se debe usar un router de borde y el programa tunslip6 presente en la carpeta tools del directorio de ContikiOS. Tunslip6 cumple la tarea de puente entre el router de borde y la tarjeta física del computador donde se ejecuta la simulación.

Para el uso de tunslip6 primero se debe compilar el código tunslip6.c, que genera el ejecutable mostrado en la Figura A.1

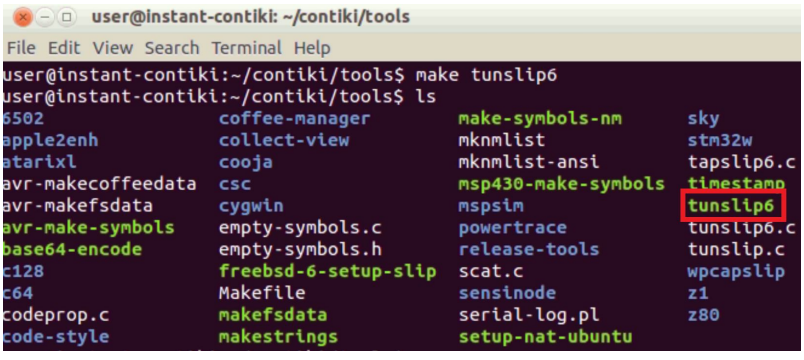


Figura A.1: *Creación del ejecutable tunslip6.*

Posteriormente, abrir un puerto serial tipo servidor en el router de borde, la Figura A.2

muestra esta acción.

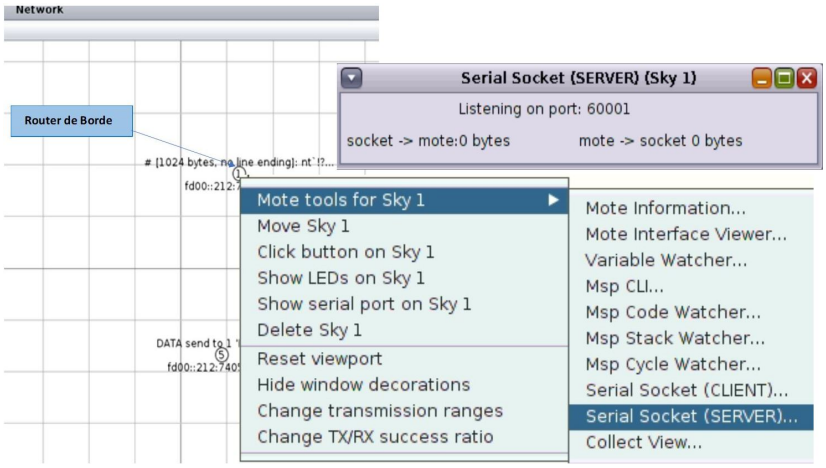


Figura A.2: *Habilitando Serial Socket Server.*

Por último, es necesario crear una conexión puente entre el router de borde y la tarjeta física del computador mediante tunslip6. El router de borde escuchará por el puerto 6001 y se creará una interfaz tun0; adicionalmente, se debe especificar el prefijo de red que compartirá el router de borde con los nodos clientes, esto se realiza al definir una dirección IPv6 que adoptará el computador. La Figura A.3 muestra la salida que se obtendrá por consola al ejecutar el siguiente comando:

```
sudo ./tunslip6 -v2 -t tun0 -a localhost -p 60001 fd00::1/64
```

```
ifconfig tun0 inet `hostname` up
ifconfig tun0 add fd00::1/64
ifconfig tun0 add fe80::0:0:0:1/64
ifconfig tun0

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:127.0.1.1  P-t-P:127.0.1.1  Mask:255.255.255.255
          inet6 addr: fe80::1/64 Scope:Link
          inet6 addr: fd00::1/64 Scope:Global
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

*** Address:fd00::1 => fd00:0000:0000:0000
Got configuration message of type P
Setting prefix fd00::
Server IPv6 addresses:
```

Figura A.3: *Uso de tunslpip.*

A.2. Uso de Mobility en COOJA

Mobility es un plugin desarrollado por Fredrik Osterlind, el siguiente link[22] contiene el código fuente y una guía detallada para su instalación y su uso en COOJA.

Para usar Mobility se requiere de un archivo con extensión .dat, se explica con más detalle la estructura de este archivo en la Sección 3.1.3. La carga del archivo a la simulación se aprecia en la Figura A.4.

