
This is the **published version** of the bachelor thesis:

Torres Delgado, Marc. *Diseño e implementación de un sistema IoT resiliente frente a ataques mediante mecanismos de alta disponibilidad*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Grau en Enginyeria Electrònica de Telecomunicació)

This version is available at <https://ddd.uab.cat/record/336122>

under the terms of the  license.

Treball Fi de Grau

Grau d'Enginyeria Electrònica de Telecomunicació

**Diseño e implementación de un sistema IoT
resiliente frente a ataques mediante mecanismos de
alta disponibilidad**

Marc Torres Delgado

Director: Albert Crespo Yepes

DEPARTAMENT DE TELECOMUNICACIÓ I D'ENGINYERIA ELECTRÒNICA
Escola d'Enginyeria
Universitat Autònoma de Barcelona (UAB)
Juliol 2026

AGRADECIMIENTOS

En primer lugar, quiero agradecer a mi familia el apoyo, la ayuda y la confianza que me han brindado durante todos estos años. En especial, quiero dar las gracias a mis padres y a mis abuelos, por haber estado siempre presentes, por animarme en los momentos más difíciles, por acompañarme durante todo este camino académico y por alegrarse y celebrar mis logros como si fueran propios. Su apoyo ha sido una parte esencial para poder llegar hasta aquí.

También quiero agradecer a mi pareja y a mis amigos su apoyo constante durante estos años. Su compañía ha sido fundamental para mantener la motivación y afrontar con más fuerza las dificultades que han ido apareciendo a lo largo del proceso. Sin duda, una de las mejores partes de mi etapa universitaria ha sido la gente que he conocido durante este camino. Su compañía ha hecho que esta etapa haya sido mucho más llevadera, enriquecedora y especial.

En el ámbito académico, quiero expresar mi agradecimiento a todos los buenos profesores que he tenido durante el grado, por los conocimientos transmitidos y por haber contribuido a mi formación como estudiante. De manera especial, quiero agradecer a mi tutor, Albert Crespo, por su orientación, disponibilidad y seguimiento durante la realización de este Trabajo de Fin de Grado.

Asimismo, quiero agradecer a Remo Suppi su ayuda en aspectos técnicos relacionados con el desarrollo del proyecto. Tanto en su caso como en el de mi tutor, Albert Crespo, quiero destacar especialmente su vocación docente, su implicación y sus ganas de motivar y ayudar a los estudiantes. Ha sido un verdadero placer poder trabajar con ambos durante el desarrollo de este Trabajo de Fin de Grado, y sus aportaciones han sido muy importantes para poder llevarlo a cabo.

Finalmente, quiero dar las gracias a todas las personas que, de una forma u otra, me han acompañado durante esta etapa. Este trabajo representa el cierre de un camino lleno de esfuerzo, aprendizaje y crecimiento, y no habría sido posible sin el apoyo recibido a lo largo de estos años.

RESUMEN

Este Trabajo de Fin de Grado se centra en el diseño, implementación y evaluación de una arquitectura IoT orientada a mejorar la disponibilidad del servicio frente a fallos y escenarios de denegación de servicio. El sistema se basa en un nodo IoT de monitorización ambiental que envía datos periódicamente a una plataforma cloud, utilizando una infraestructura de red virtualizada como medio de comunicación.

El problema que se pretende estudiar surge de la dependencia que tienen los sistemas IoT respecto a su conectividad. En este tipo de arquitecturas, no basta con que el dispositivo de adquisición funcione correctamente, ya que los datos solo resultan útiles si llegan de forma continua a la plataforma encargada de almacenarlos y visualizarlos. Por tanto, la caída del gateway, la pérdida de conectividad o una situación de degradación de la red pueden interrumpir el envío de información y afectar directamente a la disponibilidad del servicio.

La solución propuesta incorpora dos routers virtuales configurados en modo redundante: un Router Principal y un Router Respaldo. Mediante mecanismos de alta disponibilidad, el sistema permite que el router alternativo asuma la función de gateway cuando el router principal deja de estar disponible. De este modo, se busca mantener la comunicación entre el sistema IoT y la plataforma de monitorización sin requerir intervención manual sobre el cliente.

La evaluación se ha realizado en un entorno de laboratorio, donde se reprodujeron escenarios controlados de fallo, degradación de conectividad y situaciones compatibles con ataques de denegación de servicio. El comportamiento del sistema se analizó mediante métricas como la pérdida de paquetes, la latencia, el tiempo de conmutación y la continuidad del envío de datos.

RESUM

Aquest Treball de Fi de Grau se centra en el disseny, la implementació i l'avaluació d'una arquitectura IoT orientada a millorar la disponibilitat del servei davant de fallades i escenaris de denegació de servei. El sistema es basa en un node IoT de monitoratge ambiental que envia dades periòdicament a una plataforma cloud, utilitzant una infraestructura de xarxa virtualitzada com a mitjà de comunicació.

El problema que es pretén estudiar sorgeix de la dependència que tenen els sistemes IoT respecte de la seva connectivitat. En aquest tipus d'arquitectures, no n'hi ha prou que el dispositiu d'adquisició funcioni correctament, ja que les dades només són útils si arriben de manera contínua a la plataforma encarregada d'emmagatzemar-les i visualitzar-les. Per tant, la caiguda del gateway, la pèrdua de connectivitat o una situació de degradació de la xarxa poden interrompre l'enviament d'informació i afectar directament la disponibilitat del servei.

La solució proposada incorpora dos routers virtuals configurats en mode redundant: un Router Principal i un Router Reserva. Mitjançant mecanismes d'alta disponibilitat, el sistema permet que el router alternatiu assumeixi la funció de gateway quan el router principal deixa d'estar disponible. D'aquesta manera, es busca mantenir la comunicació entre el sistema IoT i la plataforma de monitoratge sense requerir intervenció manual sobre el client.

L'avaluació s'ha dut a terme en un entorn de laboratori, on s'han reproduït escenaris controlats de fallada, degradació de connectivitat i situacions compatibles amb atacs de denegació de servei. El comportament

del sistema s'ha analitzat mitjançant mètriques com la pèrdua de paquets, la latència, el temps de commutació i la continuïtat de l'enviament de dades.

ABSTRACT

This Bachelor's Thesis focuses on the design, implementation, and evaluation of an IoT architecture aimed at improving service availability against failures and denial-of-service scenarios. The system is based on an environmental monitoring IoT node that periodically sends data to a cloud platform, using a virtualised network infrastructure as the communication medium.

The problem addressed in this work arises from the dependence of IoT systems on connectivity. In this type of architecture, it is not enough for the data acquisition device to operate correctly, since the data are only useful if they are continuously delivered to the platform responsible for storing and visualising them. Therefore, a gateway failure, loss of connectivity, or network degradation can interrupt data transmission and directly affect service availability.

The proposed solution incorporates two virtual routers configured in a redundant mode: a Primary Router and a Backup Router. By using high-availability mechanisms, the system allows the alternative router to assume the gateway function when the primary router becomes unavailable. In this way, the communication between the IoT system and the monitoring platform can be maintained without requiring manual intervention on the client side.

The evaluation was carried out in a laboratory environment, where controlled failure scenarios, connectivity degradation, and situations compatible with denial-of-service attacks were reproduced. The system behaviour was analysed using metrics such as packet loss, latency, failover time, and continuity of data transmission.

Índice

CAPÍTULO 1	8
INTRODUCCIÓN Y FUNDAMENTOS TEÓRICOS	8
1.1 Contexto y motivación del proyecto	8
1.1.1 Evolución del IoT y su aplicación en monitorización ambiental	8
1.1.2 Importancia de la conectividad en sistemas IoT	8
1.2 Internet de las Cosas y monitorización ambiental	9
1.2.1 Arquitectura básica de un sistema IoT	9
1.2.2 Plataformas cloud para la recepción de datos	10
1.3 Seguridad en sistemas IoT	10
1.3.1 Confidencialidad, integridad y disponibilidad	10
1.3.2 La disponibilidad como requisito crítico	11
1.4 Problemática de la disponibilidad en sistemas IoT	11
1.5 Alta disponibilidad como mecanismo de resiliencia	11
1.6 Fundamentos de red necesarios para el proyecto	12
1.6.1 Comunicación TCP/IP en sistemas IoT	12
1.6.2 Direccionamiento IP y gateway	13
1.6.3 Enrutamiento, NAT y masquerading	13
1.6.4 VRRP, IP virtual y conmutación por fallo	13
1.7 Objetivos del trabajo	14
1.7.1 Objetivo principal	14
1.7.2 Objetivos específicos	14
 CAPÍTULO 2	 15
DISEÑO DE LA ARQUITECTURA PROPUESTA	15
2.1 Planteamiento inicial del sistema IoT físico	15
2.2 Limitaciones encontradas en la integración física con routers virtuales	15
2.3 Adaptación metodológica: sustitución del nodo físico por un Cliente-IoT virtual	16
2.4 Arquitectura final del sistema	17
2.5 Elementos del sistema	18
2.5.1 Cliente-IoT virtual	18
2.5.2 Router-Principal	18
2.5.3 Router-Respaldo	18
2.5.4 Router doméstico	18
2.5.5 Plataforma ThingSpeak	18
2.6 Direccionamiento IP del sistema	18
2.7 Flujo de comunicación extremo a extremo	19

CAPÍTULO 3	21
IMPLEMENTACIÓN DE LA INFRAESTRUCTURA VIRTUALIZADA	21
3.1 Entorno de virtualización utilizado	21
3.2 Configuración de interfaces de red en VirtualBox	22
3.3 Configuración del Router-Principal	22
3.3.1 Direcccionamiento IP	22
3.3.2 Activación del reenvío IPv4	23
3.3.3 Configuración de nftables y NAT	23
3.3.4 Persistencia de la configuración	24
3.4 Configuración del Router-Respaldo	24
3.4.1 Direcccionamiento IP	24
3.4.2 Activación del reenvío IPv4	25
3.4.3 Configuración de nftables y NAT	25
3.4.4 Persistencia de la configuración	26
3.5 Configuración del Cliente-IoT	26
3.5.1 Direcccionamiento IP	26
3.5.2 Configuración inicial del gateway	26
3.5.3 Pruebas de conectividad	27
3.6 Integración con ThingSpeak	28
3.7 Validación inicial del envío de datos	28
CAPÍTULO 4	30
IMPLEMENTACIÓN DE ALTA DISPONIBILIDAD	30
4.1 Diseño de la solución de alta disponibilidad	30
4.2 Funcionamiento de Keepalived y VRRP en la arquitectura	31
4.3 Configuración de Keepalived en Router-Principal	31
4.4 Configuración de Keepalived en Router-Respaldo	32
4.5 Definición de la IP virtual compartida	33
4.6 Configuración del gateway del Cliente-IoT	34
4.7 Funcionamiento normal del sistema	34
4.8 Funcionamiento ante fallo del Router-Principal	35
CAPÍTULO 5	37
PRUEBAS, ATAQUES Y ANÁLISIS DE RESULTADOS	37
5.1 Metodología de pruebas	37
5.2 Prueba de conectividad y envío de datos a ThingSpeak	37
5.3 Escenario base: funcionamiento sin fallo	38
5.4 Escenario de caída manual del Router-Principal	39
5.5 Escenario de fallo de interfaz del Router-Principal	39
5.6 Escenario de denegación de servicio en entorno controlado	41
5.7 Escenario de denegación de servicio externo	42
5.7.1 Ataque con hping3 en modo flood	43
5.7.2 Ataque Man-in-the-Middle y filtrado VRRP Unicast	44
5.8 Recuperación del Router-Principal	47

5.9 Pérdida de paquetes, tiempo de conmutación y continuidad del servicio.....	47
CAPÍTULO 6	49
CONCLUSIONES Y LÍNEAS FUTURAS	49
6.1 Conclusiones generales	49
6.2 Cumplimiento de los objetivos	50
6.3 Líneas de trabajo futuro	51
REFERENCIAS Y BIBLIOGRAFÍA	53

CAPÍTULO 1

Introducción y fundamentos teóricos

1.1 Contexto y motivación del proyecto

1.1.1 Evolución del IoT y su aplicación en monitorización ambiental

El Internet de las Cosas, conocido como IoT (*Internet of Things*), representa la evolución de los sistemas electrónicos y de telecomunicaciones hacia un modelo en el que objetos físicos pueden conectarse a una red, intercambiar información y ofrecer servicios avanzados. La Unión Internacional de Telecomunicaciones define el IoT como una infraestructura global que permite interconectar objetos físicos y virtuales mediante tecnologías de la información y la comunicación interoperables [1]. Esta definición resulta especialmente relevante porque no limita el IoT a un tipo concreto de dispositivo, sino que lo entiende como un ecosistema formado por sensores, redes, plataformas de procesamiento y servicios digitales.

Aunque el concepto de conectar objetos a una red no es reciente, su expansión se ha visto impulsada por varios factores: la reducción del coste de los microcontroladores, la disponibilidad de comunicaciones inalámbricas, el desarrollo de plataformas cloud y la mejora de las herramientas de análisis de datos. Gracias a ello, dispositivos de bajo coste pueden medir variables del entorno, enviar información a través de Internet y permitir su visualización remota. Informes recientes estiman que el número de dispositivos IoT conectados continúa creciendo de forma significativa, alcanzando decenas de miles de millones de dispositivos a nivel mundial [2].

Una de las aplicaciones más representativas del IoT es la monitorización ambiental. En este tipo de sistemas, sensores distribuidos recogen variables como temperatura, humedad, presión atmosférica, presencia, calidad del aire o localización geográfica. Estos datos pueden utilizarse para observar el estado de un entorno, detectar anomalías o facilitar la toma de decisiones. Su uso es habitual en ámbitos como la domótica, la agricultura inteligente, la industria, las ciudades inteligentes o los sistemas de supervisión remota.

En el contexto de este trabajo, la monitorización ambiental se toma como caso de uso para estudiar la disponibilidad de un sistema IoT. El objetivo no se centra únicamente en generar o capturar datos, sino en garantizar que estos puedan llegar de forma continua a una plataforma remota. Para ello, se emplea ThingSpeak, una plataforma de análisis IoT que permite agregar, visualizar y analizar flujos de datos en la nube [4]. De esta forma, el sistema desarrollado permite simular el comportamiento de un dispositivo IoT que envía información periódica hacia Internet, facilitando el estudio de la comunicación extremo a extremo y de los posibles fallos en la infraestructura de red.

1.1.2 Importancia de la conectividad en sistemas IoT

En un sistema IoT, la conectividad es un elemento esencial. El valor del sistema no depende únicamente de que el sensor mida correctamente una variable, sino de que esa información pueda transmitirse hasta el punto donde será almacenada, visualizada o procesada. Por este motivo, un sistema IoT debe

entenderse como una cadena completa formada por el dispositivo de adquisición, la red local, el gateway o router de salida, la conexión a Internet y la plataforma remota.

Cuando cualquiera de estos elementos falla, el sistema puede perder su utilidad. Por ejemplo, un sensor puede seguir funcionando correctamente a nivel físico, pero si no consigue enviar los datos a la nube, el usuario no podrá conocer el estado real del entorno monitorizado. En aplicaciones de supervisión continua, esta ausencia de datos puede ser crítica, ya que puede ocultar situaciones anómalas o retrasar la respuesta ante un incidente.

Desde el punto de vista de la ciberseguridad, esta problemática se relaciona directamente con el concepto de disponibilidad. La seguridad no solo consiste en proteger la información frente a accesos no autorizados o modificaciones indebidas, sino también en garantizar que el sistema siga prestando servicio cuando se necesita. Organismos como ENISA señalan que la protección de sistemas IoT debe contemplar la continuidad del servicio, especialmente cuando estos dispositivos forman parte de entornos industriales, infraestructuras críticas o servicios dependientes de datos en tiempo real [3].

1.2 Internet de las Cosas y monitorización ambiental

1.2.1 Arquitectura básica de un sistema IoT

Un sistema IoT puede entenderse como una arquitectura formada por varios elementos que trabajan de manera coordinada para transformar una magnitud física en información digital accesible de forma remota. De forma general, este tipo de sistemas incluye una capa de adquisición, una capa de comunicación, una capa de procesamiento y una capa de visualización o almacenamiento. Esta división permite comprender que el IoT no se limita únicamente al sensor, sino que depende de toda una infraestructura que permite captar, transportar y explotar los datos.

La primera capa está formada por los dispositivos de adquisición. En ella se encuentran los sensores, actuadores y microcontroladores encargados de interactuar con el entorno físico. En un sistema de monitorización ambiental, esta capa puede medir variables como temperatura, humedad, presión, presencia o calidad del aire. El microcontrolador actúa como elemento intermedio entre el sensor y la red, ya que convierte la lectura física en un dato digital que puede ser procesado y enviado.

La segunda capa corresponde a la comunicación. Su función es transportar los datos desde el dispositivo IoT hasta un destino remoto. Para ello pueden utilizarse diferentes tecnologías dependiendo del alcance, consumo energético, coste y ancho de banda requerido.

La tercera capa está relacionada con el procesamiento y almacenamiento de la información. En muchos sistemas IoT actuales, los datos no se almacenan localmente, sino que se envían a plataformas cloud. Esto permite consultar la información desde cualquier ubicación, representar gráficamente las medidas y aplicar herramientas de análisis.

Finalmente, la capa de visualización permite al usuario interpretar los datos recibidos. Esta capa puede adoptar la forma de gráficas, paneles de control, alarmas o registros históricos. En el caso de la monitorización ambiental, resulta especialmente útil para observar tendencias, detectar anomalías y comprobar si el sistema mantiene una transmisión continua.



Figura 1. Esquema genérico de un sistema IoT

1.2.2 Plataformas cloud para la recepción de datos

Las plataformas cloud desempeñan un papel importante en los sistemas IoT modernos, ya que permiten recibir, almacenar, visualizar y analizar datos generados por dispositivos conectados. En lugar de mantener toda la información en un equipo local, el dispositivo IoT envía sus medidas a un servicio remoto accesible a través de Internet. Esto facilita la consulta de los datos desde distintos dispositivos y permite centralizar la información generada por uno o varios nodos.

En este proyecto se utiliza ThingSpeak como plataforma de recepción de datos. ThingSpeak es un servicio orientado al análisis IoT que permite agregar, visualizar y analizar flujos de datos en la nube [4]. Su funcionamiento se basa en canales, donde cada canal puede contener diferentes campos asociados a las variables medidas. Por ejemplo, en un sistema de monitorización ambiental, un campo puede almacenar la temperatura y otro la humedad.

1.3 Seguridad en sistemas IoT

1.3.1 Confidencialidad, integridad y disponibilidad

La seguridad en sistemas IoT debe entenderse como un conjunto de medidas orientadas a proteger tanto la información como la infraestructura que permite capturarla, transmitirla y procesarla. En este tipo de sistemas intervienen sensores, microcontroladores, redes de comunicación, routers, servicios cloud y usuarios finales, por lo que un fallo de seguridad puede producirse en distintos puntos de la arquitectura.

Tradicionalmente, la seguridad de la información se explica mediante tres propiedades fundamentales: confidencialidad, integridad y disponibilidad.

La confidencialidad se refiere a evitar que la información sea accesible por personas o sistemas no autorizados. La integridad consiste en garantizar que la información no sea modificada de forma no autorizada. La disponibilidad, por su parte, hace referencia a que el sistema y sus datos estén accesibles cuando se necesitan.

El National Institute of Standards and Technology define la seguridad de la información como la protección de la información y de los sistemas frente a accesos, usos, divulgaciones, interrupciones, modificaciones o destrucciones no autorizadas, con el objetivo de garantizar la confidencialidad, la integridad y la disponibilidad [6]. Esta definición resulta adecuada para este proyecto, ya que permite justificar que la continuidad del servicio también forma parte de la seguridad del sistema, y no únicamente la protección frente al robo o alteración de datos.

1.3.2 La disponibilidad como requisito crítico

En los sistemas IoT de monitorización, la disponibilidad adquiere una importancia especial. Estos sistemas suelen emplearse para observar el estado de un entorno a lo largo del tiempo, por lo que la ausencia de datos puede afectar directamente a la utilidad del sistema. Si un sensor mide correctamente, pero no consigue transmitir la información hasta la plataforma cloud, el resultado práctico para el usuario es equivalente a no disponer de la medida.

