Universidad de Salamanca
Departamento de Matemática Aplicada

# Diseño e implementación de un modelo individual para la simulación de la propagación de malware en redes de sensores inalámbricas

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Tesis para optar al grado de
*Doctora por la Universidad de Salamanca*

**Farrah Kristel Batista Guerra**

Director: Dr. Ángel Martín del Rey

Co-directora: Dra. Araceli Queiruga Dios

Septiembre 2020

Dedico este trabajo a:
mi bebé Fariely Sarai,
mi esposo Santiago,
mis padres Rafael e Itzel
y mi hermana Tiffani

# Declaración de autoría

Dña. Farrah Kristel Batista Guerra, presenta la tesis doctoral titulada "Diseño e implementación de un Modelo Individual para la simulación de la Propagación de Malware en Redes de Sensores Inalámbricas", para optar al Grado de Doctora por la Universidad de Salamanca; y declara que este proyecto ha sido realizado bajo la dirección del Dr. Ángel Martín del Rey, profesor Titular de Universidad, y la co-dirección de la Dra. Araceli Queiruga Dios, profesora Titular de Universidad, del Departamento de Matemática Aplicada, Instituto Universitario de Física Fundamental y Matemáticas (IUFFyM), de la Universidad de Salamanca.

En Salamanca, a 7 de septiembre de 2020

La doctoranda

Farrah Kristel Batista Guerra

Firmado digitalmente por Farrah Kristel Batista Guerra
Fecha: 2020.09.07 23:25:18 +02'00'

Director

Dr. Ángel Martín del Rey

MARTIN DEL REY ANGEL MARIA - 07953200F

Firmado digitalmente por MARTIN DEL REY ANGEL MARIA - 07953200F
Fecha: 2020.09.07 14:34:18 +02'00'

Co-directora

Dra. Araceli Queiruga Dios

Firmado digitalmente por Araceli
Fecha: 2020.09.07 19:19:37 +02'00'

# Agradecimientos

Mis primeras palabras son para agradecer a Dios por ser mi guía en el sendero que me ha llevado a la culminación de este proyecto; y por la fortaleza para estar lejos de mis padres y hermana, durante estos años.

A mi bebé Fariely, gracias por escogerme como tu madre, y enseñarme que todos los días hay un motivo para sonreír.

A mi madre Itzel, gracias por tus consejos, valores y ser un modelo para mí ¡Gracias por darme la vida!

A mi padre Rafael, ejemplo de perseverancia, un hombre dedicado a su familia, por su trabajo para que nunca nos faltase nada ¡Gracias infinitas!

A mi hermana Tiffani, gracias por siempre haber estado allí cuando lo he necesitado.

A mi esposo Santiago, gracias por emprender esta ardua aventura conmigo, por tu apoyo y motivación en los momentos difíciles.

A mi director de tesis Ángel, gracias por recibirnos al momento de nuestra llegada a la ciudad de Salamanca; por los consejos personales y académicos; y por el tiempo dedicado durante todos estos años en las exhaustivas correcciones para el desarrollo de este proyecto.

A mi co-directora de tesis Araceli, gracias por los consejos y tu orientación durante estos años, que me han ayudado a aumentar mis conocimientos; por el tiempo dedicado a las exhaustivas revisiones de este manuscrito.

A Sofía, Inés y Yen gracias por todos los consejos y momentos compartidos, en especial por acompañarnos en el nacimiento de mi bebé. ¡Nunca lo olvidaré!

A la Universidad de Salamanca, agradezco todos los recursos académicos que han puesto a mi disposición para la ejecución de este proyecto.

Agradezco al Instituto de Tecnologías Físicas y de la Información (ITEFI) del Consejo Superior de Investigaciones Científicas (CSIC), por los recursos ofrecidos para realizar la estancia de investigación.

Finalmente, agradezco a la Secretaría Nacional de Ciencia, Tecnología e Innovación (SENACYT), y al Instituto para la Formación y Aprovechamiento de Recursos Humanos (IFARHU), de la República de Panamá, por la beca otorgada para realizar este estudio doctoral.