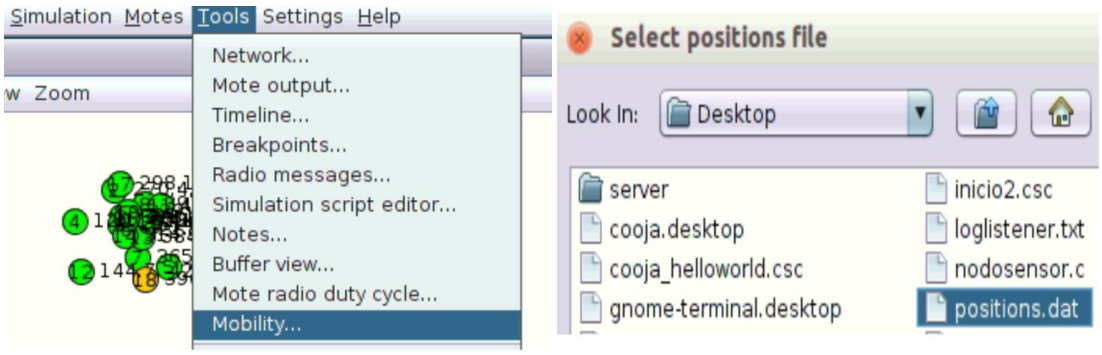


Figura A.4: *Uso de Mobility.*

A.3. Automatización de las simulaciones

Entre las funcionalidades que ofrece COOJA se tiene el inicio de simulaciones sin necesidad de una interfaz gráfica, además de efectuar múltiples repeticiones sin tener que iniciarlas manualmente.

El uso de COOJA sin interfaz gráfica permite aumentar considerablemente la rapidez de ejecución de las simulaciones, ideal para simulaciones complejas con los mismos recursos del host.

Para ejecutar simulaciones sin GUI se requiere disponer de una simulación preparada, y activar el "script editor" que se muestra en la Figura A.5.

En la Figura A.5 se aprecia un ejemplo de un script. La variable "TIMEOUT" establece la duración de la simulación en milisegundos, la orden log.log almacena en un archivo con

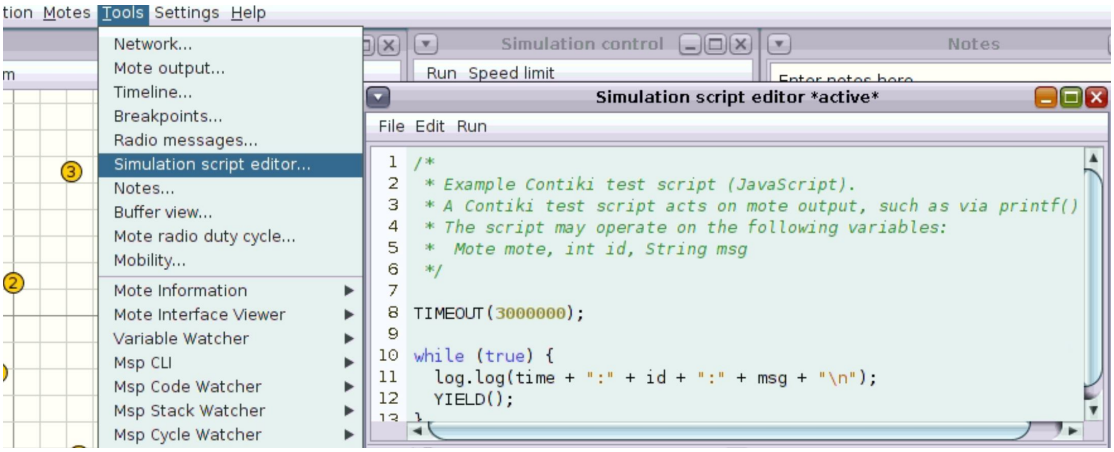


Figura A.5: *script editor.*

extensión .log la información que cada nodo ha mostrado en su consola con su determinada marca de tiempo e ID de nodo.