Esta situación es especialmente relevante en aplicaciones donde la información debe recibirse de forma continua o con una periodicidad definida. En un sistema ambiental, una interrupción puede impedir detectar cambios bruscos de temperatura o humedad. En un entorno industrial, la pérdida de comunicación puede dificultar la supervisión de un proceso. En un sistema de seguridad, la falta de datos puede ocultar un evento relevante. Por tanto, la disponibilidad no debe tratarse como una característica secundaria, sino como un requisito funcional del sistema.

En el ámbito IoT, la disponibilidad depende de varios elementos: el dispositivo de adquisición, la red local, el gateway, la conexión a Internet y la plataforma remota. Esto significa que el fallo de un único componente puede interrumpir toda la cadena de comunicación. Desde el punto de vista de la ciberseguridad, este tipo de situación se relaciona con los ataques o fallos de denegación de servicio, cuyo objetivo es impedir que un sistema preste servicio con normalidad.

1.4 Problemática de la disponibilidad en sistemas IoT

La disponibilidad es uno de los principales retos en los sistemas IoT, especialmente cuando estos se utilizan para monitorización continua. En este tipo de arquitecturas, el dispositivo de adquisición no trabaja de forma aislada, sino que depende de una cadena de comunicación compuesta por varios elementos: el propio nodo IoT, la red local, el gateway, la conexión a Internet y la plataforma cloud. Si cualquiera de estos elementos falla, los datos pueden dejar de llegar al sistema remoto, aunque el sensor o el cliente IoT sigan funcionando correctamente.

Esta problemática es especialmente importante porque, en muchos casos, el fallo no genera necesariamente un dato erróneo, sino una ausencia de información. En un sistema de monitorización ambiental, por ejemplo, la pérdida de conectividad puede provocar que la plataforma deje de recibir valores actualizados de temperatura o humedad. Si el usuario no detecta esta interrupción, podría interpretar que el entorno se mantiene estable, cuando en realidad simplemente no se están recibiendo datos nuevos.

Desde el punto de vista de la ciberseguridad, esta situación se relaciona con los ataques o fallos de denegación de servicio. Un ataque de denegación de servicio, o DoS, busca impedir que un sistema, recurso o servicio esté disponible para sus usuarios legítimos. En sistemas IoT, este efecto puede producirse de muchas formas: saturando la red, bloqueando el gateway, interrumpiendo la comunicación inalámbrica o provocando la caída de un elemento intermedio de la infraestructura.

1.5 Alta disponibilidad como mecanismo de resiliencia

La alta disponibilidad es una estrategia de diseño orientada a reducir el impacto de los fallos sobre la continuidad de un servicio. Su objetivo no es asumir que los sistemas nunca van a fallar, sino preparar la arquitectura para que, cuando se produzca una caída, exista un mecanismo capaz de mantener o recuperar el servicio en el menor tiempo posible. En este sentido, la alta disponibilidad está directamente

relacionada con el concepto de resiliencia, entendido como la capacidad de un sistema para resistir, adaptarse y recuperarse ante condiciones adversas.

En una arquitectura IoT, la alta disponibilidad puede aplicarse a distintos niveles: servidores, plataformas cloud, bases de datos, enlaces de comunicación o dispositivos de red. En este proyecto, el foco se sitúa en el gateway de salida, ya que es el elemento que permite al cliente IoT alcanzar Internet y enviar datos a ThingSpeak. Si el gateway principal falla y no existe una alternativa, la comunicación con la nube se interrumpe. Por tanto, este elemento constituye un punto crítico dentro de la arquitectura.

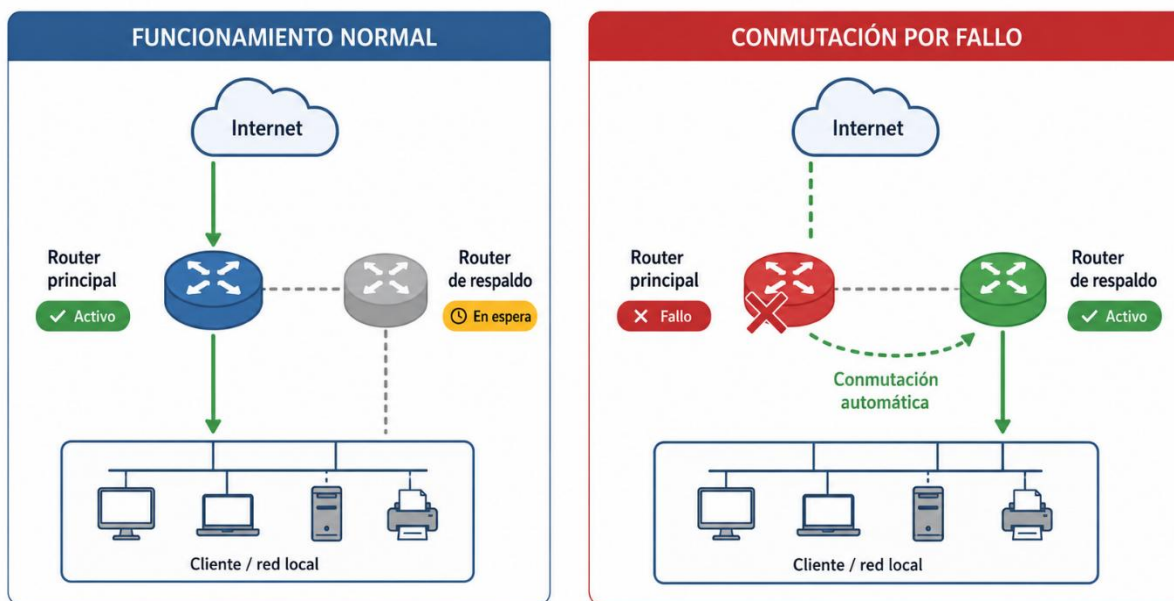


Figura 2. Esquema de alta disponibilidad con dos routers

1.6 Fundamentos de red necesarios para el proyecto

Para comprender la arquitectura implementada en este trabajo es necesario introducir algunos conceptos básicos de redes IP. Aunque el objetivo principal del proyecto está relacionado con la disponibilidad de un sistema IoT, dicha disponibilidad depende directamente de cómo se comunican los equipos, cómo se enruta el tráfico entre redes y cómo se realiza la conmutación entre routers cuando se produce un fallo. En este apartado se describen los fundamentos de red utilizados posteriormente en la implementación: comunicación TCP/IP, direccionamiento IP, gateway, enrutamiento, NAT, masquerading, VRRP, IP virtual y conmutación por fallo. Estos conceptos permiten entender por qué el Cliente-IoT puede comunicarse con ThingSpeak a través de los routers virtuales y cómo el sistema mantiene la conectividad cuando el Router-Principal deja de estar disponible.

1.6.1 Comunicación TCP/IP en sistemas IoT

El protocolo de control de transmisión (TCP) es un estándar de comunicación que permite que los programas de aplicaciones y los dispositivos informáticos intercambien mensajes de forma fiable a través de una red. [15]

Antes de enviar la información, TCP establece una conexión lógica entre el origen y el destino, y posteriormente divide los datos en segmentos para facilitar su transmisión. Una de sus características principales es que controla que la información llegue correctamente, en orden y sin pérdidas no

detectadas. Por este motivo, se utiliza en servicios que necesitan una comunicación fiable. En este proyecto, TCP interviene cuando el Cliente-IoT virtual envía datos a ThingSpeak mediante una petición HTTP.

1.6.2 Direccionamiento IP y gateway

Una dirección IP es un identificador numérico único que se asigna a cada dispositivo dentro de una red. Su función es permitir que los equipos puedan localizarse y comunicarse entre sí [16]. Las direcciones IP pueden utilizarse dentro de redes locales o en Internet. En una red local, cada dispositivo necesita una dirección dentro del mismo rango para poder comunicarse con el resto de equipos de esa red.

Cuando un equipo necesita comunicarse con otro dispositivo que está fuera de su propia red local, debe enviar el tráfico a una puerta de enlace o gateway. El gateway actúa como punto de salida hacia otras redes, es decir, recibe los paquetes generados por los dispositivos de la red local y los reenvía hacia su destino correspondiente, ya sea otra red interna o Internet [17].

1.6.3 Enrutamiento, NAT y masquerading

El enrutamiento es el proceso mediante el cual los paquetes de datos se encaminan desde un origen hasta un destino a través de una o varias redes. Cuando dos dispositivos se encuentran dentro de la misma red local, pueden comunicarse directamente. Sin embargo, cuando el destino se encuentra en otra red, el tráfico debe enviarse a un router que se encarga de decidir por qué interfaz o camino debe reenviarse el paquete. Para que un equipo pueda actuar como router, debe disponer de al menos dos interfaces de red y tener habilitado el reenvío de paquetes. Esto significa que el sistema no solo recibe tráfico destinado a él mismo, sino que también puede recibir paquetes por una interfaz y reenviarlos por otra. Esta función es fundamental en redes donde se necesita conectar una red interna con otra red externa.

NAT, o traducción de direcciones de red, es una técnica que permite modificar las direcciones IP de los paquetes cuando atraviesan un router. Su uso más habitual consiste en permitir que varios equipos de una red privada accedan a redes externas utilizando una dirección IP diferente, normalmente la dirección del router. De esta forma, los dispositivos internos pueden comunicarse con el exterior sin que sus direcciones privadas sean visibles directamente fuera de la red.

El masquerading es una forma específica de NAT utilizada habitualmente en sistemas Linux. Consiste en sustituir automáticamente la dirección IP de origen de los paquetes que salen de una red interna por la dirección IP de la interfaz externa del router. Gracias a este mecanismo, las respuestas procedentes del exterior pueden volver al router, que se encarga de reenviarlas al dispositivo interno correspondiente.

1.6.4 VRRP, IP virtual y conmutación por fallo

VRRP, o Virtual Router Redundancy Protocol, es un protocolo diseñado para proporcionar redundancia entre routers dentro de una misma red. Su objetivo es evitar que la caída de un único router deje sin conectividad a los dispositivos que dependen de él como puerta de enlace. Para ello, varios routers pueden formar un grupo de redundancia y compartir una misma dirección IP virtual.

Una IP virtual es una dirección que no pertenece de forma permanente a un único equipo físico o virtual. En lugar de estar asociada siempre al mismo router, puede ser asumida por diferentes routers según el estado del sistema. Normalmente, uno de los routers actúa como principal o maestro, mientras que otro

permanece como respaldo. Si el router principal funciona correctamente, la IP virtual se mantiene activa en él.

La conmutación por fallo, también conocida como **failover**, se produce cuando el router principal deja de estar disponible. En ese momento, el router de respaldo detecta la ausencia del router activo y asume la IP virtual. Gracias a este mecanismo, los dispositivos de la red pueden seguir utilizando la misma puerta de enlace, aunque internamente haya cambiado el router que presta el servicio.

1.7 Objetivos del trabajo

1.7.1 Objetivo principal

El objetivo principal de este Trabajo de Fin de Grado es diseñar e implementar un sistema IoT con mecanismos de alta disponibilidad que mantenga la continuidad del servicio ante escenarios de denegación de servicio o degradación del gateway principal. El trabajo no se centra únicamente en conseguir que el cliente pueda enviar datos a Internet, sino en analizar cómo puede mantenerse esa comunicación cuando se produce un fallo en un elemento crítico de la red. Mediante el uso de routers de respaldo, se pretende observar cómo afectan escenarios de denegación de servicio o de fallo sobre el router principal y ver cómo se comporta y recupera el sistema ante estos escenarios.

1.7.2 Objetivos específicos

Para alcanzar el objetivo principal, se definen los siguientes objetivos específicos:

1. Diseñar e implementar un nodo IoT basado en ESP capaz de adquirir datos de un sensor y transmitirlos periódicamente a una plataforma de monitorización mediante WiFi.
2. Implementar una infraestructura de red virtualizada compuesta por dos routers, permitiendo la conmutación automática ante fallos.
3. Modelar y reproducir en entorno controlado escenarios de degradación del router principal compatibles con ataques.
4. Diseñar mecanismos de detección de degradación basados en métricas de red y disponibilidad del servicio.
5. Implementar y evaluar el proceso de failover automático hacia el router alternativo, analizando métricas como:
 - Tiempo de conmutación.
 - Pérdida de mensajes.
 - Impacto en latencia.
 - Disponibilidad global del sistema.
6. Analizar la arquitectura propuesta como mecanismo de resiliencia frente a ataques de denegación de servicio en entornos IoT.

CAPÍTULO 2

Diseño de la arquitectura propuesta

2.1 Planteamiento inicial del sistema IoT físico

El planteamiento inicial del proyecto partía de un sistema IoT físico basado en un módulo ESP-01, perteneciente a la familia de dispositivos basados en el chip ESP8266, conectado a un sensor DHT11 de temperatura y humedad. La función del nodo consistía en adquirir datos ambientales y enviarlos periódicamente a la plataforma ThingSpeak mediante conexión WiFi. Este planteamiento permitía representar un caso sencillo y realista de monitorización ambiental, en el que un dispositivo de bajo coste captura información del entorno y la transmite hacia una plataforma cloud para su visualización.

En esta primera fase, el funcionamiento del sistema era directo: el nodo IoT se conectaba a la red WiFi doméstica y utilizaba el router de casa como puerta de enlace hacia Internet. Una vez conectado, el dispositivo generaba peticiones HTTP hacia ThingSpeak, incluyendo los valores de temperatura y humedad en los campos correspondientes del canal configurado. Este esquema permitió validar que el sensor, el microcontrolador, la conexión WiFi y la plataforma cloud funcionaban correctamente de forma independiente.

La parte principal del proyecto, sin embargo, consistía en introducir una infraestructura de red intermedia capaz de proporcionar alta disponibilidad para los datos enviados. Para ello, se planteó inicialmente que el nodo físico dejara de depender directamente del router doméstico y pasara a utilizar como gateway un router virtual creado en una máquina virtual Ubuntu. Este router virtual actuaría como intermediario entre el dispositivo IoT y la salida a Internet.

La idea original, por tanto, consistía en situar el sistema IoT físico detrás de una infraestructura formada por dos routers virtuales: un Router-Principal y un Router-Respaldo. En condiciones normales, el tráfico del nodo IoT pasaría por el Router-Principal. En caso de fallo, el Router-Respaldo asumiría la función de gateway, permitiendo mantener la comunicación con ThingSpeak. Este diseño permitía estudiar la disponibilidad del sistema y analizar cómo afectaba la caída de un router al envío periódico de datos.

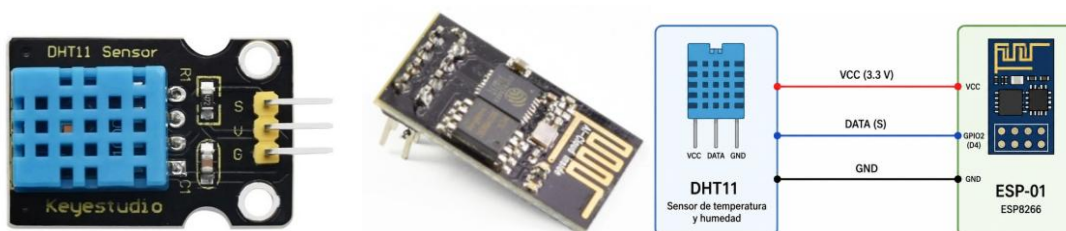


Figura 3. Sensor DHT11, módulo ESP-01 y esquema de conexión

2.2 Limitaciones encontradas en la integración física con routers virtuales

Durante el desarrollo del proyecto se identificaron limitaciones técnicas al intentar integrar el nodo IoT físico, formado por el módulo ESP-01 y el sensor DHT11, directamente con los routers virtuales. El principal problema estaba relacionado con la topología de red disponible y con las restricciones del

hardware utilizado. El equipo anfitrión solo disponía de una interfaz WiFi, utilizada para conectarse a la red doméstica, y las máquinas virtuales se ejecutaban dentro de VirtualBox sobre dicho equipo.

La conexión de un dispositivo físico externo, como una ESP-01 conectada por WiFi, a una red interna virtual no es directa si no existe una interfaz física adicional que permita separar ambos dominios de red. Esta limitación impedía construir de forma limpia la topología inicialmente deseada, donde el nodo físico quedara completamente detrás de los routers virtuales.

Como consecuencia, el sistema físico podía enviar datos correctamente a ThingSpeak cuando se conectaba directamente al router doméstico, pero no se pudo realizar una integración estable y reproducible cuando se intentaba forzar su tráfico a través del router virtual por limitaciones en el hardware disponible. Cabe recalcar que el planteamiento inicial se podría reproducir fácilmente disponiendo de dos adaptadores WiFi adicionales para el ordenador, utilizando uno para cada uno de los routers virtuales creados.

Al no disponer del hardware necesario, se optó por estudiar el comportamiento del sistema IoT y recrearlo simulándolo con otra máquina virtual (Cliente-IoT) que reprodujera el comportamiento del nodo físico original pero todo virtualizado para así solventar el problema y poder desarrollar el estudio principal de este trabajo.

2.3 Adaptación metodológica: sustitución del nodo físico por un Cliente-IoT virtual

A partir de las limitaciones descritas anteriormente, se decidió adaptar la metodología del proyecto sustituyendo el nodo IoT físico por una máquina virtual denominada Cliente-IoT. Esta decisión no modifica el objetivo principal del trabajo, ya que el estudio se centra en la disponibilidad de la infraestructura de red y no en el nodo físico del sistema.

El Cliente-IoT virtual reproduce el comportamiento funcional del nodo original basado en ESP-01 y DHT11. En lugar de leer directamente valores de temperatura y humedad desde un sensor físico, la máquina virtual genera valores realistas y los envía a ThingSpeak mediante peticiones HTTP. De esta forma, se mantiene el flujo esencial del sistema IoT: generación de datos, transmisión a través de la red, salida hacia Internet y recepción en una plataforma cloud.

Esta adaptación aporta varias ventajas desde el punto de vista experimental. En primer lugar, permite situar al Cliente-IoT dentro de una red interna de VirtualBox, completamente separada de la red doméstica. En segundo lugar, garantiza que todo el tráfico generado por el cliente atraviese obligatoriamente los routers virtuales. En tercer lugar, facilita la repetición y validación de las pruebas, ya que desde la máquina virtual pueden ejecutarse comandos de diagnóstico como ping, ip route, curl o consultas de tabla ARP.

Por tanto, la sustitución del nodo físico por el Cliente-IoT no debe interpretarse como una eliminación del enfoque IoT, sino como una adaptación metodológica para estudiar de forma más precisa la resiliencia de la infraestructura de red.

2.4 Arquitectura final del sistema

La arquitectura final del sistema se basa en un entorno virtualizado diseñado para estudiar la disponibilidad de una comunicación IoT hacia una plataforma cloud. El sistema está formado por un Cliente-IoT virtual, dos routers virtuales Ubuntu, el router doméstico y la plataforma ThingSpeak.

El Cliente-IoT virtual representa el nodo encargado de generar y enviar los datos. Esta máquina se encuentra situada dentro de una red interna de VirtualBox, separada de la red doméstica. Su función es simular el comportamiento del nodo físico original, generando valores de prueba y enviándolos periódicamente a ThingSpeak mediante peticiones HTTP. Al estar conectado únicamente a la red interna, el Cliente-IoT no dispone de salida directa a Internet, por lo que todo su tráfico debe atravesar obligatoriamente la infraestructura de routers virtuales.

Los routers virtuales constituyen el núcleo de la arquitectura de alta disponibilidad. El Router-Principal actúa como gateway activo en condiciones normales, mientras que el Router-Respaldo permanece preparado para asumir esta función en caso de fallo. Ambos routers disponen de una interfaz conectada a la red interna del Cliente-IoT y otra interfaz conectada a la red externa con salida hacia el router doméstico. De esta forma, los routers actúan como intermediarios entre la red IoT virtual y la red doméstica.

El router doméstico mantiene su función habitual como salida real hacia Internet. Los routers virtuales utilizan esta red externa para alcanzar la plataforma ThingSpeak.

Por su parte, ThingSpeak actúa como destino final de los datos enviados por el Cliente-IoT y permite comprobar si la comunicación extremo a extremo se mantiene correctamente.