# Resumen

Las redes de sensores inalámbricas están formadas por un conjunto de dispositivos denominados sensores, que han sido desplegados en un área determinada; además, forman una red sin arquitectura pre-establecida de tipo ad-hoc, es decir, son redes descentralizadas que no requieren una infraestructura preexistente, como los dispositivos de red para el enrutamiento o puntos de acceso inalámbricos. En los últimos años, el malware se ha convertido en una potencial amenaza para las vulnerabilidades del Internet de las Cosas; por lo tanto, estas amenazas en constante evolución afectan a las redes de sensores inalámbricas.

La propagación del malware en redes de sensores inalámbricas se ha estudiado desde diferentes perspectivas, con la finalidad de conocer cómo se producen estos ataques y poder definir medidas de seguridad especializadas. Estos estudios se realizan a través de modelos matemáticos, utilizando diferentes herramientas de modelado, como los sistemas de ecuaciones diferenciales ordinarias y sistemas de ecuaciones en derivadas parciales, cadenas de Markov, autómatas celulares o agentes. Sin embargo, la mayoría de estos modelos propuestos excluyen las características individuales de los componentes principales de la red.

El objetivo de esta tesis doctoral es definir un modelo individual basado en agentes y desarrollar un entorno computacional que permita la simulación y análisis de la propagación de diferentes tipos de malware en redes de sensores inalámbricas.

La metodología que se ha utilizado en este trabajo comienza con una revisión del estado del arte de los temas principales, que incluyen redes de sensores inalámbricas, malware y modelos basados en agentes. Posteriormente, se ha realizado una revisión bibliográfica de los modelos matemáticos que se han propuesto para la simulación del malware en redes de sensores inalámbricas.

A continuación se han extraído las características más significativas de las redes de sensores inalámbricas, lo que permitirá crear un modelo matemático bajo el paradigma de modelos basados en agentes, donde se han detallado los agentes involucrados, los coeficientes que les afectan y las reglas de transición que utilizan. Por último, se ha implementado computacionalmente el modelo, utilizando el entorno de trabajo Mesa, desarrollado en Python, que ha permitido analizar los resultados en diferentes escenarios y para diferentes topologías.

Finalmente, se ha concluido que tanto los entornos como las topologías influyen en el proceso de propagación del malware. Además, las características computacionales de los sensores pueden ayudar a evitar una rápida propagación, puesto que puede hacer sensores con altas características computacionales que dispongan de algún tipo de mecanismo de seguridad ya implementado.

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# Introducción

Una red de sensores inalámbrica (WSN, *wireless sensor network*) está formada por un conjunto de microsensores que se encuentran espacialmente distribuidos en posiciones no determinadas, con una infraestructura de comunicaciones inalámbricas que ha sido diseñada para monitorizar, en la región donde ha sido desplegada, algún fenómeno físico de interés, en áreas como defensa, vigilancia, salud, o medio ambiente; es decir, una WSN reúne datos de variables como temperatura, humedad, sonido, vibración, presión o contaminantes. La información recogida por cada sensor es enviada desde la puerta de enlace hacia un servidor principal, llamado estación base, que es el que se encarga de procesarla. La tecnología de las WSN es uno de los fundamentos del Internet de las Cosas (IoT) y la Industria 4.0 [1].

Usualmente, las WSN proporcionan soluciones de rastreo y monitorización a gran escala y bajo coste, debido al bajo consumo de energía de los sensores [2, 3]. Sin embargo, estos dispositivos tienen un radio de alcance corto y un área de cobertura pequeña, por lo que es necesario un gran número de sensores [4].

Las WSN pueden gestionar información sensible y operar en entornos hostiles y sin vigilancia; por ello, es necesario considerar medidas de seguridad sólidas en el diseño de la red, que impidan que pueda verse afectada por algún tipo de software malicioso. Sin embargo, algunas características de estas redes como las limitaciones computacionales de los sensores, el espacio limitado para el almacenamiento de datos, la fuente de energía que utilizan y el canal de comunicación poco fiable, son obstáculos importantes para la aplicación de técnicas de ciberseguridad [5]. Estas restricciones son la razón por la que el estudio de la propagación de malware en una WSN ha generado un interés creciente en los últimos años. Además, los dispositivos inteligentes, como los teléfonos móviles de última generación, desempeñan un papel importante en la gestión de las WSN, por lo que se han desarrollado nuevas técnicas para la detección de software malicioso [6].