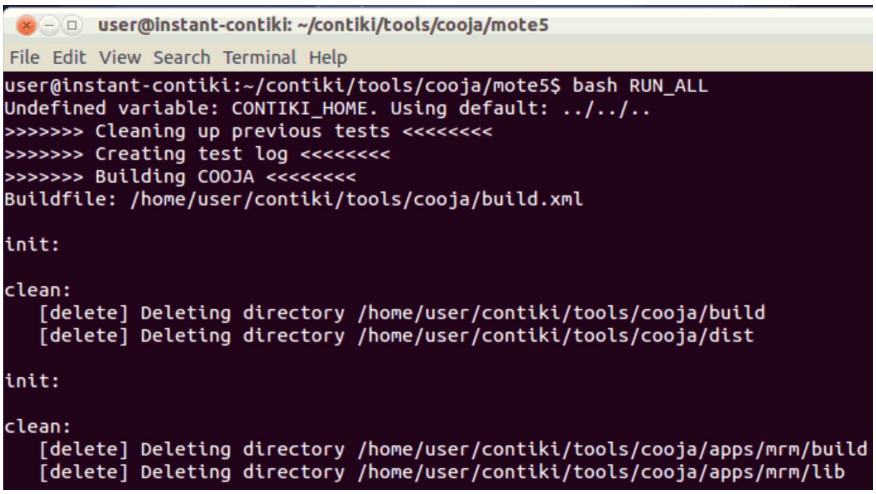
Como se necesita también la herramienta tunslip6 para conectar los router de borde a la red externa, se usa el siguiente script de la Figura A.6

inicio.sh

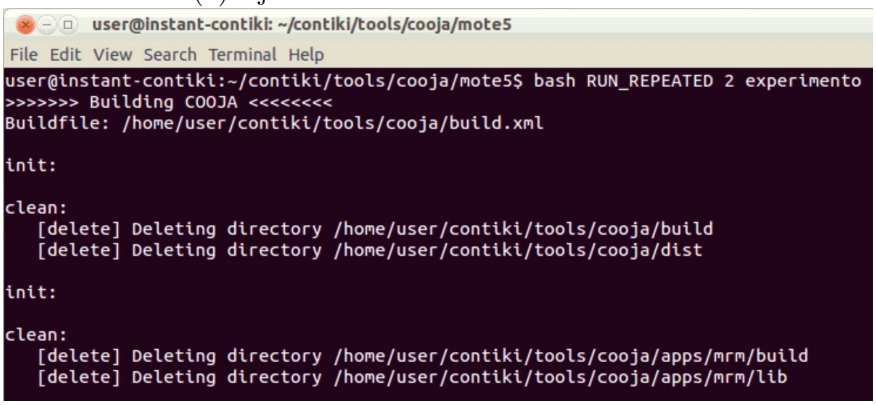
```
#!/bin/bash
cd /home/user/contiki/tools
echo "Iniciando tuneles"
./tunslip6 -v2 -t tun0 -a localhost -p 60018 fd00::1/64 &
./tunslip6 -v2 -t tun1 -a localhost -p 60019 fd01::1/64 &
./tunslip6 -v2 -t tun2 -a localhost -p 60020 fd02::1/64 &
./tunslip6 -v2 -t tun3 -a localhost -p 60021 fd03::1/64 &
echo "finalizado"
```

Figura A.6: *script tunslip6.*

Por último, mediante los script RUN_ALL ó RUN_REPEATED se puede ejecutar una única simulación o múltiples simulaciones respectivamente. La ejecución de cada uno de los scripts mencionados se muestra en la Figura A.7.



(a) Ejecución de una única simulación.



(b) Ejecución de 2 simulaciones.

Figura A.7: *Simulación en COOJA sin GUI.*

Apéndice B

Códigos y archivos usados en el TFM

Los códigos y configuración de cada uno de los escenarios se ubican en el siguiente repositorio: GithubTFM [34].

Además, la estructura del repositorio corresponde a lo siguiente:

- **Automatizar:** Scripts necesarios para automatizar las simulaciones con COOJA.
- **Servidor:** Código de cada uno de los servidores, al igual que una guía para su uso, instalación y configuración, sea el caso.
- **codigos:** Carpeta que contiene los archivos fuente para cada nodo usado, script de generación de movimiento para su uso en COOJA con el plugin Mobility.
- **contiki:** Contiene todos los archivos fuente de ContikiOS, se incluyen las carpetas personalizadas usadas en el presente TFM.
- **escenarios:** Archivos de simulación de cada escenario.
- **resultados_excel:** Hojas de cálculo usadas en cada escenario.