Figura 4. Esquema de la arquitectura final del sistema

2.5 Elementos del sistema

2.5.1 Cliente-IoT virtual

El Cliente-IoT virtual es la máquina encargada de simular el comportamiento del nodo IoT físico planteado inicialmente. Su función es generar datos de prueba equivalentes a medidas ambientales y enviarlos hacia la plataforma ThingSpeak mediante peticiones HTTP.

2.5.2 Router-Principal

El Router-Principal es el router virtual que actúa como gateway activo durante el funcionamiento normal del sistema. Su función es recibir el tráfico procedente del Cliente-IoT, reenviarlo hacia la red externa y aplicar las reglas necesarias de enrutamiento y NAT.

2.5.3 Router-Respaldo

El Router-Respaldo es el segundo router virtual de la arquitectura y permanece preparado para asumir el servicio si el Router-Principal falla. En condiciones normales se mantiene en espera, pero mediante Keepalived puede asumir automáticamente la IP virtual y convertirse en el nuevo gateway activo.

2.5.4 Router doméstico

El router doméstico es el dispositivo físico que proporciona la salida real a Internet desde la red local. En la arquitectura propuesta, los routers virtuales utilizan este equipo como siguiente salto para alcanzar servicios externos.

2.5.5 Plataforma ThingSpeak

ThingSpeak es la plataforma cloud utilizada como destino final de los datos generados por el Cliente-IoT virtual. Permite recibir, almacenar y visualizar los valores enviados por el sistema, actuando como punto de validación de la comunicación extremo a extremo.

2.6 Direccionamiento IP del sistema

El direccionamiento IP del sistema se ha diseñado separando dos redes principales: la red IoT interna y la red externa. Esta separación permite que el Cliente-IoT virtual no tenga acceso directo a Internet, sino que deba comunicarse obligatoriamente a través de los routers virtuales.

La red IoT interna corresponde al segmento donde se encuentra el Cliente-IoT virtual y las interfaces internas de los routers. Esta red utiliza el rango 192.168.50.0/24. Dentro de esta red, al Cliente-IoT se le ha asignado la dirección 192.168.50.50. El Router-Principal utiliza la dirección interna 192.168.50.201 y el Router-Respaldo utiliza la dirección interna 192.168.50.202.

La puerta de enlace configurada en el Cliente-IoT no es la dirección física de ninguno de los dos routers, sino una IP virtual compartida: 192.168.50.250. Esta dirección funciona como gateway del cliente y puede estar activa en el Router-Principal o en el Router-Respaldo dependiendo del estado del sistema. En funcionamiento normal, la IP virtual se encuentra asociada al Router-Principal. Si este falla, el Router-Respaldo asume dicha dirección y continúa proporcionando conectividad.

La red externa corresponde al segmento conectado al router doméstico, que proporciona la salida real hacia Internet. Esta red utiliza el rango 192.168.1.0/24. En ella, el Router-Principal dispone de la dirección externa 192.168.1.201 y el Router-Respaldo de la dirección externa 192.168.1.202. Ambos utilizan como siguiente salto el router doméstico.

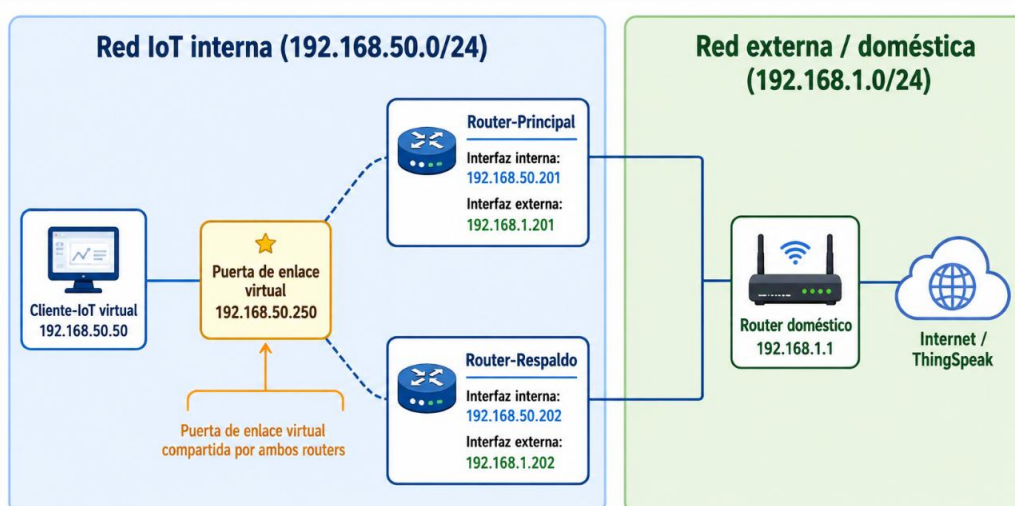


Figura 5. Esquema de las conexiones del sistema

2.7 Flujo de comunicación extremo a extremo

En primer lugar, el Cliente-IoT genera los datos que simulan las medidas ambientales del sistema. Estos datos se envían mediante una petición HTTP dirigida a ThingSpeak. Como el cliente se encuentra en la red IoT interna y no tiene salida directa a Internet, la petición se entrega a su puerta de enlace configurada, que es la IP virtual compartida por los routers.

En funcionamiento normal, la IP virtual está activa en el Router-Principal. Por tanto, el tráfico del Cliente-IoT llega primero a este router. El Router-Principal reenvía el paquete desde su interfaz interna hacia su interfaz externa y aplica la traducción de direcciones necesaria mediante NAT, permitiendo que el tráfico procedente de la red IoT interna pueda salir hacia la red doméstica.

A continuación, el paquete llega al router doméstico, que actúa como puerta de salida real hacia Internet. Desde este punto, la comunicación continúa hasta alcanzar ThingSpeak, donde los datos son recibidos y almacenados en el canal correspondiente. La correcta aparición de los datos en la plataforma permite validar que todo el camino de comunicación funciona correctamente.

En caso de fallo del Router-Principal, el Router-Respaldo asume la IP virtual y pasa a encargarse del reenvío del tráfico. Desde el punto de vista del Cliente-IoT, el gateway sigue siendo el mismo. La diferencia se produce internamente en la infraestructura: el tráfico deja de pasar por el Router-Principal y se encamina a través del Router-Respaldo. Este comportamiento permite mantener la continuidad del servicio ante la caída del router activo.

La Figura 6 muestra la visualización de los datos recibidos en la plataforma ThingSpeak. En ella se representan los valores enviados por el sistema IoT correspondientes a las variables de humedad y temperatura. La gráfica de la izquierda corresponde al campo de humedad, mientras que la gráfica de la derecha corresponde al campo de temperatura.

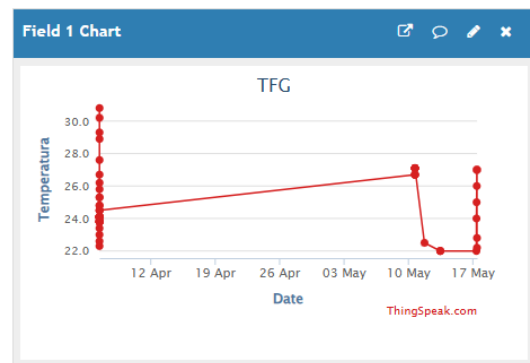
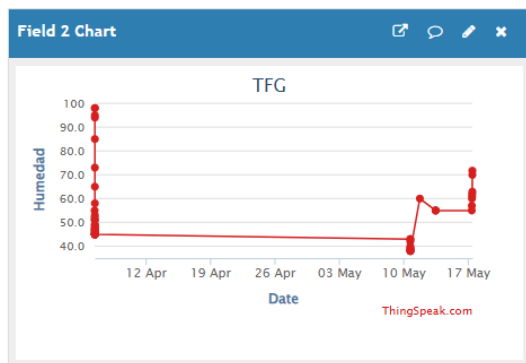


Figura 6. Visualización de los datos en ThingSpeak

CAPÍTULO 3

Implementación de la infraestructura virtualizada

3.1 Entorno de virtualización utilizado

La implementación del sistema se ha realizado mediante un entorno virtualizado sobre un equipo anfitrión. Para ello se ha utilizado Oracle VM VirtualBox, una herramienta de virtualización que permite crear y ejecutar varias máquinas virtuales sobre un mismo ordenador físico. Esta solución permite simular una infraestructura de red completa sin necesidad de disponer de varios equipos físicos independientes.

El uso de virtualización resulta adecuado para este proyecto porque permite crear diferentes máquinas con funciones específicas dentro de la arquitectura. En este caso, se han desplegado tres máquinas virtuales principales: un Cliente-IoT, un Router-Principal y un Router-Respaldo. Cada una de ellas ejecuta Ubuntu 24.04 y tiene una función concreta dentro del sistema.

El entorno virtualizado permite también configurar diferentes tipos de interfaces de red, lo que resulta esencial para separar la red IoT interna de la red externa con salida a Internet. Gracias a esta separación, el Cliente-IoT no se conecta directamente a Internet, sino que debe atravesar obligatoriamente los routers virtuales.

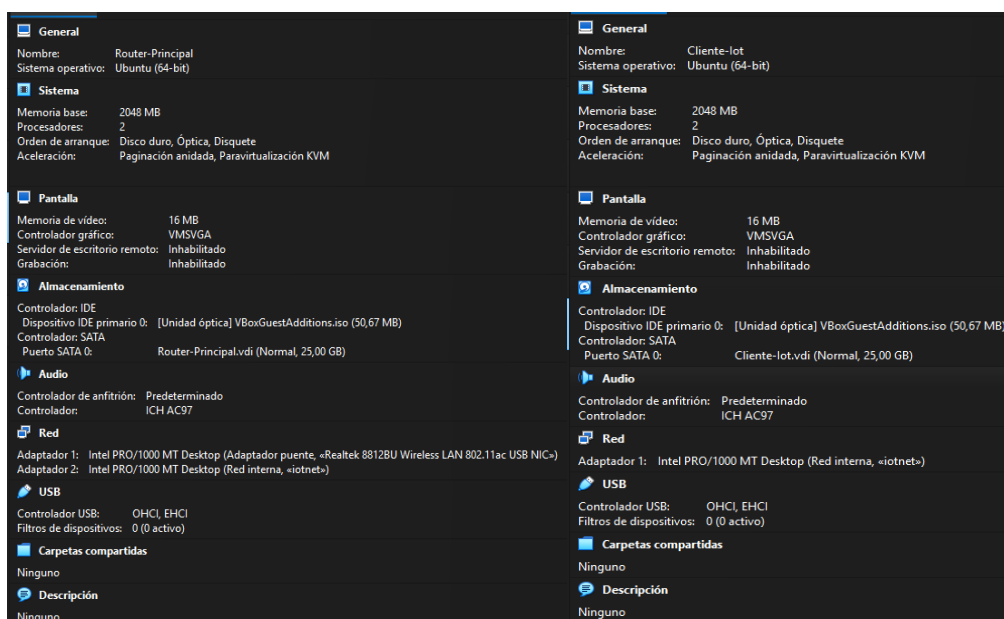


Figura 7. Características de los routers virtuales y del Cliente-IoT.

3.2 Configuración de interfaces de red en VirtualBox

La configuración de red en VirtualBox es uno de los elementos más importantes de la implementación para llevar a cabo el estudio planteado en este trabajo, ya que permite separar la red IoT interna de la red externa. Para conseguirlo, se utilizaron dos tipos de conexión: red interna y adaptador puente.

La red interna se utilizó para conectar entre sí el Cliente-IoT, el Router-Principal y el Router-Respaldo. Esta red, denominada **iotnet**, representa la red IoT del sistema. Al estar aislada del resto de redes, el Cliente-IoT no dispone de salida directa a Internet y necesita utilizar los routers virtuales como puerta de enlace. Esta configuración permite reproducir de forma controlada el comportamiento de una red IoT situada detrás de una infraestructura de routing.

Por otro lado, los routers virtuales disponen también de un adaptador configurado en modo puente. Este adaptador permite que Router-Principal y Router-Respaldo se conecten a la red doméstica del equipo anfitrión, obteniendo conectividad hacia el router físico de casa y, posteriormente, hacia Internet. De esta forma, cada router virtual actúa como intermediario entre la red IoT interna y la red externa.

El Cliente-IoT solo dispone de una interfaz de red conectada a la red interna **iotnet**. En cambio, cada router virtual dispone de dos interfaces: una conectada a la red externa mediante adaptador puente y otra conectada a la red interna **iotnet**. Esta configuración es fundamental para que los routers puedan recibir tráfico desde el cliente, reenviarlo hacia Internet y aplicar posteriormente las reglas de NAT y alta disponibilidad.

La separación entre interfaces permite construir una topología clara y controlada: el Cliente-IoT se comunica con su gateway dentro de la red interna, los routers gestionan el tráfico entre redes y el router doméstico proporciona la salida final hacia Internet. Esta estructura reproduce el funcionamiento de una arquitectura real con red interna y gateway de salida, pero implementada de forma virtualizada.

3.3 Configuración del Router-Principal

3.3.1 Direccionamiento IP

El Router-Principal se configuró con dos direcciones IP estáticas, una para cada interfaz de red. La interfaz interna, conectada a la red IoT, utiliza la dirección 192.168.50.201/24. Esta interfaz permite la comunicación con el Cliente-IoT y con el Router-Respaldo dentro de la red interna. La interfaz externa, conectada a la red doméstica, utiliza la dirección 192.168.1.201/24 y tiene como puerta de enlace el router doméstico.

Esta configuración permite que el Router-Principal actúe como punto intermedio entre ambas redes. Por un lado, recibe tráfico procedente del Cliente-IoT a través de la red 192.168.50.0/24. Por otro lado, puede reenviar dicho tráfico hacia la red doméstica 192.168.1.0/24 y, posteriormente, hacia Internet. La correcta asignación de estas direcciones se comprobó mediante comandos de diagnóstico de red, verificando que ambas interfaces se encontraban activas.

```
vboxuser@Router-Principal:~/Desktop$ ip -br addr
lo                UNKNOWN          127.0.0.1/8 ::1/128
enp0s3            UP                192.168.1.201/24
enp0s8            UP                192.168.50.201/24 192.168.50.250/24
```

Figura 8. Direcciones IP Router-Principal

3.3.2 Activación del reenvío IPv4

Para que una máquina Linux pueda actuar como router, no basta con que disponga de dos interfaces de red. Es necesario activar el reenvío de paquetes IPv4, ya que por defecto un equipo final no reenvía tráfico entre interfaces. En este proyecto se habilitó la opción **net.ipv4.ip_forward**, permitiendo que el Router-Principal pudiera reenviar paquetes entre la red IoT interna y la red externa.

Además, se ajustaron parámetros relacionados con el filtrado de rutas inversas y el envío de redirecciones ICMP. Estas opciones permiten evitar comportamientos no deseados cuando el equipo trabaja como router entre redes distintas. Una vez aplicados los cambios, se verificó que el valor de **net.ipv4.ip_forward** estaba activado, confirmando que el sistema estaba preparado para reenviar tráfico.

```
vboxuser@Router-Principal:~/Desktop$ sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 1
```

Figura 9. Comprobación reenvío

3.3.3 Configuración de nftables y NAT

Una vez activado el reenvío IPv4, fue necesario configurar reglas de NAT mediante nftables. El Cliente-IoT pertenece a la red interna 192.168.50.0/24, por lo que sus direcciones privadas no pueden salir directamente hacia Internet sin traducción. Para resolverlo, el Router-Principal aplica una regla de masquerading sobre el tráfico procedente de la red IoT interna que sale por la interfaz externa.

La configuración de nftables incluye una tabla específica para el router, una cadena de reenvío y una cadena de postrouting. La cadena de reenvío permite aceptar el tráfico entre la interfaz interna y la externa, así como las respuestas asociadas a conexiones ya establecidas. La cadena de postrouting aplica la traducción de direcciones, sustituyendo la dirección origen del Cliente-IoT por la dirección externa del Router-Principal.

Gracias a esta configuración, el Cliente-IoT puede comunicarse con servicios externos como ThingSpeak aunque se encuentre en una red interna aislada. Desde el punto de vista de Internet, el tráfico parece proceder del Router-Principal, mientras que internamente el router se encarga de mantener la correspondencia entre las conexiones.

```
vboxuser@Router-Principal:~/Desktop$ sudo nft -a list ruleset
table ip iot_router { # handle 6
    chain forward { # handle 1
        type filter hook forward priority filter; policy accept;
        iifname "enp0s8" oifname "enp0s3" counter packets 520 bytes 33896 accept # handle 3
        iifname "enp0s3" oifname "enp0s8" ct state established,related counter packets 684 bytes 951347 accept # handle 4
    }

    chain postrouting { # handle 2
        type nat hook postrouting priority srcnat; policy accept;
        ip saddr 192.168.50.0/24 oifname "enp0s3" counter packets 16 bytes 1312 masquerade # handle 5
    }
}
```

Figura 10. Reglas nftables y NAT

3.3.4 Persistencia de la configuración

Para que la configuración del router no se perdiera tras reiniciar la máquina virtual, se configuró la persistencia de los parámetros de red y de las reglas de nftables. Las direcciones IP estáticas se mantuvieron mediante la configuración de NetworkManager, mientras que las reglas de filtrado y NAT se guardaron en el archivo de configuración de nftables.

Además, se habilitó el servicio **nftables** para que las reglas se cargaran automáticamente al iniciar el sistema. Esta persistencia es fundamental en el proyecto, ya que permite que el Router-Principal conserve su función de gateway sin necesidad de volver a introducir manualmente todas las reglas después de cada reinicio. La comprobación se realizó reiniciando el servicio y verificando posteriormente que las reglas seguían presentes en el sistema.

```
vboxuser@Router-Principal:~/Desktop$ sudo systemctl is-enabled nftables
sudo systemctl is-active nftables
sudo nft -a list ruleset
[sudo] password for vboxuser:
enabled
active
table ip iot_router { # handle 6
    chain forward { # handle 1
        type filter hook forward priority filter; policy accept;
        iifname "enp0s8" oifname "enp0s3" counter packets 520 bytes 33896 accept # handle 3
        iifname "enp0s3" oifname "enp0s8" ct state established,related counter packets 684 bytes 951347 accept # handle 4
    }

    chain postrouting { # handle 2
        type nat hook postrouting priority srcnat; policy accept;
        ip saddr 192.168.50.0/24 oifname "enp0s3" counter packets 16 bytes 1312 masquerade # handle 5
    }
}
```

Figura 11. Reactivación del servicio para confirmar la persistencia

3.4 Configuración del Router-Respaldo

3.4.1 Direcccionamiento IP

El Router-Respaldo también dispone de dos interfaces de red. La interfaz interna, conectada a la red IoT, se configuró con la dirección 192.168.50.202/24. Esta dirección permite que el router se comunique con el Cliente-IoT y con el Router-Principal dentro de la red interna. La interfaz externa, conectada a la red doméstica, se configuró con la dirección 192.168.1.202/24 y utiliza como puerta de enlace el router doméstico.

Esta configuración permite que el Router-Respaldo pueda realizar las mismas funciones de routing que el Router-Principal. Aunque inicialmente permanece como router secundario, dispone de conectividad tanto hacia la red IoT interna como hacia la red externa. La correcta configuración de sus interfaces se verificó mediante comandos de visualización de direcciones IP y rutas.

```
vboxuser@Router-Respaldo:~/Desktop$ ip -br addr
lo                UNKNOWN          127.0.0.1/8 ::1/128
enp0s3            UP                192.168.1.202/24
enp0s8            UP                192.168.50.202/24 192.168.50.250/24
```

Figura 12. Direcciones IP Router-Respaldo

3.4.2 Activación del reenvío IPv4

Al igual que en el Router-Principal, se activó el reenvío IPv4 en el Router-Respaldo. Esta configuración permite que la máquina pueda reenviar tráfico entre sus dos interfaces de red. Sin esta opción, el Router-Respaldo podría comunicarse con ambas redes de forma individual, pero no actuar como intermediario entre ellas.