La epidemiología matemática consiste en el estudio de la propagación de enfermedades contagiosas utilizando herramientas matemáticas que permiten predecir su comportamiento y desarrollar estrategias para controlarla [7]. La epidemiología matemática se ha considerado la base de muchos de los estudios que se encuentran en la literatura sobre de la propagación del malware en diferentes tipos de redes informáticas, incluidas las redes de sensores inalámbricas.

Para estudiar la propagación de malware en WSN se han desarrollado varios modelos matemáticos basados en el modelo epidemiológico de Kermack y McKendrick, el cual utiliza los sistemas ecuaciones diferenciales ordinarias [8]. El modelo desarrollado por Kermack y McKendrick es un modelo global en el que no se tiene en cuenta el comportamiento individual. En los modelos globales la topología de conexión entre individuos o dispositivos se modeliza utilizando un grafo completo (es decir, se supone que todos los nodos están en contacto con todos los nodos de la red considerada, en todo momento).

El modelo epidemiológico más sencillo es el clásico modelo SI [9], que solo considera los estados susceptible ($S$) e infectado ($I$), o el modelo SIRD, que considera, además de los estados susceptible e infectado, el recuperado ($R$) y el dañado ($D$) [10]; en estos modelos se tiene en cuenta la topología, pero no consideran las características particulares del resto de componentes de la red, es decir son modelos globales. Por el contrario, los modelos basados en individuos tienen en cuenta las conexiones y características individuales de los nodos de una red. Para definir modelos individuales se utilizan herramientas basadas en autómatas celulares o en agentes. Algunos ejemplos de modelos individuales son los modelos modificados de propagación epidemiológicos propuestos por Nwokoye, SEIR-V, [11] y el SIRD de Wang et al. [12], en los que han sido considerados otros estados como el expuesto ($E$), vacunado ($V$) y muerto ($D$).

Los modelos basados en agentes (MBA) son aquellos en los que los individuos (denominados agentes o actores), se consideran entidades únicas y autónomas, que interactúan entre sí y con su entorno local. Un MBA se compone de un entorno donde unos agentes evolucionan según unas reglas [13]; donde los diferentes tipos de agentes representan a los individuos dentro del sistema simulado. Comúnmente, este paradigma se utiliza para sistemas con agentes heterogéneos, autónomos y proactivos, en los que no se pueden ignorar las características individuales, por lo que resultan adecuados para ser utilizados en el estudio de la propagación de malware en una red de sensores inalámbrica. Estos modelos se han utilizado a menudo en disciplinas como la salud [13], ciencias sociales [14], arqueología [15], economía [16] o en ecología [17].

En los últimos años, se han modelizado problemas epidemiológicos utilizando los MBA, debido a las importantes ventajas de este modelo en comparación con la modelización

matemática tradicional [18, 19]. Entre estas ventajas se pueden destacar las interacciones agente-agente y agente-entorno. Además, las reglas de transición pueden ajustarse con parámetros reales de acuerdo al fenómeno de interés. Otra ventaja de este modelo, es la posibilidad de analizar y monitorizar el comportamiento de un nodo específico de la red o de la red en general.

Sin embargo, una desventaja de los modelos MBA es que las simulaciones en redes a gran escala requieren más recursos computacionales que la simulación de una red pequeña. De esta manera, el software de simulación puede limitar el número de nodos de la red. Además, hay que tener en cuenta las fases de verificación y validación del modelo, lo que puede hacer que el proceso sea más largo.

En algunos de los modelos basados en individuos propuestos por otros autores, se han considerado como actores las características de la red, los diferentes tipos de nodos, el malware y, en algunos casos recientes, la topología [20]. El modelo basado en agentes SEIRS-D, propuesto en este estudio, tiene en cuenta otras características esenciales de estos actores; además, incluye nuevos actores que aún no han sido considerados en otros modelos, y que intervienen activamente en la propagación del malware.

Consecuentemente, la propuesta de este modelo basado en agentes analiza la propagación de malware en las redes de sensores inalámbricas, identificando las características individuales de los agentes implicados, así como las interacciones agente-agente y agente-entorno; asimismo, se ha utilizado la epidemiología matemática para determinar los estados compartimentales de los agentes en cada período de tiempo.

## 1.1. Hipótesis

Las redes de sensores inalámbricas son vulnerables a ataques maliciosos, por lo tanto, el malware puede propagarse por toda la red causando pérdida o robo de información, así como el mal funcionamiento de la red; esto se debe a los bajos recursos computacionales que poseen los nodos de la red para disponer de medidas de seguridad. El comportamiento del malware en estas redes puede ser simulado a través de la implementación computacional de modelos matemáticos.

Los modelos matemáticos propuestos para simular la propagación del malware en redes de sensores inalámbricas suelen ser modelos globales, es decir que incluyen características generales de la red, o modelos individuales que incluyen algunas de las características básicas de los componentes de la red.

Las características individuales de los sensores y la topología de red pueden mejorar significativamente el análisis de la propagación del malware. Para esto, es posible diseñar

un modelo de naturaleza individual basado en agentes, que incluya estas características individuales y considere la topología de la red, así como otros elementos que puedan influir en el comportamiento del malware.

## 1.2. Objetivos

El objetivo general de esta tesis doctoral es desarrollar un entorno computacional que permita la simulación y el análisis de la propagación de diferentes tipos de malware en redes de sensores inalámbricas.

Los objetivos específicos que han sido establecidos para alcanzar este objetivo general son:

1. Realizar una revisión bibliográfica de los modelos matemáticos propuestos para la simulación de malware en redes de sensores inalámbricas.

2. Definir un modelo matemático basado en agentes para estudiar el comportamiento del malware en redes de sensores inalámbricas.

3. Desarrollar un entorno de simulación computacional basado en el modelo matemático diseñado.

## 1.3. Metodología

Los modelos basados en agentes han sido utilizados para simular la propagación de diferentes enfermedades de transmisión biología, entre ellas el dengue, la chikunguña o la malaria [13]. En este sentido, se ha hecho una analogía entre virus biológicos y virus informáticos, debido a su forma de propagación. Por lo tanto, los modelos basados en agentes pueden resultar útiles para estudiar la propagación del malware en redes de sensores inalámbricas.

La realización de este estudio se ha dividido en dos fases claramente diferenciadas: el estudio teórico del modelo basado en agentes y su implementación práctica en diferentes escenarios.

La fase del estudio teórico se inició con la revisión de los conceptos generales y de seguridad de las redes de sensores inalámbricas, las definiciones y formas de propagación de los distintos tipos de malware, y los fundamentos teóricos de los modelos basados en agentes.

Posteriormente, se llevó a cabo una revisión bibliográfica de los modelos matemáticos existentes para explicar la propagación del malware en redes de sensores inalámbricas, con el fin de detectar aquellas características adecuadas para ser incluidas en el nuevo modelo.

También se han estudiado escenarios de redes de sensores inalámbricas potencialmente reales, para determinar nuevas características que pueden influir directamente en el comportamiento del malware. Finalmente, se ha diseñado un modelo teórico teniendo en cuenta los conocimientos y experiencias adquiridos previamente.

Durante la fase de implementación práctica, se ha seleccionado una herramienta especializada para la simulación de modelos basados en agentes que se adaptase al modelo propuesto. Consecuentemente, se ha realizado la simulación de la propagación de malware con el nuevo modelo y se han analizado los resultados en diferentes escenarios.

## 1.4. Resumen de los contenidos

Se presenta a continuación un resumen de los contenidos de cada uno de los capítulos que forman esta memoria:

**Capítulo 2:** Las redes de sensores inalámbricas son redes informáticas susceptibles a sufrir ataques con diferentes tipos malware. El comportamiento del malware puede ser estudiado utilizando diferentes técnicas, como la modelización; los modelos basados en agentes pueden ser útiles para este tipo de análisis. En este capítulo, se detallan los conceptos generales y de seguridad de las redes de sensores inalámbricas, la definición y tipos de malware, así como los fundamentos de los modelos basados en agentes.

**Capítulo 3:** Los modelos basados en agentes permiten el estudio de las interacciones de los individuos, tanto con otros individuos como con el entorno donde se desenvuelven. Esto permite que las características individuales de los agentes sean uno de los factores más importantes para la creación de estos modelos. En este capítulo, se describe el modelo basado en agentes propuesto, SEIRS-D, incluyendo la definición de los agentes, los coeficientes y las reglas de transición establecidas.