De esta manera, el sistema queda preparado para actuar como router real dentro de la arquitectura. La activación del reenvío IPv4 se comprobó verificando el valor de **net.ipv4.ip_forward**.

```
vboxuser@Router-Respaldo:~/Desktop$ sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 1
```

Figura 13. Comprobación reenvío

3.4.3 Configuración de nftables y NAT

El Router-Respaldo requiere una configuración de nftables equivalente a la del Router-Principal. Las reglas permiten aceptar el tráfico procedente de la red IoT interna hacia la red externa y aplicar NAT mediante masquerading. De esta forma, cuando el Router-Respaldo asume el papel de gateway activo, puede proporcionar salida a Internet al Cliente-IoT.

La cadena de reenvío permite el paso de paquetes desde la interfaz interna hacia la externa y acepta las respuestas de conexiones ya establecidas. La cadena de postrouting aplica la traducción de direcciones al tráfico procedente de la red 192.168.50.0/24 cuando sale hacia la red doméstica. Esta configuración garantiza que el Router-Respaldo pueda reemplazar funcionalmente al Router-Principal en caso de fallo.

```
vboxuser@Router-Respaldo:~/Desktop$ sudo nft -a list ruleset
[sudo] password for vboxuser:
table ip iot_router { # handle 6
  chain forward { # handle 1
    type filter hook forward priority filter; policy accept;
    iifname "enp0s8" oifname "enp0s3" counter packets 0 bytes 0 accept # handle 3
    iifname "enp0s3" oifname "enp0s8" ct state established,related counter packets 0 bytes 0 accept # handle 4
  }

  chain postrouting { # handle 2
    type nat hook postrouting priority srcnat; policy accept;
    ip saddr 192.168.50.0/24 oifname "enp0s3" counter packets 0 bytes 0 masquerade # handle 5
  }
}
```

Figura 14. Reglas nftables y NAT

3.4.4 Persistencia de la configuración

La configuración del Router-Respaldo también se hizo persistente para evitar que se perdiera tras reiniciar la máquina virtual. Las direcciones IP se conservaron mediante **NetworkManager** y las reglas de nftables se guardaron en el archivo de configuración correspondiente. Posteriormente, se habilitó el servicio **nftables** para que las reglas se cargaran automáticamente durante el arranque.

Esta persistencia es especialmente importante en el Router-Respaldo, ya que debe estar preparado en todo momento para asumir el servicio. Si sus reglas de NAT o su configuración de reenvío no se cargaran correctamente, el sistema podría detectar el fallo del Router-Principal pero no sería capaz de mantener la comunicación hacia Internet. Por ello, se verificó que tras reiniciar el servicio las reglas continuaban activas.

3.5 Configuración del Cliente-IoT

3.5.1 Direccionamiento IP

El Cliente-IoT se configuró con una dirección IP estática dentro de la red IoT interna. La dirección asignada fue 192.168.50.50/24, perteneciente al rango 192.168.50.0/24. Esta red es compartida con las interfaces internas del Router-Principal y del Router-Respaldo.

La configuración estática permite mantener siempre la misma dirección en el cliente, lo que facilita las pruebas de conectividad, el análisis de tráfico y la documentación del sistema. Además, evita depender de un servidor DHCP dentro de la red interna, simplificando la arquitectura del entorno de pruebas.

Con esta configuración, el Cliente-IoT puede comunicarse con los routers virtuales dentro de la red interna. Sin embargo, para poder alcanzar Internet necesita utilizar una puerta de enlace, ya que su interfaz no está conectada directamente a la red externa.

```
vboxuser@Cliente-Iot:~/Desktop$ ip -br addr
lo                UNKNOWN    127.0.0.1/8 ::1/128
enp0s3            UP          192.168.50.50/24
```

Figura 15. Direcciones IP Cliente-IoT

3.5.2 Configuración inicial del Gateway

La puerta de enlace configurada en el Cliente-IoT es la IP virtual 192.168.50.250. Esta dirección no pertenece de forma permanente a una máquina concreta, sino que es compartida por el Router-Principal y el Router-Respaldo mediante el mecanismo de alta disponibilidad implementado posteriormente.

En funcionamiento normal, la IP virtual está activa en el Router-Principal, por lo que el tráfico generado por el Cliente-IoT se encamina a través de este router. Si el Router-Principal deja de estar disponible, el Router-Respaldo asume la IP virtual y continúa actuando como gateway del cliente. Desde el punto de vista del Cliente-IoT, la puerta de enlace no cambia, aunque internamente cambie el router que presta el servicio.

Esta configuración permite desacoplar al cliente de una dirección física concreta. En lugar de depender directamente de 192.168.50.201 o 192.168.50.202, el Cliente-IoT utiliza siempre 192.168.50.250. Esto resulta imprescindible para conseguir la conmutación automática ante fallos sin tener que modificar manualmente la configuración de red del cliente.

```
vboxuser@Cliente-Iot:~/Desktop$ ip route
default via 192.168.50.250 dev enp0s3 proto static metric 100
192.168.50.0/24 dev enp0s3 proto kernel scope link src 192.168.50.50 metric 100
```

Figura 16. Dirección gateway

3.5.3 Pruebas de conectividad

Una vez configurado el direccionamiento IP y la puerta de enlace, se realizaron varias pruebas para verificar que el Cliente-IoT podía comunicarse correctamente a través de la infraestructura de red. En primer lugar, se comprobó la conectividad con la IP virtual configurada como gateway. Esta prueba permite confirmar que el cliente puede alcanzar la puerta de enlace de la red IoT interna.

Posteriormente, se comprobó la comunicación con direcciones externas, como servidores de Internet. Esta prueba permite verificar que los routers virtuales están reenviando correctamente el tráfico desde la red interna hacia la red doméstica y que las reglas de NAT están funcionando. Si el cliente puede alcanzar una dirección externa, significa que el tráfico está atravesando correctamente el router activo.

Finalmente, se verificó la comunicación con ThingSpeak mediante una petición HTTP. Esta prueba es especialmente importante, ya que reproduce el comportamiento real esperado del Cliente-IoT: enviar datos hacia una plataforma cloud. La recepción correcta de la respuesta de ThingSpeak confirma que la comunicación extremo a extremo funciona correctamente, desde el Cliente-IoT hasta la plataforma remota.

Estas pruebas permiten validar que el Cliente-IoT está correctamente integrado en la arquitectura y que depende de los routers virtuales para acceder a Internet. Por tanto, el cliente queda preparado para las pruebas posteriores de alta disponibilidad y conmutación por fallo.

En la imagen de a continuación se verifica el correcto funcionamiento, el curl devuelve el número actual de petición guardada en la plataforma. En este caso es el dato 63 dentro de la plataforma.

```
vboxuser@Cliente-Iot:~/Desktop$ ping -c 4 192.168.50.250
PING 192.168.50.250 (192.168.50.250) 56(84) bytes of data.
64 bytes from 192.168.50.250: icmp_seq=1 ttl=64 time=0.850 ms
64 bytes from 192.168.50.250: icmp_seq=2 ttl=64 time=0.598 ms
64 bytes from 192.168.50.250: icmp_seq=3 ttl=64 time=0.488 ms
64 bytes from 192.168.50.250: icmp_seq=4 ttl=64 time=0.856 ms

--- 192.168.50.250 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3024ms
rtt min/avg/max/mdev = 0.488/0.698/0.856/0.159 ms
vboxuser@Cliente-Iot:~/Desktop$ ping -c 4 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=24.9 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=20.9 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=18.3 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=116 time=41.3 ms

--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3009ms
rtt min/avg/max/mdev = 18.323/26.360/41.340/8.961 ms
vboxuser@Cliente-Iot:~/Desktop$ curl "http://api.thingspeak.com/update?api_key=7SRDCT0WWWTQINA9&field1=55"
63vboxuser@Cliente-Iot:~/Desktop$
```

Figura 17. Comprobación funcionamiento del sistema

3.6 Integración con ThingSpeak

La integración con ThingSpeak permite comprobar que el Cliente-IoT virtual no solo dispone de conectividad hacia Internet, sino que también es capaz de enviar datos a una plataforma cloud externa. En este proyecto, ThingSpeak actúa como destino final de la información generada por el cliente y permite validar la comunicación extremo a extremo de la arquitectura.

ThingSpeak organiza la información mediante canales. Cada canal dispone de varios campos donde pueden almacenarse valores numéricos enviados por dispositivos IoT. En este caso, el Cliente-IoT envía valores de prueba que simulan medidas ambientales, como temperatura y humedad. Estos valores se transmiten mediante una petición HTTP que incluye la clave de escritura del canal y los campos que se desean actualizar.

La comunicación se realiza desde el Cliente-IoT utilizando la herramienta **curl**. Esta herramienta permite generar peticiones HTTP desde la terminal, reproduciendo de forma sencilla el comportamiento que tendría un nodo IoT al enviar datos hacia una API en la nube. El uso de **curl** facilita además la comprobación del sistema, ya que permite observar directamente la respuesta devuelta por ThingSpeak. El comando utilizado para enviar datos a la plataforma incluye la Write API Key del canal y los valores asociados a los campos correspondientes. Cuando ThingSpeak recibe correctamente la petición, devuelve un número mayor que cero, que representa el identificador de la nueva entrada almacenada en el canal. Si la respuesta es cero, normalmente indica que el dato no ha sido aceptado, por ejemplo, por superar el límite de frecuencia de actualización de la plataforma.

3.7 Validación inicial del envío de datos

Una vez configurado el Cliente-IoT, los routers virtuales y la integración con ThingSpeak, se realizó una validación inicial del sistema. El objetivo de esta prueba fue comprobar que la arquitectura básica de comunicación funcionaba correctamente antes de introducir las pruebas específicas de alta disponibilidad.

La validación consistió en enviar datos desde el Cliente-IoT virtual hacia ThingSpeak mediante una petición HTTP. Para ello, el cliente generó valores de prueba asociados a los campos del canal configurado y los transmitió utilizando la herramienta **curl**. Esta petición debía atravesar la red IoT interna, alcanzar el router activo, salir hacia la red doméstica mediante NAT, acceder a Internet y llegar finalmente a la plataforma cloud.

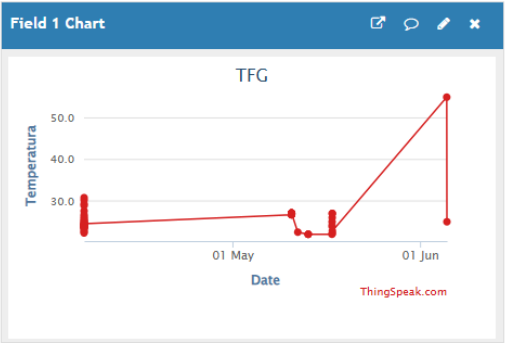
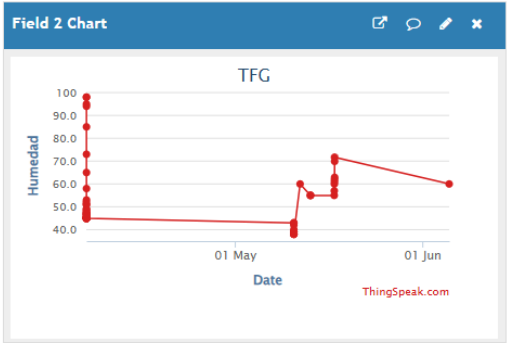
La respuesta devuelta por ThingSpeak permitió confirmar si el dato había sido recibido correctamente. Cuando la plataforma acepta una actualización, devuelve un número mayor que cero, correspondiente al identificador de la entrada creada en el canal. En la prueba realizada, la respuesta obtenida confirmó que los datos enviados por el Cliente-IoT fueron almacenados correctamente.

Esta validación demuestra que el flujo de comunicación extremo a extremo estaba operativo: el Cliente-IoT disponía de direccionamiento correcto, el router activo reenviaba el tráfico, las reglas de NAT funcionaban adecuadamente y la plataforma ThingSpeak recibía los datos. Por tanto, el sistema quedaba preparado para la siguiente fase del proyecto, centrada en la implementación de alta disponibilidad y en el análisis de la conmutación entre routers.

```
vboxuser@Cliente-Iot:~/Desktop$ curl "http://api.thingspeak.com/update?api_key=7SRDCT0WWMTQINA9&field1=25&field2=60"
64vboxuser@Cliente-Iot:~/Desktop$
```

Channel Stats

Created: 2.months.ago
Last entry: less.than.a.minute.ago
Entries: 64



Figuras 18 y 19. Envío de solicitud con temperatura = 22, humedad = 60 y validación en ThingSpeak

CAPÍTULO 4

Implementación de alta disponibilidad

4.1 Diseño de la solución de alta disponibilidad

La solución de alta disponibilidad implementada en este trabajo tiene como objetivo evitar que la caída del router principal interrumpa completamente la comunicación entre el Cliente-IoT y la plataforma ThingSpeak. En una arquitectura con un único gateway, el fallo de dicho equipo provocaría que el cliente perdiera la salida hacia Internet. Para reducir este riesgo, se introduce un segundo router virtual capaz de asumir el servicio cuando el router principal deja de estar disponible.

La arquitectura se basa en dos routers virtuales Ubuntu: el Router-Principal y el Router-Respaldo. En condiciones normales, el Router-Principal actúa como gateway activo de la red IoT interna y se encarga de reenviar el tráfico del Cliente-IoT hacia la red externa. El Router-Respaldo permanece preparado para asumir esta función si se produce un fallo en el router activo.

Para que el Cliente-IoT no dependa directamente de la dirección física de un router concreto, se utiliza una IP virtual compartida. Esta IP virtual actúa como puerta de enlace del cliente y puede estar activa en uno u otro router dependiendo del estado del sistema. En funcionamiento normal, la IP virtual se encuentra asociada al Router-Principal. Si este deja de estar disponible, el Router-Respaldo asume la IP virtual y continúa proporcionando conectividad.

El mecanismo de conmutación se implementa mediante Keepalived, que utiliza **VRRP** para gestionar el estado de los routers y determinar cuál debe actuar como equipo activo. De esta forma, el cliente mantiene siempre la misma puerta de enlace, mientras que internamente la infraestructura decide qué router debe prestar el servicio.

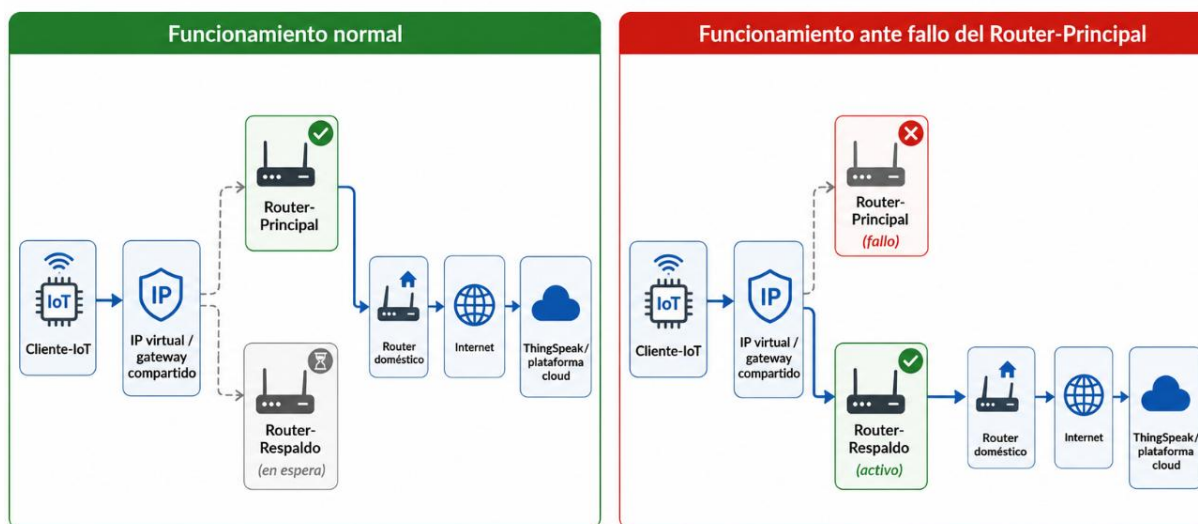


Figura 20. Esquema de funcionamiento del sistema completo en condiciones normales y en condiciones anómalas

4.2 Funcionamiento de Keepalived y VRRP en la arquitectura

Keepalived es la herramienta utilizada para implementar la alta disponibilidad entre el Router-Principal y el Router-Respaldo. Su funcionamiento se basa en el protocolo VRRP, que permite que varios routers compartan una misma dirección IP virtual.

En esta arquitectura, la IP virtual 192.168.50.250 actúa como puerta de enlace del Cliente-IoT. En condiciones normales, esta dirección está activa en el Router-Principal, que funciona como equipo MASTER y se encarga de reenviar el tráfico hacia la red externa. Mientras tanto, el Router-Respaldo permanece en estado BACKUP, preparado para asumir el servicio si detecta que el Router-Principal deja de estar disponible.

El Router-Principal envía periódicamente anuncios VRRP para indicar que sigue activo. Si el Router-Respaldo deja de recibir estos anuncios, interpreta que se ha producido un fallo y asume automáticamente la IP virtual. De esta forma, el Cliente-IoT mantiene siempre la misma puerta de enlace, aunque internamente cambie el router que presta el servicio.

4.3 Configuración de Keepalived en Router-Principal

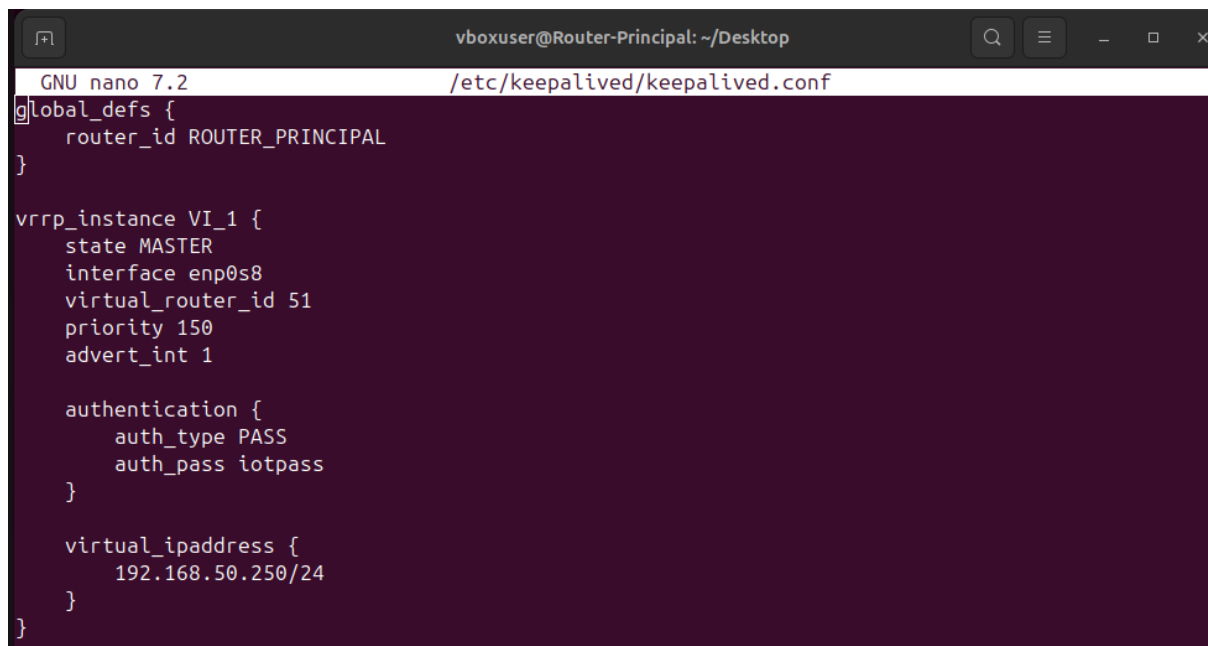
El Router-Principal se configura como el equipo activo dentro del grupo de alta disponibilidad. Para ello, se instala y configura Keepalived de forma que este router tenga prioridad sobre el Router-Respaldo y asuma la IP virtual durante el funcionamiento normal del sistema.

La configuración principal se realiza en el archivo `/etc/keepalived/keepalived.conf`. En este archivo se define una instancia VRRP asociada a la interfaz conectada a la red IoT, la interfaz que corresponde a la red interna del router, `red iotnet`. Esta instancia indica el estado inicial del equipo, la interfaz utilizada, el identificador del grupo VRRP, la prioridad del router y la IP virtual que debe asumir.

En el Router-Principal, el estado inicial se define como **MASTER**, ya que es el equipo que debe actuar como gateway activo en condiciones normales. Además, se le asigna una prioridad superior a la del Router-Respaldo. Esta diferencia de prioridad permite que, cuando ambos routers están disponibles, el Router-Principal sea el encargado de mantener la IP virtual.

La IP virtual configurada es la dirección utilizada por el Cliente-IoT como puerta de enlace. Esta dirección se añade a la interfaz interna del Router-Principal cuando Keepalived está funcionando en estado activo. De esta forma, el Cliente-IoT envía su tráfico al gateway virtual y el Router-Principal se encarga de reenviarlo hacia la red externa.

Una vez creada la configuración, se habilita y se inicia el servicio Keepalived. Posteriormente, se verifica que el servicio está activo y que la IP virtual aparece asociada a la interfaz interna del Router-Principal. Esta comprobación confirma que el Router-Principal está actuando como gateway activo dentro de la arquitectura de alta disponibilidad.

The image shows a terminal window with a dark background. At the top, the title bar reads 'vboxuser@Router-Principal: ~/Desktop'. Below it, the prompt 'GNU nano 7.2' is visible, followed by the file path '/etc/keepalived/keepalived.conf'. The terminal displays the following configuration code:

```
global_defs {
    router_id ROUTER_PRINCIPAL
}

vrrp_instance VI_1 {
    state MASTER
    interface enp0s8
    virtual_router_id 51
    priority 150
    advert_int 1

    authentication {
        auth_type PASS
        auth_pass iotpass
    }

    virtual_ipaddress {
        192.168.50.250/24
    }
}
```

Figura 21. Fichero `/etc/keepalived/keepalived.conf` del Router-Principal

Explicación del contenido de la Figura 21:

En la configuración de Keepalived de la imagen, el bloque `global_defs` define la identificación global del equipo mediante **router_id ROUTER_PRINCIPAL**, que le da un nombre único dentro del clúster de alta disponibilidad. A continuación, la instancia **vrrp_instance VI_1** agrupa todos los parámetros del protocolo VRRP para gestionar una IP virtual flotante. La línea **state MASTER** establece que este nodo actuará como el servidor principal mientras funcione correctamente, mientras que `interface enp0s8` especifica la tarjeta de red física que utilizará para comunicarse con el resto de nodos y para asignar la IP virtual. El parámetro **virtual_router_id 51** asigna un identificador numérico común a todos los equipos que compartirán la misma IP virtual, y **priority 150** determina la prioridad de este nodo frente a otros (a mayor número, más probabilidad de ser el maestro). El valor **advert_int 1** fija el intervalo de anuncios en 1 segundo, es decir, la frecuencia con la que el nodo maestro envía mensajes de "sigo vivo" al clúster. Dentro del bloque **authentication**, **auth_type PASS** indica que se usará una contraseña simple para autenticar los mensajes VRRP, y **auth_pass iotpass** es la contraseña concreta que deben compartir todos los miembros del grupo. Finalmente, el bloque **virtual_ipaddress** contiene la dirección 192.168.50.250/24, que es la IP flotante que este nodo tomará cuando sea el maestro y que pasará al nodo de respaldo en caso de fallo.

4.4 Configuración de Keepalived en Router-Respaldo

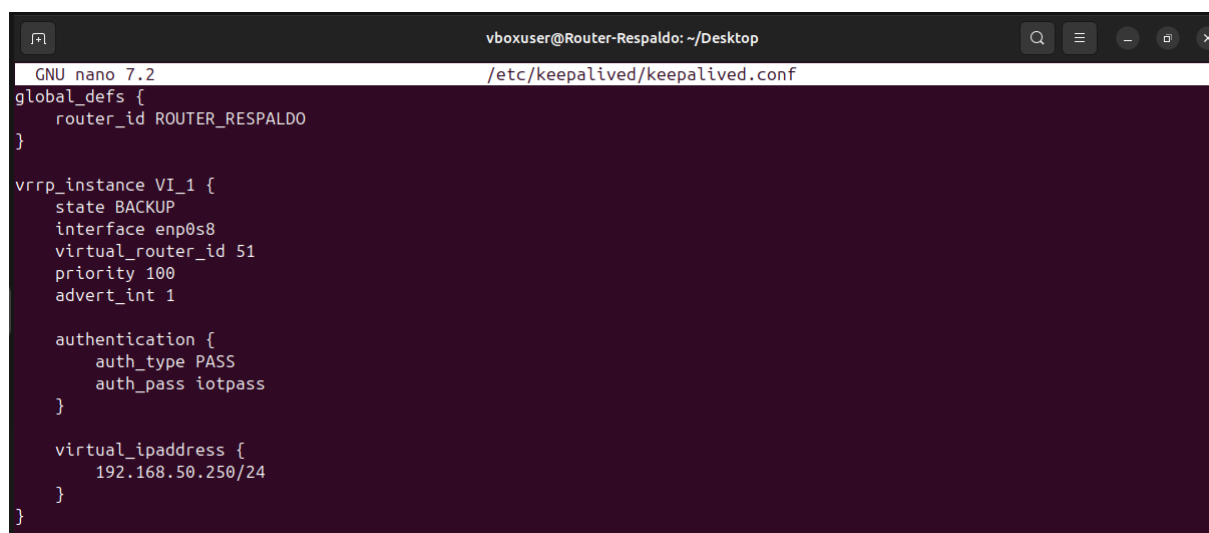
El Router-Respaldo se configura como el equipo secundario dentro de la solución de alta disponibilidad. Su función es permanecer en espera durante el funcionamiento normal del sistema y asumir la IP virtual únicamente cuando el Router-Principal deja de estar disponible. De esta forma, el Router-Respaldo actúa como mecanismo de continuidad del servicio ante fallos del gateway principal.

La configuración de Keepalived se realiza también en el archivo `/etc/keepalived/keepalived.conf`. En este caso, la instancia VRRP se asocia a la interfaz interna del Router-Respaldo, conectada a la red IoT. Esta interfaz es la que permite que el router pueda asumir la IP virtual y comunicarse con el Cliente-IoT cuando se produce una conmutación por fallo.

A diferencia del Router-Principal, el Router-Respaldo se configura con estado inicial **BACKUP** y con una prioridad inferior. Esta diferencia de prioridad permite que, cuando ambos routers están operativos, el Router-Principal conserve el papel de equipo activo. Sin embargo, si el Router-Respaldo deja de recibir los anuncios VRRP del router principal, interpreta que este ha fallado y pasa a estado activo, asumiendo la IP virtual compartida.

La IP virtual configurada debe ser la misma que en el Router-Principal, ya que ambos routers forman parte del mismo grupo de redundancia. Esta dirección es la que utiliza el Cliente-IoT como puerta de enlace. Por tanto, cuando el Router-Respaldo asume la IP virtual, el cliente puede continuar enviando tráfico hacia Internet sin modificar su configuración de red.

Tras configurar el archivo, se habilita e inicia el servicio Keepalived en el Router-Respaldo. Posteriormente, se comprueba que el servicio está en ejecución y que el router se encuentra preparado para actuar como respaldo. En condiciones normales, la IP virtual no debe aparecer asociada al Router-Respaldo mientras el Router-Principal esté activo. Esta IP solo aparecerá en el Router-Respaldo cuando se produzca un fallo o una conmutación.



```
GNU nano 7.2 /etc/keepalived/keepalived.conf
global_defs {
    router_id ROUTER_RESPALDO
}

vrrp_instance VI_1 {
    state BACKUP
    interface enp0s8
    virtual_router_id 51
    priority 100
    advert_int 1

    authentication {
        auth_type PASS
        auth_pass iotpass
    }

    virtual_ipaddress {
        192.168.50.250/24
    }
}
```

Figura 22. Fichero `/etc/keepalived/keepalived.conf` del Router-Respaldo

4.5 Definición de la IP virtual compartida

La IP virtual es uno de los elementos principales de la solución de alta disponibilidad. Su función es actuar como puerta de enlace común para el Cliente-IoT, evitando que este dependa directamente de la dirección física del Router-Principal o del Router-Respaldo. En este proyecto, la IP virtual definida es 192.168.50.250/24.

Esta dirección pertenece a la red IoT interna, por lo que puede ser utilizada por el Cliente-IoT como gateway dentro del mismo segmento de red. La IP virtual no se configura de forma permanente en la interfaz de un único router, sino que es gestionada dinámicamente por Keepalived. En funcionamiento normal, la dirección se encuentra asociada al Router-Principal. Si este deja de estar disponible, el Router-Respaldo asume automáticamente la misma dirección.

Para que el mecanismo funcione correctamente, ambos routers deben tener configurada la misma IP virtual dentro de sus respectivos archivos de Keepalived, como se ha visto en los apartados anteriores. La diferencia entre ellos no está en la IP virtual, sino en el estado y la prioridad asignada a cada router.

El Router-Principal se configura con mayor prioridad, mientras que el Router-Respaldo permanece en espera con una prioridad inferior.

Desde el punto de vista del Cliente-IoT, la puerta de enlace es siempre la misma: 192.168.50.250. Esto permite que el cliente mantenga su configuración de red aunque cambie el router que presta el servicio. Por tanto, la IP virtual actúa como un gateway lógico compartido, proporcionando continuidad de comunicación ante la caída del router principal.

La correcta definición de esta IP virtual permite desacoplar el funcionamiento del cliente respecto a los routers físicos o virtuales concretos. Gracias a ello, el sistema puede realizar la conmutación por fallo de manera transparente para el Cliente-IoT.

```
vboxuser@Router-Principal:~/Desktop$ ip -br addr show enp0s8
enp0s8          UP          192.168.50.201/24 192.168.50.250/24
```

Figura 23. Vista de la dirección 192.168.50.250 activa en el Router-Principal

```
vboxuser@Router-Respaldo:~/Desktop$ ip -br addr show enp0s8
enp0s8          UP          192.168.50.202/24
```

Figura 24. Vista de la dirección 192.168.50.250 no activa en el Router-Respaldo mientras el Router-Principal permanece activo.

4.6 Configuración del gateway del Cliente-IoT

Una vez definida la IP virtual compartida por los routers, el siguiente paso consiste en configurar dicha dirección como puerta de enlace del Cliente-IoT. La puerta de enlace configurada en el Cliente-IoT es la dirección 192.168.50.250, correspondiente a la IP virtual gestionada por Keepalived. De esta forma, todo el tráfico generado por el cliente hacia redes externas se envía inicialmente a esta dirección. Desde el punto de vista del Cliente-IoT, la configuración de red no cambia durante el proceso de conmutación. El cliente sigue enviando sus paquetes a la misma puerta de enlace, aunque internamente cambie el router que responde por dicha dirección.

```
vboxuser@Cliente-Iot:~/Desktop$ ip route
default via 192.168.50.250 dev enp0s3 proto static metric 20100
192.168.50.0/24 dev enp0s3 proto kernel scope link src 192.168.50.50 metric 100
```

Figura 25. Gateway configurado en el Cliente-IoT mediante la IP virtual

4.7 Funcionamiento normal del sistema

En condiciones normales de funcionamiento, el Router-Principal actúa como equipo activo dentro de la arquitectura de alta disponibilidad. Esto significa que mantiene asociada la IP virtual configurada como puerta de enlace del Cliente-IoT y se encarga de reenviar el tráfico desde la red IoT interna hacia la red externa.

Durante este estado, el Router-Respaldo permanece operativo, pero en modo de espera. Su función es supervisar la presencia del Router-Principal mediante Keepalived y VRRP. Mientras el Router-Principal continúa enviando anuncios de disponibilidad, el Router-Respaldo no asume la IP virtual y se mantiene preparado para intervenir únicamente si se produce un fallo.

El flujo de comunicación en funcionamiento normal comienza cuando el Cliente-IoT genera una petición hacia ThingSpeak. Como el destino se encuentra fuera de la red IoT interna, el cliente envía el tráfico a

su gateway, que corresponde a la IP virtual. En este estado, dicha IP está activa en el Router-Principal, por lo que este router recibe los paquetes, aplica el reenvío y la traducción de direcciones necesaria, y los envía hacia la red externa.

Desde la red externa, el tráfico continúa hacia el router doméstico, posteriormente a Internet y finalmente a la plataforma ThingSpeak. La recepción correcta de los datos en la plataforma permite confirmar que el sistema funciona correctamente en su estado base y que la comunicación extremo a extremo se realiza a través del Router-Principal.

En la siguiente figura, Figura 26, se puede ver el funcionamiento del sistema en condiciones normales, en las cuatro ventanas abiertas en la misma imagen se puede observar: en la primera los datos recibidos en la plataforma ThingSpeak, en la segunda el Cliente-IoT enviando los datos, en la tercera el Router-Respaldo sin la IP 192.168.50.250 activa y en la cuarta, Router-Principal con la IP 192.168.50.250 activa.

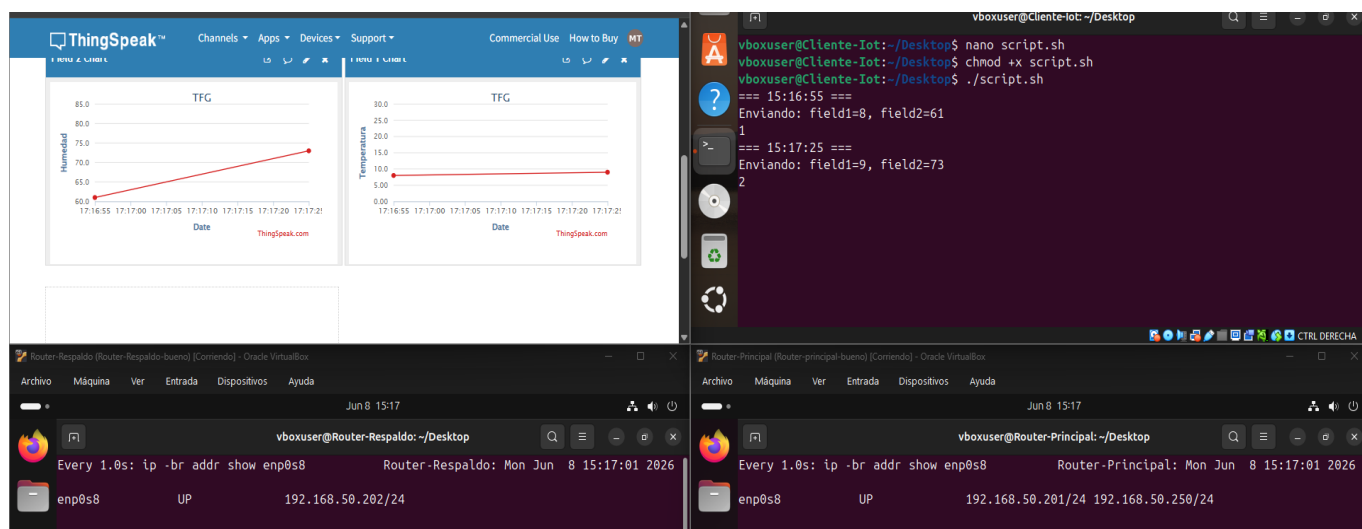


Figura 26. Vista de funcionamiento del sistema en condiciones normales (cuatro ventanas capturadas en la misma imagen)

4.8 Funcionamiento ante fallo del Router-Principal

Para comprobar el comportamiento de la arquitectura ante un fallo del Router-Principal, se simula la indisponibilidad del router activo apagando la máquina que contiene el Router-Principal. Cuando el Router-Respaldo deja de recibir los anuncios VRRP del Router-Principal, interpreta que se ha producido un fallo en el gateway activo. En ese momento, cambia su estado y asume la IP virtual compartida. De esta forma, la dirección utilizada por el Cliente-IoT como puerta de enlace pasa a estar activa en el Router-Respaldo.

Desde el punto de vista del Cliente-IoT, la puerta de enlace configurada no cambia. El cliente sigue enviando sus paquetes a la misma dirección IP virtual. Sin embargo, internamente, el router que responde por esa dirección deja de ser el Router-Principal y pasa a ser el Router-Respaldo. Durante la conmutación puede producirse una breve interrupción en la comunicación, asociada al tiempo necesario para detectar el fallo y reasignar la IP virtual. No obstante, una vez que el Router-Respaldo asume la dirección, el sistema recupera la conectividad y puede continuar enviando datos hacia ThingSpeak.

En la siguiente figura, Figura 27, se puede ver el funcionamiento del sistema en condiciones anómalas, en las tres ventanas abiertas en la misma imagen se puede observar: en la primera el Cliente-IoT enviando los datos, en la segunda los datos recibidos en la plataforma ThingSpeak, en la tercera el Router-Respaldo con la IP 192.168.50.250 activa debido a la caída de Router-Principal.

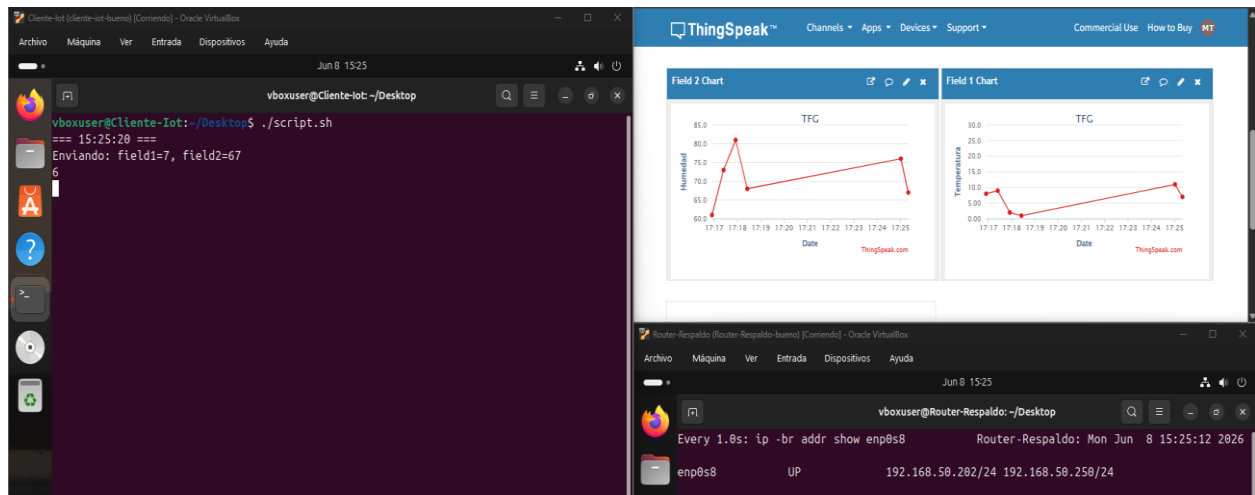


Figura 27. Vista de funcionamiento del sistema en condiciones anómalas (tres ventanas capturadas en la misma imagen)

CAPÍTULO 5

Pruebas, ataques y análisis de resultados

5.1 Metodología de pruebas

Una vez implementada la infraestructura de alta disponibilidad, se procede a validar su funcionamiento mediante un conjunto de pruebas experimentales. El objetivo de estas pruebas es comprobar si el sistema es capaz de mantener la comunicación entre el Cliente-IoT y la plataforma ThingSpeak ante distintos escenarios de fallo o degradación del Router-Principal.

Las pruebas se realizan en un entorno de laboratorio controlado, formado por el Cliente-IoT virtual, el Router-Principal, el Router-Respaldo, el router doméstico y la plataforma ThingSpeak. En este entorno se pueden provocar fallos de forma segura sin afectar a redes externas ni a servicios de terceros. La metodología consiste en partir de un estado inicial estable, en el que el Router-Principal actúa como gateway activo y el Router-Respaldo permanece en espera. A partir de este estado se ejecutan diferentes pruebas: funcionamiento sin fallo, envío de datos a ThingSpeak, parada manual del servicio Keepalived, caída de la interfaz interna del Router-Principal y simulación controlada de una situación compatible con denegación de servicio.