**Capítulo 4:** La simulación permite ejecutar en tiempo real un modelo matemático, definiendo variables y parámetros, que permite obtener resultados que puedan ser analizados, en la mayoría de las ocasiones, de forma visual. En este capítulo, se realiza la simulación del modelo SEIRS-D en diferentes escenarios, donde las variables definidas en cada uno de esos escenario tienen un papel significativo en los resultados alcanzados.

**Capítulo 5:** Los resultados de las simulaciones permiten validar la precisión del modelo; para esto, los diferentes programas de simulación generan resultados visuales en forma de gráficas, para que sean fácilmente comprensibles. En este capítulo, se analizan los resultados obtenidos de la simulación en los diferentes escenarios, con la finalidad de determinar qué factores influyen en el comportamiento del malware.

**Capitulo 6:** La creación, simulación y desarrollo del modelo basado en agentes, SEIRS-D, permite el análisis de los diferentes comportamientos del malware en una red de sensores inalámbrica, en diferentes escenarios. En este caso, las topologías, el entorno y las características de los sensores influyen en la propagación del malware. En este capítulo, se incluyen las conclusiones del estudio realizado, además de las limitaciones que han surgido durante el desarrollo del proyecto. Por último, se presentan futuros trabajos.

# Capítulo 2

# Marco Teórico

Las redes de sensores inalámbricas pueden sufrir ataques ocasionados por diferentes tipos de software malicioso. Generalmente, estas redes cuentan con medidas de seguridad que resultan vulnerables a los ataques. Para encontrar la forma de prevenir ataques malintencionados se han realizado estudios sobre la propagación del malware en estas redes, utilizando modelos matemáticos, como los modeles individuales basados en autómatas celulares o modelos basados en agentes.

En este capítulo, se establece el marco teórico en el que se enmarca esta memoria. Se definen las características de las redes de sensores inalámbricas, su arquitectura y los estándares que utilizan. También se detallan los aspectos de seguridad que afectan a este tipo de redes. Posteriormente se introduce el concepto de malware, los diferentes tipos que existen y los modelos de propagación de malware que se han definido, haciendo hincapié en los modelos individuales, en particular los basados en agentes.

## 2.1. Redes de Sensores Inalámbricas

Las redes de sensores inalámbricas están formadas por un conjunto de sensores inteligentes que se encargan de monitorizar el área donde ha sido desplegada. Estos sensores son los responsables de recopilar los datos de las variables que se quiere monitorizar. Normalmente la cantidad de datos recogidos es considerable. Estos datos son enviados a la estación base, y posteriormente a un servidor para su análisis y toma de decisiones. Las WSN proporcionan una solución económica, de bajo consumo de energía y además requieren poca o ninguna supervisión, son redes de fácil despliegue, control y mantenimiento, por lo que están siendo ampliamente utilizadas en el IoT [5, 21].

Cada nodo de una red de sensores inalámbrica está compuesto por microprocesadores o microcontroladores integrados, batería, memoria, antena, puertos de comunicación y una

interfaz para la medición de las condiciones de su entorno (temperatura, humedad, luz, movimiento, etc.). La comunicación entre los sensores de la red, además de ser inalámbrica, no requiere una infraestructura preexistente. Esta comunicación sigue el estándar IEEE 802.15.4 y utiliza protocolos de comunicación jerárquicos, centrados en los datos o basados en la localización geográfica [22].

### 2.1.1. Características

Para disponer de una red de sensores eficiente y que se adapte al entorno donde va a ser desplegada es necesario analizar sus particularidades. A continuación, se describen algunas características generales que caracterizan una red de sensores [23]:

- Fenómeno de interés: depende del entorno donde la red se va a desplegar y operar. Los fenómenos de interés pueden ser puramente físicos, como es el caso de fábricas en las que se dan fugas de productos químicos tóxicos o contaminantes, o instalaciones de almacenamiento de residuos nucleares e incendios forestales; o pueden ser fenómenos observables, como la aparición de anomalías aéreas debidas al movimiento de una nave o fallos en un sistema integrado.

- Modo de despliegue: existen dos estrategias para el despliegue de los nodos, que pueden ser aleatorias o precisas. El despliegue aleatorio consiste en distribuir aleatoriamente los nodos en el entorno, y por el contrario, en el despliegue preciso, los nodos se disponen de forma manual o con una configuración preestablecida.