Esta metodología permite analizar de forma práctica la eficacia de la arquitectura redundante implementada. El objetivo no es eliminar completamente cualquier interrupción, sino reducir el impacto de la caída del router principal y demostrar que el sistema puede recuperar la comunicación de forma automática mediante el Router-Respaldo.

5.2 Prueba de conectividad y envío de datos a ThingSpeak

La primera prueba realizada consiste en verificar que el Cliente-IoT es capaz de enviar un dato individual a la plataforma ThingSpeak. Esta comprobación permite validar la comunicación extremo a extremo antes de realizar una comunicación permanente y continua entre emisor y receptor.

Para ello, desde el Cliente-IoT se ejecuta una petición HTTP mediante la herramienta curl. En dicha petición se incluye la clave de escritura del canal de ThingSpeak y un valor de prueba asociado a los campos configurados. El objetivo es comprobar que el dato atraviesa correctamente la red IoT interna, el router activo, la red doméstica e Internet hasta llegar a la plataforma cloud.

Cuando ThingSpeak recibe correctamente la petición, devuelve un número mayor que cero. Este valor corresponde al identificador de la entrada almacenada en el canal y permite confirmar que la actualización ha sido aceptada. Por tanto, la respuesta numérica obtenida en la terminal constituye una evidencia directa de que el envío del dato se ha realizado correctamente.

Esta prueba demuestra que la conectividad inicial del sistema es correcta y que la infraestructura implementada permite transmitir información desde el Cliente-IoT hasta la plataforma de monitorización. Una vez validado este envío puntual, se puede utilizar el sistema como base para analizar el comportamiento ante fallos del Router-Principal.

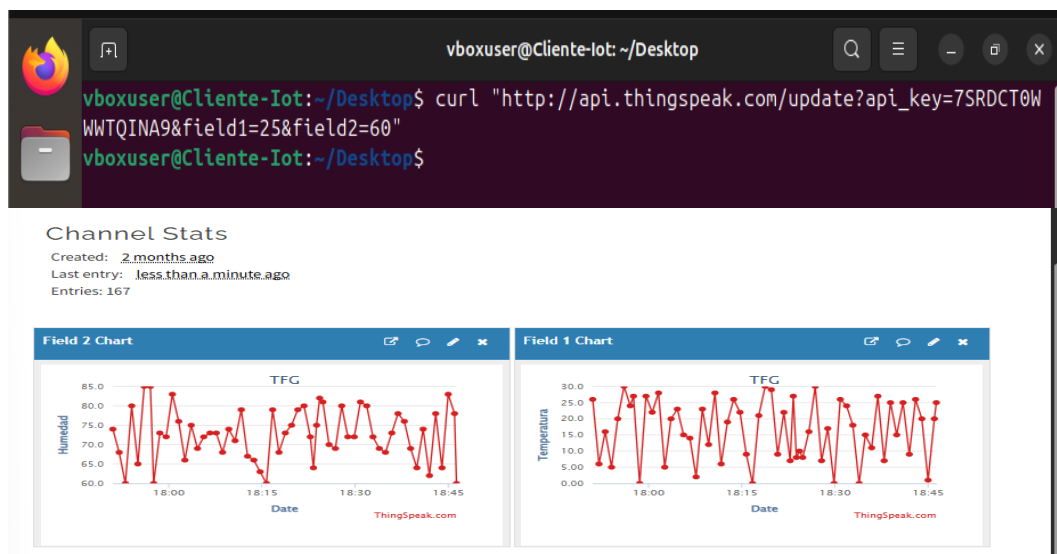


Figura 28. Envío puntual de datos desde el Cliente-IoT hacia ThingSpeak

5.3 Escenario base: funcionamiento sin fallo

Antes de realizar las pruebas de fallo y degradación, se establece un escenario base en el que toda la infraestructura funciona correctamente. Este escenario sirve como referencia para comparar el comportamiento del sistema antes y después de provocar una incidencia sobre el Router-Principal.

En este estado, el Router-Principal actúa como gateway activo de la red IoT interna y mantiene asociada la IP virtual compartida. El Router-Respaldo permanece en espera.

Este escenario ya ha sido descrito y verificado en el apartado 4.7, donde se comprobó que el sistema funciona correctamente en condiciones normales. En el presente capítulo se utiliza como punto de partida para analizar el impacto de los fallos posteriores y comprobar si la arquitectura de alta disponibilidad permite recuperar la comunicación.

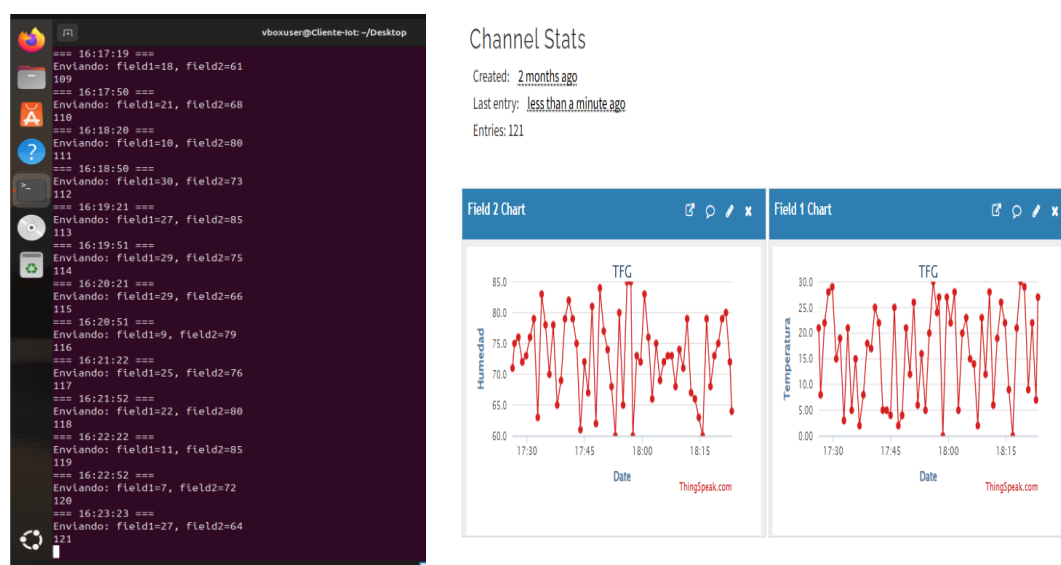


Figura 29. Vista de funcionamiento del sistema en escenario base.

5.4 Escenario de caída manual del Router-Principal

En esta prueba se simula una caída manual del Router-Principal con el objetivo de comprobar si el sistema de alta disponibilidad es capaz de reaccionar ante la pérdida del gateway activo. Este escenario representa una situación en la que el router principal deja de estar disponible y, por tanto, ya no puede mantener activa la IP virtual utilizada por el Cliente-IoT como puerta de enlace.

Para realizar la prueba, se parte del escenario base descrito anteriormente, en el que el Router-Principal actúa como equipo activo y el Router-Respaldo permanece en espera. A continuación, se provoca la caída del Router-Principal apagando la máquina virtual correspondiente. Al producirse esta caída, el Router-Respaldo deja de recibir los anuncios VRRP enviados por el router activo.

Tras detectar la ausencia del Router-Principal, el Router-Respaldo cambia su estado y asume la IP virtual 192.168.50.250 tras un breve intervalo de detección. Aunque durante el proceso de conmutación puede producirse una interrupción breve, el sistema recupera la conectividad una vez que el Router-Respaldo toma el control de la IP virtual, lo que supone un lapso de tiempo muy pequeño según se ha podido observar, aproximadamente 2 segundos, que es el tiempo que tarda en darse cuenta que Router-Principal ha caído, ya que ambos están configurados para mandar mensajes de confirmación de correcto funcionamiento cada segundo. La comprobación de este escenario se realiza observando que el Router-Principal deja de estar disponible, que el Router-Respaldo pasa a tener activa la IP virtual y que el Cliente-IoT puede seguir enviando datos hacia ThingSpeak.

De esta forma, se demuestra que la arquitectura implementada permite mantener la comunicación ante la caída manual del router principal. Todo esto puede apreciarse en la Figura 27 del apartado 4.8 donde se ha llevado a cabo esta demostración de fallo de Router-Principal y que copio a continuación.

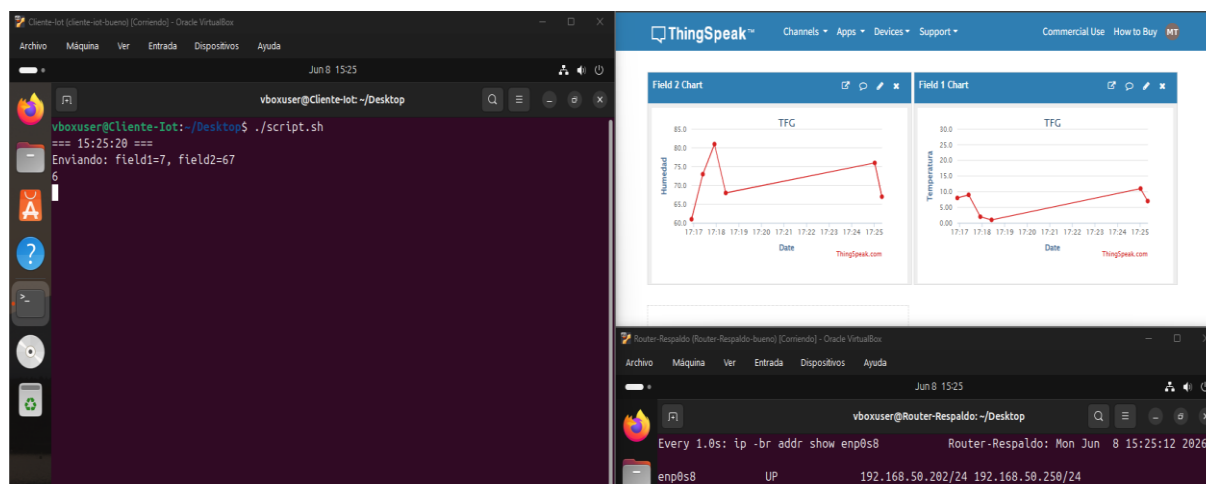


Figura 30. Réplica de la Figura 27 donde se observa al sistema funcionando con Router-Principal apagado manualmente.

5.5 Escenario de fallo de interfaz del Router-Principal

En esta prueba se simula un fallo parcial del Router-Principal mediante la desactivación de su interfaz interna conectada a la red IoT. A diferencia del escenario anterior, en este caso la máquina virtual del

Router-Principal no se apaga por completo, sino que permanece encendida, pero pierde la conectividad con la red interna donde se encuentra el Cliente-IoT.

Este escenario permite representar una situación en la que el router principal sigue operativo a nivel de sistema, pero no puede prestar servicio como gateway para la red IoT debido a la caída de una de sus interfaces. En concreto, se desactiva la interfaz interna del Router-Principal, que es la encargada de comunicarse con el Cliente-IoT y de mantener activa la IP virtual dentro de la red 192.168.50.0/24.

Antes de provocar el fallo, el Router-Principal actúa como equipo activo y mantiene asociada la IP virtual 192.168.50.250. El Router-Respaldo permanece en espera, sin tener activa dicha dirección. A continuación, se desactiva la interfaz interna del Router-Principal mediante un comando de red, provocando que este deje de estar disponible para el Cliente-IoT.

Al producirse la caída de la interfaz, el Router-Respaldo deja de recibir correctamente los anuncios VRRP del Router-Principal y asume la IP virtual compartida. De esta forma, el gateway utilizado por el Cliente-IoT pasa a estar activo en el Router-Respaldo, manteniendo la misma dirección IP configurada como puerta de enlace.

Durante la prueba puede producirse una interrupción, en este caso ha sido una interrupción de aproximadamente 2 segundos, en la comunicación, correspondiente al tiempo necesario para detectar el fallo y realizar la conmutación. Sin embargo, una vez que el Router-Respaldo asume la IP virtual, el Cliente-IoT puede continuar enviando datos hacia ThingSpeak sin modificar su configuración de red.

Esta prueba demuestra que la arquitectura implementada no solo responde ante la caída completa del Router-Principal, sino también ante fallos parciales que afectan a su conectividad con la red IoT.

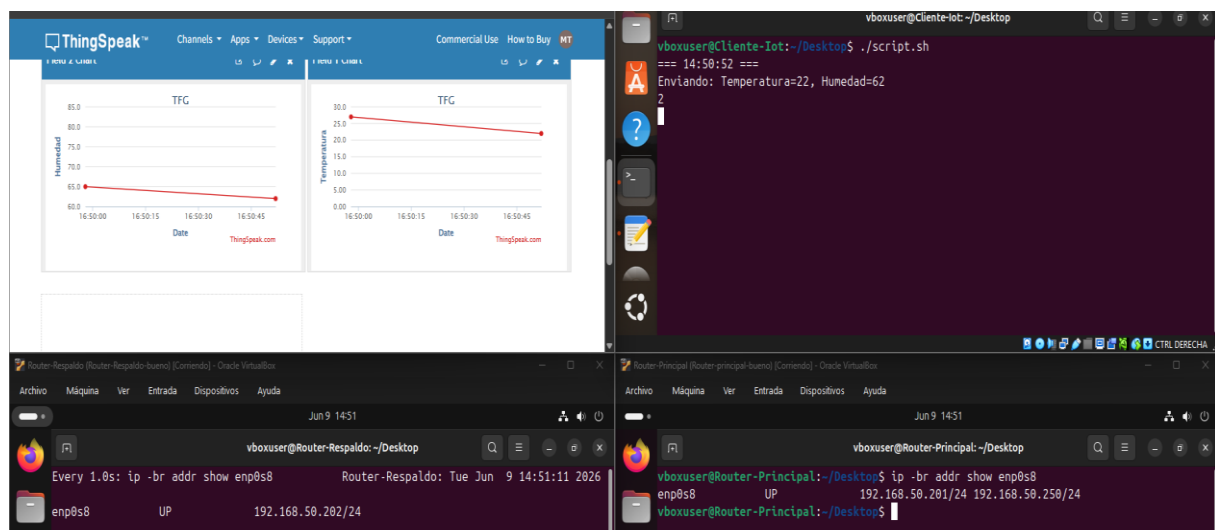


Figura 31. Funcionamiento normal del sistema antes de fallo en la interfaz

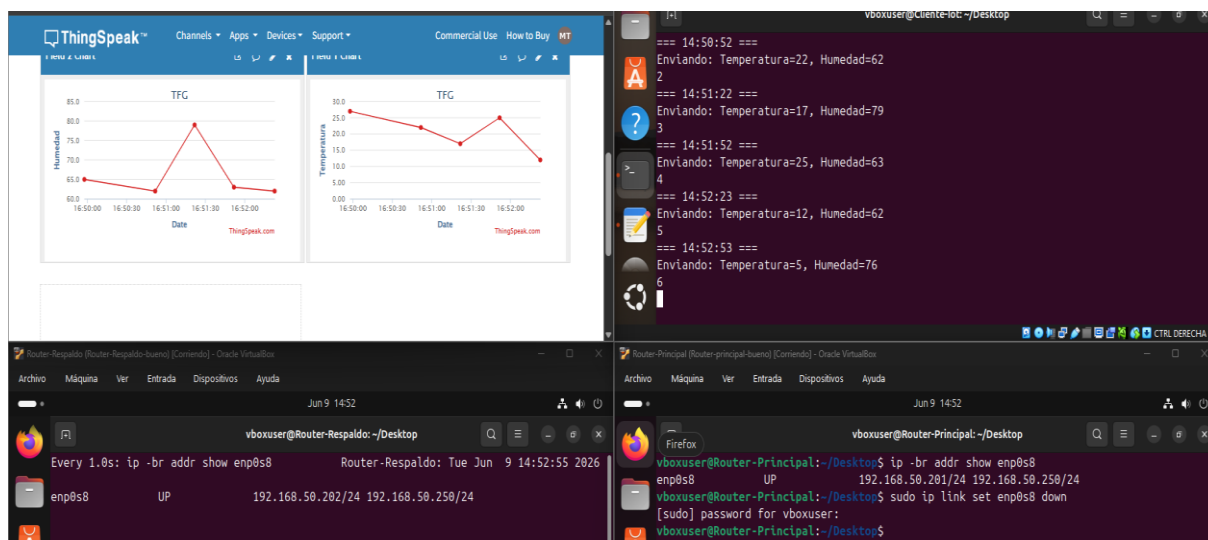


Figura 32. Funcionamiento del sistema después de fallo en la interfaz de red de Router-Principal

Una vez se acaba la prueba, se vuelve a habilitar la interfaz y Router-Principal vuelve a asumir automáticamente la IP virtual utilizada como gateway. Cliente-IoT vuelve a transmitir a través de Router-Principal en lugar de Router-Respaldo.

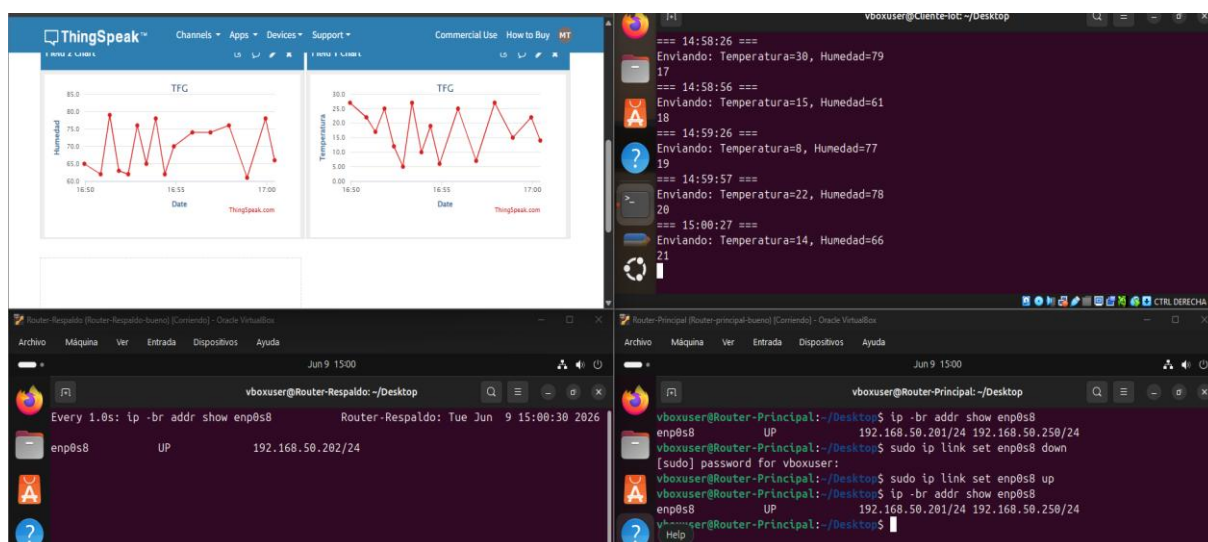


Figura 33. Recuperación de la interfaz caída

5.6 Escenario de denegación de servicio en entorno controlado

En esta prueba se reproduce una situación compatible con un escenario de denegación de servicio en un entorno controlado. El objetivo no es realizar un ataque real contra una red externa, sino generar una degradación sobre el Router-Principal para analizar cómo afecta al funcionamiento del sistema IoT y a la continuidad del envío de datos hacia ThingSpeak.

Para simular esta situación, se genera carga de procesamiento sobre el Router-Principal mediante la herramienta stress-ng. Esta herramienta permite someter al sistema a carga de CPU, memoria y entrada/salida durante un intervalo de tiempo determinado. En este caso, se ejecuta una prueba de estrés

de dos minutos sobre el Router-Principal, con el fin de observar si la degradación de recursos afecta al envío de datos o provoca una conmutación hacia el Router-Respaldo.

Antes de iniciar la prueba, el sistema se encuentra en el escenario base: el Router-Principal mantiene activa la IP virtual 192.168.50.250, el Router-Respaldo permanece en espera y el Cliente-IoT envía datos hacia ThingSpeak a través del gateway virtual. A continuación, se inicia la carga controlada sobre el Router-Principal mientras se monitoriza el comportamiento del Cliente-IoT, del Router-Respaldo y de la plataforma ThingSpeak.

Durante la prueba, el Router-Principal fue sometido a carga simultánea de CPU, entrada/salida y memoria. La ejecución finalizó correctamente después de aproximadamente dos minutos, lo que indica que el router fue degradado de forma controlada pero no llegó a quedar completamente inoperativo. Mientras tanto, el Cliente-IoT continuó enviando datos hacia ThingSpeak y la plataforma siguió aceptando las actualizaciones generadas.

Además, durante la ejecución de la prueba se comprobó que el Router-Respaldo no asumió la IP virtual 192.168.50.250. En su interfaz interna únicamente permanecía configurada su dirección propia, 192.168.50.202, lo que indica que Keepalived siguió considerando al Router-Principal como equipo activo. Por tanto, la carga generada no provocó una conmutación automática hacia el router alternativo. Este resultado permite diferenciar entre una caída completa del Router-Principal y una situación de degradación parcial. En los escenarios anteriores, la pérdida completa del router o de su interfaz interna provocaba la conmutación hacia el Router-Respaldo. Sin embargo, en este caso el Router-Principal siguió funcionando lo suficiente como para mantener los anuncios VRRP y continuar prestando servicio, aunque estuviera sometido a carga.

La prueba demuestra que la arquitectura implementada responde correctamente ante fallos claros del gateway principal, pero no necesariamente ante degradaciones parciales si el router continúa operativo.

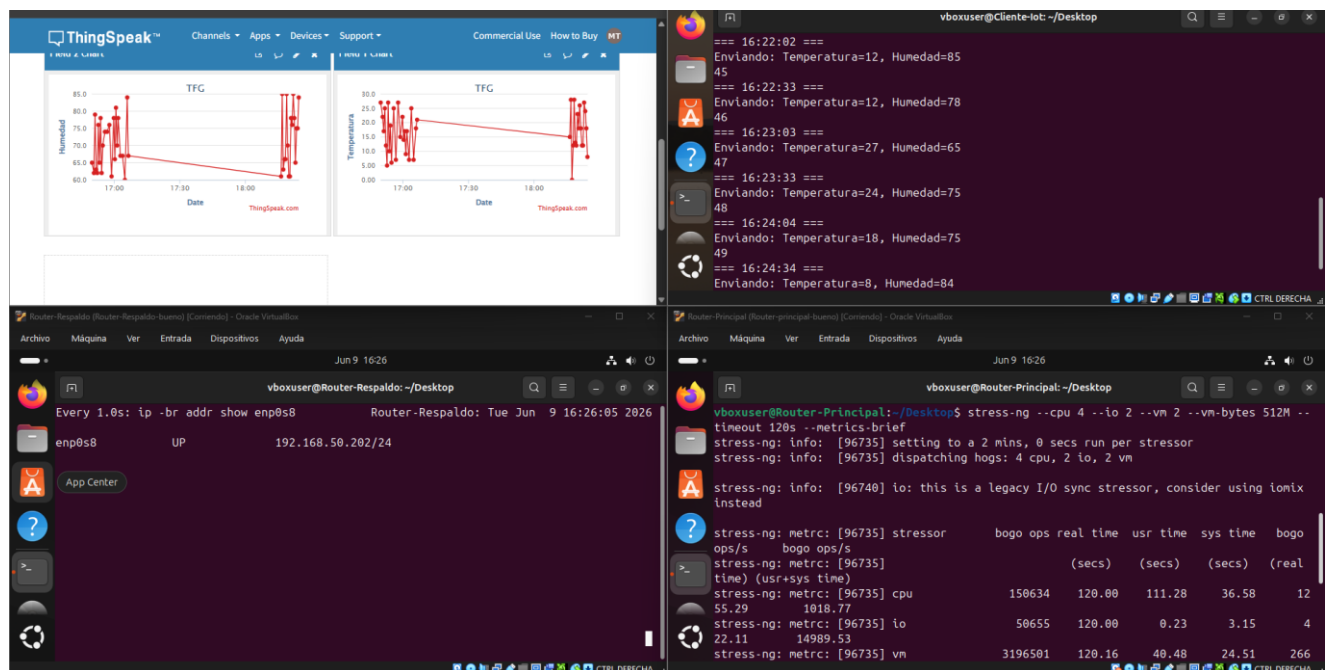


Figura 34. Sistema ante stress-ng

5.7 Escenario de denegación de servicio externo

Con el objetivo de aproximar el entorno de pruebas a una situación más realista, se incorporó al laboratorio una nueva máquina virtual denominada Atacante-DoS. La finalidad de esta máquina fue

simular el comportamiento de un equipo malicioso dentro de la red IoT, capaz de generar tráfico anómalo contra el sistema. De esta forma, la prueba no se limita a fallos manuales provocados sobre los routers, sino que introduce un origen externo de tráfico malicioso.

5.7.1 Ataque con hping3 en modo flood

En esta prueba se empleó la máquina virtual Atacante-DoS para generar tráfico ICMP intensivo contra el Router-Principal. Esta máquina genera tráfico anómalo dirigido al gateway principal del sistema, con el fin de comprobar si dicha sobrecarga puede afectar al funcionamiento del router activo y provocar una conmutación hacia el Router-Respaldo.

Para realizar la prueba se utilizó la herramienta hping3 en modo flood, generando paquetes ICMP de forma intensiva hacia el Router-Principal. Antes de iniciar la prueba, el sistema se encontraba en condiciones normales de funcionamiento: el Router-Principal actuaba como equipo MASTER, mantenía activa la IP virtual 192.168.50.250 y el Router-Respaldo permanecía en estado BACKUP.

Durante la ejecución de la prueba se monitorizó simultáneamente el estado del Router-Principal, del Router-Respaldo y del Cliente-IoT. La máquina Atacante-DoS comenzó a enviar tráfico ICMP intensivo hacia el Router-Principal, mientras el Cliente-IoT continuaba intentando enviar datos hacia ThingSpeak utilizando como gateway la IP virtual 192.168.50.250.

A pesar del tráfico generado, el Router-Principal no llegó a quedar inoperativo. El servicio Keepalived continuó funcionando correctamente y siguió enviando anuncios VRRP hacia el Router-Respaldo. Como consecuencia, el Router-Respaldo no interpretó la situación como una caída del router activo y no asumió la IP virtual. Por tanto, durante esta prueba no se produjo conmutación entre routers.

El resultado obtenido permite comprobar una limitación importante de la configuración básica de alta disponibilidad implementada. Keepalived y VRRP detectan correctamente situaciones de caída completa del Router-Principal o pérdida de comunicación en la interfaz interna, pero no necesariamente detectan una situación de degradación parcial si el router continúa operativo y sigue enviando anuncios VRRP.

Por tanto, el ataque con hping3 en modo flood no provocó la caída del Router-Principal ni el cambio automático de la IP virtual al Router-Respaldo. Sin embargo, la prueba resulta relevante porque permite demostrar que una generación elevada de tráfico no siempre es suficiente para activar el mecanismo de conmutación.

Esta prueba muestra que la arquitectura implementada responde adecuadamente ante fallos claros del router activo, pero presenta limitaciones frente a escenarios de sobrecarga o degradación parcial. Este resultado sirve como base para justificar mejoras posteriores orientadas a detectar de forma más precisa el estado operativo del Router-Principal.

A continuación se muestra una imagen del sistema funcionando correctamente a pesar de la sobrecarga de trabajo a la que estaba sometido el Router-Principal debido al ataque:

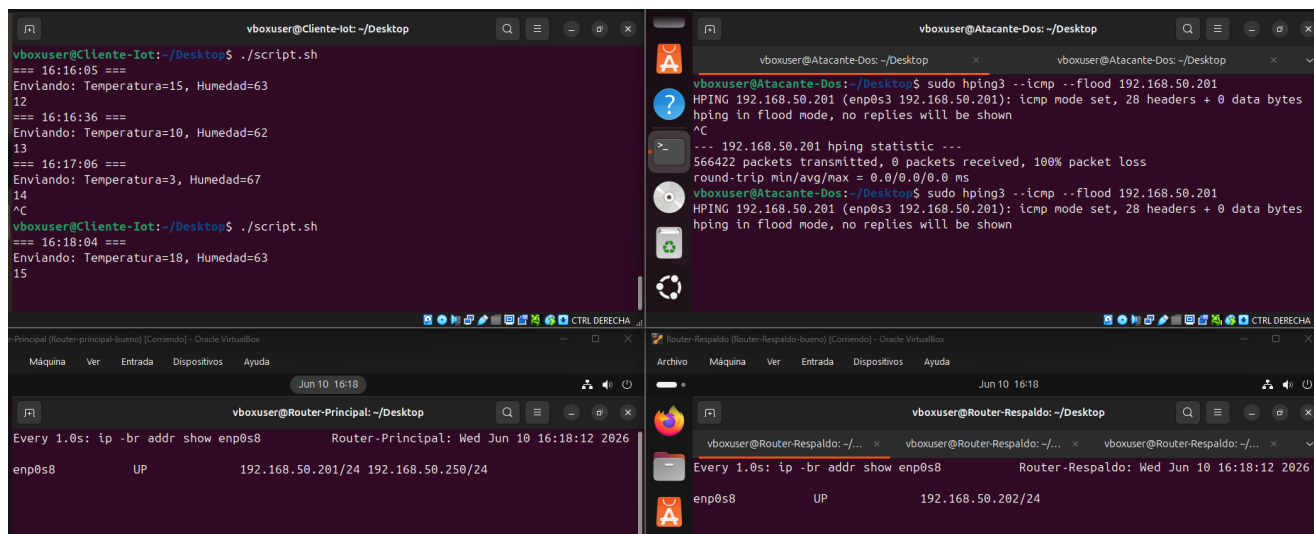


Figura 35. Sistema hping3 y flood

La Figura 36 representa de forma esquemática el escenario de ataque con hping3. Este esquema permite identificar el origen del tráfico anómalo, el equipo afectado por la sobrecarga y el flujo normal de comunicación del sistema IoT.

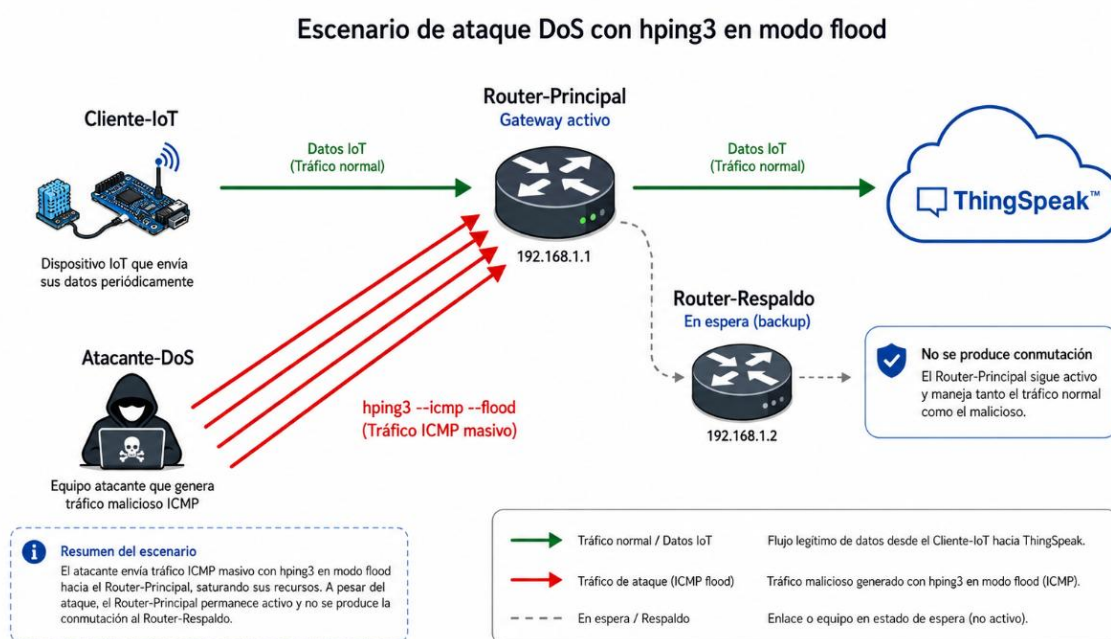


Figura 36. Esquema del ataque ICMP flood generado con hping3 contra el Router-Principal.

5.7.2 Ataque Man-in-the-Middle y filtrado VRRP Unicast

Tras comprobar que la generación intensiva de tráfico ICMP mediante hping3 en modo flood no provocaba la caída del Router-Principal ni la conmutación hacia el Router-Respaldo, se planteó un segundo escenario de prueba más específico. El objetivo de este nuevo escenario fue analizar el

comportamiento de la arquitectura de alta disponibilidad cuando se interrumpe la comunicación de control entre los dos routers, sin apagar físicamente el Router-Principal ni desactivar manualmente ninguna de sus interfaces.

Para ello, se modificó la configuración de Keepalived para que los mensajes VRRP dejaran de enviarse mediante multicast y pasaran a transmitirse en modo unicast entre el Router-Principal y el Router-Respaldo. En la configuración inicial, VRRP utilizaba la dirección multicast 224.0.0.18 para distribuir los anuncios de disponibilidad. Este comportamiento dificultaba la interceptación del tráfico en el entorno virtualizado, ya que los mensajes multicast no se comportaban como una comunicación directa entre dos direcciones IP concretas. Al configurar VRRP en modo unicast, los anuncios pasaron a enviarse directamente desde la dirección interna del Router-Principal, 192.168.50.201, hacia la dirección interna del Router-Respaldo, 192.168.50.202, y viceversa.

Una vez aplicado este cambio, se ejecutó una prueba de interceptación desde la máquina Atacante-DoS, situada en la red interna iotnet con dirección 192.168.50.60. La finalidad de esta máquina fue simular un nodo malicioso o comprometido dentro de la red IoT capaz de interferir en la comunicación entre los routers. La prueba consistió en generar una situación de Man-in-the-Middle, en la que el tráfico de control entre el Router-Principal y el Router-Respaldo pasaba por la máquina atacante. Sobre ese tráfico interceptado se aplicó un filtrado específico para descartar los paquetes correspondientes al protocolo VRRP.

Durante la ejecución de la prueba, el Router-Principal continuó encendido y operativo. Sin embargo, los anuncios VRRP que debía recibir el Router-Respaldo dejaron de llegar correctamente debido al filtrado aplicado en la máquina atacante. Como consecuencia, el Router-Respaldo interpretó que el Router-Principal había dejado de estar disponible y cambió su estado interno a MASTER, asumiendo la IP virtual 192.168.50.250.

El resultado obtenido fue diferente al de las pruebas anteriores. En este caso sí se consiguió que el Router-Respaldo asumiera la IP virtual, pero no se produjo una conmutación limpia. Al no haberse apagado realmente el Router-Principal, este continuó manteniendo también la IP virtual en su interfaz interna. Por tanto, la dirección 192.168.50.250 quedó activa simultáneamente en ambos routers.

Este comportamiento corresponde a un escenario conocido como Split-Brain o cerebro dividido. En una arquitectura de alta disponibilidad, se produce una situación de Split-Brain cuando dos nodos creen de forma simultánea que deben actuar como equipo activo. En este caso, el Router-Principal mantenía la IP virtual porque seguía funcionando correctamente desde su propio punto de vista, mientras que el Router-Respaldo también asumía la misma IP al dejar de recibir los mensajes VRRP del principal.

La Figura 37 muestra el estado del sistema durante la prueba. En ella se observa la máquina Atacante-DoS ejecutando la interceptación de tráfico, el Cliente-IoT realizando envíos hacia ThingSpeak y ambos routers con la IP virtual 192.168.50.250 activa al mismo tiempo. Esta situación evidencia que la interrupción de los mensajes VRRP puede provocar una conmutación aparente, pero también puede generar un conflicto de IP si el router principal no ha caído realmente.

Desde el punto de vista del Cliente-IoT, este escenario genera un comportamiento indeterminado. El cliente sigue utilizando la dirección 192.168.50.250 como puerta de enlace, pero al existir dos routers respondiendo por la misma IP, la comunicación depende de la resolución ARP existente en cada momento. El Cliente-IoT enviará su tráfico al router cuya dirección MAC quede asociada en su tabla ARP a la IP virtual. Por tanto, el tráfico puede salir por el Router-Principal o por el Router-Respaldo dependiendo de qué respuesta ARP sea registrada por el cliente y la comunicación con la plataforma no se interrumpe.

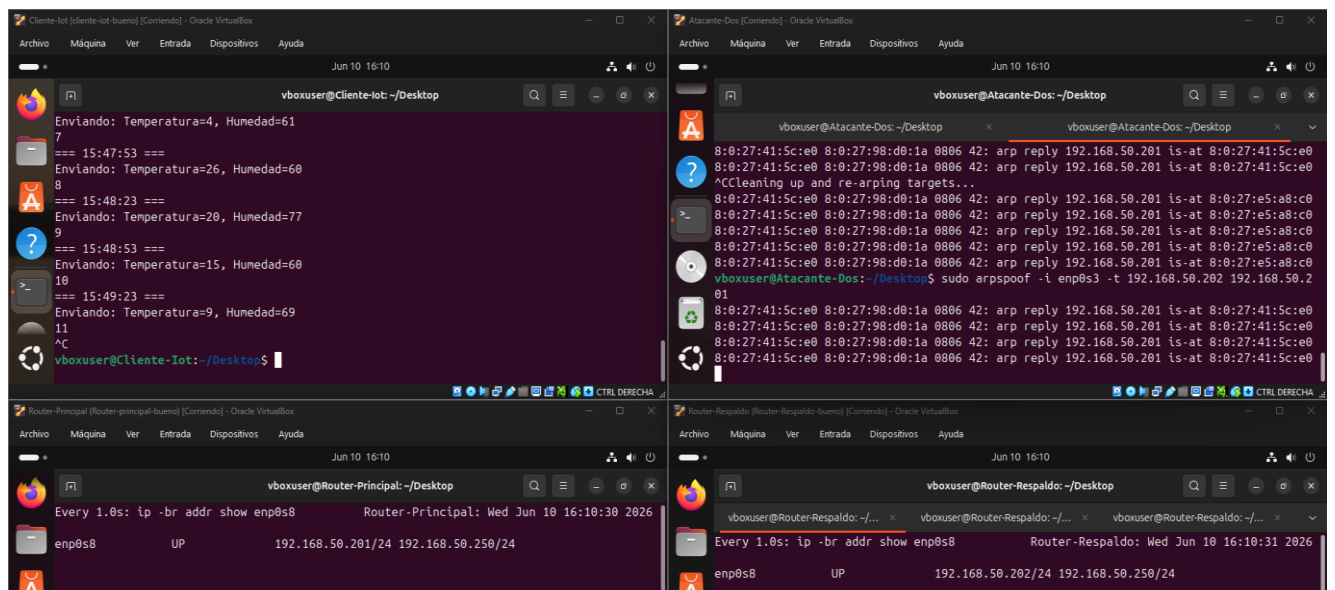


Figura 37. Sistema ante man-in-the-middle y filtrado VRRP Unicast

La Figura 38 muestra el esquema del ataque Man-in-the-Middle aplicado sobre la comunicación VRRP Unicast entre el Router-Principal y el Router-Respaldo.

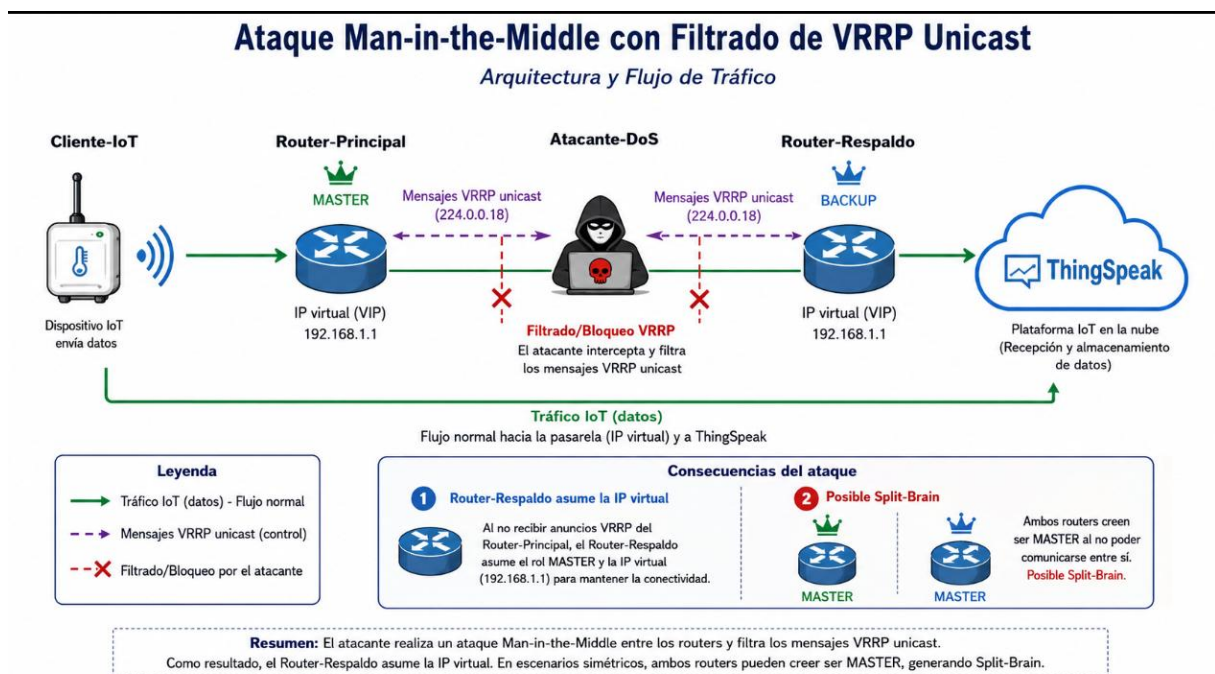


Figura 38. Esquema del ataque Man-in-the-Middle con filtrado de mensajes VRRP Unicast.

5.8 Recuperación del Router-Principal

Una vez finalizadas las pruebas de fallo, degradación y generación de tráfico, se analizó el proceso de recuperación del Router-Principal. El objetivo de esta fase fue comprobar si el sistema podía volver a su estado normal de funcionamiento después de que el Router-Principal recuperara su disponibilidad o después de detener las condiciones anómalas introducidas durante las pruebas.

En los escenarios de caída manual y fallo de interfaz interna, la recuperación del Router-Principal se produjo de forma ordenada. Tras volver a encender la máquina virtual o reactivar la interfaz de red afectada, el Router-Principal recuperó la conectividad con la red IoT interna y volvió a participar en el grupo de alta disponibilidad gestionado por Keepalived. Al tener una prioridad superior a la del Router-Respaldo, el Router-Principal volvió a asumir la IP virtual 192.168.50.250, recuperando su función como gateway activo del sistema.

En el caso de la prueba con stress-ng y del ataque con hping3 en modo flood, no fue necesario realizar una recuperación de la IP virtual, ya que el Router-Principal no llegó a perder el estado MASTER. En ambos casos, el router activo continuó operativo y siguió enviando anuncios VRRP hacia el Router-Respaldo. Por tanto, la IP virtual permaneció asociada al Router-Principal durante toda la prueba.

El escenario más complejo se produjo durante la prueba Man-in-the-Middle con filtrado de VRRP Unicast. En este caso, el Router-Respaldo asumió la IP virtual al dejar de recibir los anuncios VRRP del Router-Principal. Sin embargo, como el Router-Principal no se había apagado realmente, ambos routers llegaron a mantener activa la dirección 192.168.50.250 de forma simultánea, generando una situación de Split-Brain. Al detener la prueba, la recuperación no fue inmediata, ya que las tablas ARP podían conservar temporalmente asociaciones alteradas. Como consecuencia, fue necesario esperar a que las tablas de vecindad se actualizaran o limpiar manualmente la información ARP para que ambos routers recuperaran una visión coherente del estado de la red, en este caso, el sistema tardó unos diez segundos en recuperar su funcionamiento normal una vez se detuvo la prueba aunque esto no afectó al correcto funcionamiento del sistema ya que la plataforma no dejó de recibir los datos en ningún momento.

5.9 Pérdida de paquetes, tiempo de conmutación y continuidad del servicio

El análisis de las pruebas realizadas permite evaluar el comportamiento del sistema en términos de pérdida de comunicación, tiempo de conmutación y continuidad del envío de datos hacia ThingSpeak. En el escenario base, el sistema funcionó de forma estable, con el Router-Principal actuando como gateway activo y el Cliente-IoT enviando datos correctamente a través de la IP virtual 192.168.50.250.

En la prueba de caída manual del Router-Principal, el Router-Respaldo asumió la IP virtual de forma automática tras dejar de recibir los anuncios VRRP. La interrupción observada fue muy breve, aproximadamente de dos segundos, y el Cliente-IoT pudo continuar enviando datos sin modificar su configuración de red.

En el escenario de fallo de interfaz interna del Router-Principal, la conmutación también se realizó correctamente. En este caso, la interrupción fue de aproximadamente dos segundos, tras los cuales el Router-Respaldo asumió la IP virtual y el sistema recuperó la conectividad.

En las pruebas de degradación mediante stress-ng y generación de tráfico con hping3 en modo flood, no se produjo conmutación. El Router-Principal continuó operativo y siguió enviando anuncios VRRP, por lo que el Router-Respaldo permaneció en estado de espera. Aun así, el Cliente-IoT mantuvo la comunicación con ThingSpeak.

El escenario Man-in-the-Middle con filtrado VRRP Unicast provocó que el Router-Respaldo asumiera la IP virtual, pero al no producirse una caída real del Router-Principal se generó una situación de Split-Brain, con la IP virtual activa simultáneamente en ambos routers. En este caso, la continuidad del servicio quedó condicionada por la resolución ARP del Cliente-IoT, pero no se perdió en ningún momento ni se interrumpió el servicio.

En conjunto, las pruebas demuestran que la arquitectura responde correctamente ante fallos claros del Router-Principal, con tiempos de conmutación reducidos y con pocas pérdidas de paquetes.

CAPÍTULO 6

Conclusiones y líneas futuras

6.1 Conclusiones generales

El desarrollo de este Trabajo de Fin de Grado ha permitido diseñar, implementar y evaluar una arquitectura IoT orientada a mejorar la disponibilidad del servicio frente a fallos del gateway principal y escenarios de degradación de la red. A partir de un planteamiento inicial basado en un nodo físico ESP-01 con sensor DHT11, el proyecto evolucionó hacia una arquitectura virtualizada que permitió estudiar con mayor precisión el comportamiento de la infraestructura de comunicación y los mecanismos de alta disponibilidad implementados.

Una de las principales conclusiones del trabajo es que la disponibilidad en sistemas IoT no depende únicamente del correcto funcionamiento del sensor o del dispositivo encargado de generar los datos. Aunque el nodo IoT pueda medir y procesar información correctamente, el sistema pierde utilidad si dichos datos no llegan a la plataforma cloud encargada de almacenarlos y visualizarlos. Por este motivo, el gateway de salida se convierte en un elemento crítico dentro de la arquitectura, ya que su caída puede interrumpir completamente la comunicación entre el cliente IoT y la plataforma remota.

La solución implementada, basada en dos routers virtuales y una IP virtual gestionada mediante Keepalived y VRRP, ha demostrado ser eficaz ante fallos claros del Router-Principal. En las pruebas de caída manual del router activo y fallo de su interfaz interna, el Router-Respaldo asumió automáticamente la IP virtual utilizada como puerta de enlace por el Cliente-IoT. Gracias a ello, el cliente pudo mantener la misma configuración de red y recuperar la comunicación con ThingSpeak sin intervención manual. Estos resultados confirman que el uso de una IP virtual compartida permite desacoplar al cliente de un router físico o virtual concreto, facilitando la continuidad del servicio ante fallos del gateway principal.

También se ha comprobado que la arquitectura proporciona tiempos de conmutación reducidos. En las pruebas realizadas, la interrupción observada fue breve, aproximadamente de dos o tres segundos según el escenario, lo que permitió que el envío de datos hacia la plataforma se mantuviera de forma general. Esto demuestra que una solución de alta disponibilidad basada en VRRP puede ser adecuada para mejorar la resiliencia de sistemas IoT sencillos, especialmente cuando el problema principal es la caída completa del router activo o la pérdida de conectividad en su interfaz interna.

Sin embargo, el trabajo también ha permitido identificar limitaciones importantes. Las pruebas de degradación mediante stress-ng y generación intensiva de tráfico ICMP mediante hping3 mostraron que el Router-Principal podía seguir siendo considerado activo por Keepalived mientras continuara enviando anuncios VRRP. En estos casos, aunque el sistema estuviera sometido a carga o tráfico anómalo, no se producía una conmutación hacia el Router-Respaldo. Esta observación evidencia que una configuración básica de VRRP detecta correctamente fallos claros, pero no necesariamente situaciones de degradación parcial del servicio.

El escenario Man-in-the-Middle con filtrado de VRRP Unicast permitió analizar una situación más compleja. En esta prueba, el Router-Respaldo dejó de recibir los anuncios VRRP del Router-Principal y asumió la IP virtual. No obstante, como el Router-Principal no había caído realmente, ambos routers llegaron a mantener activa la misma dirección IP virtual, generando una situación de Split-Brain. Este resultado es especialmente relevante porque demuestra que no toda conmutación implica necesariamente una mejora de la disponibilidad. Si la comunicación de control entre routers se interrumpe sin que el router activo deje de funcionar, el sistema puede entrar en un estado inconsistente que provoque conflictos de IP y un comportamiento dependiente de la resolución ARP del cliente.

En conjunto, el proyecto demuestra que los mecanismos de alta disponibilidad son una herramienta útil para mejorar la continuidad de servicio en arquitecturas IoT, pero también que deben diseñarse con una visión más amplia que la simple detección de caída del router activo. Para entornos reales, sería necesario complementar Keepalived y VRRP con mecanismos adicionales de monitorización, validación del estado real del servicio, protección frente a manipulación del tráfico de control y medidas que eviten escenarios de Split-Brain.

Por tanto, se puede concluir que la arquitectura desarrollada cumple con el propósito principal del trabajo: mejorar la resiliencia de un sistema IoT frente a fallos del gateway y analizar su comportamiento ante escenarios de degradación y ataque controlado. El sistema implementado ha permitido mantener la comunicación en los fallos principales estudiados, identificar los límites de una solución básica de alta disponibilidad y extraer conclusiones útiles para el diseño de infraestructuras IoT más robustas.

6.2 Cumplimiento de los objetivos

El objetivo principal de este Trabajo de Fin de Grado era diseñar e implementar un sistema IoT con mecanismos de alta disponibilidad que permitiera mantener la continuidad del servicio ante fallos del gateway principal y escenarios de degradación o ataque controlado. A partir de los resultados obtenidos, se puede afirmar que este objetivo se ha cumplido, ya que se ha construido una arquitectura funcional capaz de enviar datos hacia ThingSpeak y de mantener la comunicación mediante un router de respaldo cuando el router principal deja de estar disponible.

En relación con el primer objetivo específico, se planteó inicialmente el uso de un nodo IoT físico basado en un módulo ESP-01 y un sensor DHT11 para adquirir datos ambientales y enviarlos mediante WiFi a una plataforma cloud. Este objetivo se abordó durante la primera fase del proyecto, en la que se validó el funcionamiento del sistema físico y su capacidad para enviar datos a ThingSpeak. Sin embargo, debido a las limitaciones técnicas encontradas al intentar integrar dicho nodo físico detrás de los routers virtuales, se decidió sustituirlo por un Cliente-IoT virtual. Esta adaptación permitió mantener la lógica principal del sistema IoT, basada en la generación y envío periódico de datos, y facilitó el estudio de la disponibilidad de la infraestructura de red.

El segundo objetivo específico consistía en implementar una infraestructura de red virtualizada compuesta por dos routers, permitiendo la conmutación automática ante fallos. Este objetivo se ha cumplido mediante la creación de dos routers virtuales Ubuntu, denominados Router-Principal y Router-Respaldo, configurados con interfaces internas y externas, direccionamiento IP estático, reenvío IPv4, reglas de NAT mediante nftables y persistencia de la configuración. Además, mediante Keepalived y VRRP se ha implementado una IP virtual compartida que permite al Cliente-IoT utilizar siempre la misma puerta de enlace, independientemente del router que esté prestando el servicio en cada momento.

El tercer objetivo específico era modelar y reproducir en un entorno controlado escenarios de degradación del router principal compatibles con ataques. Este objetivo también se ha cumplido, ya que se han diseñado y ejecutado diferentes pruebas experimentales sobre la arquitectura implementada. Entre ellas se incluyen la caída manual del Router-Principal, el fallo de su interfaz interna, la generación de carga mediante stress-ng, el envío intensivo de tráfico ICMP mediante hping3 y un escenario Man-in-the-Middle con filtrado de mensajes VRRP en modo unicast.

El cuarto objetivo específico consistía en diseñar mecanismos de detección de degradación basados en métricas de red y disponibilidad del servicio. Este objetivo se ha abordado parcialmente. La arquitectura implementada permite detectar correctamente fallos claros del router principal mediante la ausencia de anuncios VRRP, lo que activa la conmutación hacia el router de respaldo. Además, se han observado

métricas como la continuidad del envío de datos, el estado de la IP virtual, la conectividad del Cliente-IoT y la recepción de datos en ThingSpeak. No obstante, las pruebas de degradación parcial han demostrado que una configuración básica de Keepalived y VRRP no detecta por sí sola todos los escenarios de deterioro del servicio, especialmente si el router principal sigue enviando anuncios VRRP. Por tanto, este objetivo se considera cumplido desde el punto de vista experimental y de análisis, aunque se identifica como una línea clara de mejora futura.

El quinto objetivo específico consistía en implementar y evaluar el proceso de failover automático hacia el router de respaldo, analizando métricas como el tiempo de conmutación, la pérdida de mensajes, el impacto en la latencia y la disponibilidad global del sistema. Este objetivo se ha cumplido mediante las pruebas de caída manual y fallo de interfaz del router principal. En ambos casos, el router de respaldo asumió automáticamente la IP virtual y permitió recuperar la comunicación con ThingSpeak sin modificar la configuración del Cliente-IoT. Los tiempos de conmutación observados fueron reducidos, aproximadamente de dos o tres segundos según el escenario, y la continuidad del servicio se mantuvo de forma general.

Finalmente, el sexto objetivo específico era analizar la arquitectura propuesta como mecanismo de resiliencia frente a ataques de denegación de servicio en entornos IoT. Este objetivo se ha cumplido mediante el análisis de los escenarios de stress-ng, hping3 y Man-in-the-Middle. Los resultados muestran que la arquitectura mejora la resiliencia frente a fallos claros del gateway principal, pero también presenta limitaciones ante situaciones de degradación parcial o interferencia en la comunicación de control entre routers. En particular, el escenario de Split-Brain permitió comprobar que la interrupción de los mensajes VRRP puede provocar estados inconsistentes si el router principal continúa operativo.

En conjunto, los objetivos planteados al inicio del trabajo se han cumplido de forma satisfactoria. La arquitectura desarrollada ha permitido validar el funcionamiento de un sistema IoT con alta disponibilidad, comprobar la conmutación automática entre routers, analizar su comportamiento ante distintos escenarios de fallo y ataque controlado, e identificar las limitaciones que deberían abordarse en futuras mejoras del sistema.

6.3 Líneas de trabajo futuro

Como líneas de trabajo futuro, una primera mejora consistiría en complementar la configuración básica de Keepalived y VRRP con mecanismos adicionales de monitorización. En este trabajo se ha comprobado que el sistema responde correctamente ante fallos claros del router principal, pero no necesariamente ante degradaciones parciales si el router continúa enviando anuncios VRRP. Por ello, sería recomendable incorporar scripts de comprobación que verificaran no solo el estado del router, sino también la conectividad real hacia Internet, la carga del sistema y la capacidad de enviar datos correctamente a la plataforma cloud.

Otra posible mejora sería reforzar la seguridad de la comunicación entre los routers. El escenario Man-in-the-Middle con filtrado de VRRP Unicast demostró que la interrupción de los mensajes de control puede provocar una situación de Split-Brain. Para reducir este riesgo, podrían estudiarse mecanismos alternativos que mejoren la comunicación entre ambos routers.

También sería interesante trasladar la arquitectura a un entorno físico completo, integrando el nodo ESP-01 con sensor DHT11 detrás de routers físicos o virtuales conectados mediante interfaces de red

adicionales, adquiriendo el hardware necesario para ello. Esto permitiría validar el sistema en condiciones más cercanas a un despliegue real.

REFERENCIAS Y BIBLIOGRAFÍA

- [1] ITU-T, "Recommendation ITU-T Y.4000/Y.2060: Overview of the Internet of Things," International Telecommunication Union, 2012. https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-Y.2060-201206-!!!PDF-S&type=items
- [2] IoT Analytics, "State of IoT 2024: Number of connected IoT devices growing 13% to 18.8 billion globally," 2024. <https://iot-analytics.com/number-of-connected-iot-devices-2024/>
- [3] ENISA, "Baseline Security Recommendations for IoT in the context of Critical Information Infrastructures," European Union Agency for Cybersecurity, 2017. https://www.enisa.europa.eu/sites/default/files/publications/WP2017%20O-1-1-2%201%20Baseline%20Security%20Recommendations%20for%20IoT%20in%20the%20context%20of%20CII_FINAL.pdf
- [4] MathWorks, "ThingSpeak Documentation," MathWorks. <https://thingspeak.mathworks.com/>
- [5] J. Voas, "Networks of 'Things'," National Institute of Standards and Technology, <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-183.pdf>
- [6] NIST, "An Introduction to Information Security," <https://csrc.nist.gov/pubs/sp/800/12/r1/final>
- [7] NIST, "Developing Cyber-Resilient Systems: A Systems Security Engineering Approach," NIST Special Publication 800-160, Vol. 2 Rev. 1, National Institute of Standards and Technology, 2021.: <https://csrc.nist.gov/pubs/sp/800/160/v2/r1/final>
- [8] S. Nadas, "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6," RFC 5798, <https://www.rfc-editor.org/rfc/rfc5798>
- [9] Sensor DHT11. <https://programarfacil.com/blog/arduino-blog/sensor-dht11-temperatura-humedad-arduino/>
- [10] ESP-01. <https://naylorpmechatronics.com/espressif-esp/48-modulo-esp-01-esp8266-wifi-serial.html>
- [11] Instrucciones Linux 1. <https://medium.com/@samiashafique/mastering-network-diagnostics-a-simple-guide-to-curl-traceroute-and-ping-8c8f315ebf67>
- [12] Instrucciones Linux 2. <https://www.digitalocean.com/community/tutorials/linux-commands>
- [13] Oracle VM VirtualBox User Manual. <https://www.virtualbox.org/manual/>
- [14] Comunicación sistemas IoT. <https://es.digi.com/blog/post/how-do-iot-devices-communicate>
- [15] Protocolo TCP/IP. <https://www.fortinet.com/lat/resources/cyberglossary/tcp-ip>
- [16] Direcciones Ips y protocolos de comunicación. <https://vitriko.eu/tipos-direcciones-ip/>
- [17] Direccionamiento IP y Gateway. <https://keepcoding.io/blog/que-es-gateway-ip-o-puerta-de-enlace-y-usos/>
- [18] NAT y masquerading. <https://www.ibm.com/docs/es/i/7.5.0?topic=translation-masquerade-hide-nat>

- [19] VRRP. https://www.cisco.com/c/es_mx/support/docs/security/vpn-3000-series-concentrators/7210-vrrp.html
- [20] IPs virtuales. <https://www.ibm.com/docs/es/aix/7.2.0?topic=protocol-virtual-ip-address>
- [21] Keepalived: concepto general. <https://mytcpip.com/linux-ha-keepalived-en-5-minutos/>
- [22] Instalación y configuración de Keepalived. <https://www.youtube.com/watch?v=dGDOPGYT-SE>
- [23] Alta disponibilidad en Ubuntu. <https://www.youtube.com/watch?v=mVWzwbAXhmo>
- [24] Apagar y encender interfaz de conexión: <https://bbs.archlinux.org/viewtopic.php?id=197268>
- [25] Instrucción stress-ng. https://www.reddit.com/r/techsupport/comments/1jmrzq7/how_to_stress_test_the_ram_with_stressng/?tl=es-es