- Organización y arquitectura: la red de sensores se puede organizar según dos tipos de arquitectura: plana o jerárquica. En la organización plana, todos los nodos de la red tienen la misma funcionalidad; sin embargo, en la organización jerárquica, se agrupan los nodos de acuerdo a sus funciones, es decir, puede haber sensores con propósito general o sensores con tareas específicas, como el procesamiento de los datos.

- Vida útil: la duración de la fuente de alimentación de energía de los nodos, que puede afectar al funcionamiento de la red a medida que pasa el tiempo.

- Comunicación ad-hoc: la falta de infraestructura en las WSN crea la necesidad de utilizar protocolos que sean capaces de establecer y mantener una infraestructura, a través del autodescubrimiento (los nodos pueden descubrir a otros nodos vecinos), configuración (no es necesaria una configuración manual), enrutamiento (búsqueda de rutas para la transmisión de los datos) y tolerancia a fallos.

- Autodescubrimiento y organización: los protocolos se encargan de descubrir nuevos nodos y organizar la infraestructura inalámbrica, independientemente de los cambios en la topología o los fallos en los nodos; algunos de estos protocolos han sido adaptados de los protocolos de Bluetooth y otros han sido desarrollados específicamente para WSN, como es el caso del protocolo Zigbee [24].

- Conectividad y cobertura: es importante analizar la región que debe cubrir la red para asegurar la conexión entre los nodos.

Las redes de sensores inalámbricas se pueden desplegar en tierra, bajo tierra y bajo el agua. Más específicamente, se clasifican en: terrestres (el despliegue se realiza sobre la superficie de la tierra, en un área determinada), subterráneas (los sensores se entierran bajo tierra o en una cueva o mina utilizada para monitorizar las condiciones subterráneas), submarinas (despliegue bajo el agua), multimedia (sensores de bajo coste equipados con cámaras y micrófonos para monitorizar y seguir eventos en formato multimedia, como de vídeo, audio e imágenes), y móviles (sensores que pueden moverse por sí mismos e interactuar con el entorno físico).

Si se tiene en cuenta el objetivo o variable a medir, la red puede estar formada por sensores de temperatura, humedad, movimiento, condiciones de luz, presión, niveles de ruido, detección de objetos u otros [25]. Dependiendo de la finalidad para la que sea diseñada la red, se consideran las siguientes características [22]:

- Fidelidad: característica que abarca una serie de parámetros que pueden afectar al rendimiento de la red, como pueden ser la coherencia en la trasmisión de los datos, la precisión y latencia en la detección de eventos u otras medidas relacionadas con la calidad del servicio, que pueden depender del fenómeno de interés y el entorno operativo.

- Escalabilidad: se refiere a la propiedad de mantener la fidelidad de la red cuando se aumenta la cantidad de nodos. La escalabilidad se formula también en términos de confiabilidad, capacidad de la red, consumo de energía y cualquier otro parámetro que se pueda ver afectado por el número de nodos.

- Consumo de energía: las limitaciones de energía en el diseño de la red deben considerarse de acuerdo a la fuente de energía y el fenómeno de interés, ya que pueden existir entornos donde la recarga de energía puede ser muy complicada o imposible.

- Implementación: puede realizarse de forma planificada o aleatoria, y puede ser un proceso iterativo, ya que los nodos pueden ser agregados en una única ocasión o perió-

dicamente. Los parámetros que pueden verse afectados durante la implementación son la densidad de los nodos, cobertura, fiabilidad, asignación de tareas y comunicaciones.

- Topología: puede ser estática o dinámica, es decir ad-hoc. Una WSN puede evolucionar a lo largo del tiempo, debido a la gran cantidad de actividades que realiza y por lo tanto la topología puede cambiar en cualquier momento. Las características de la red que puede afectar la topología son la latencia, robustez y capacidad, enrutamiento y procesamiento.

- Cobertura: mide el grado del área de cobertura de la red, puede ser escasa, densa o redundante, además es determinada por la demanda de la aplicación.

- Comunicación y enrutamiento: factores como el ancho de banda, procesamiento, energía, y las operaciones en zonas hostiles, influyen directamente en la topología y cobertura de la red, por lo tanto, los nodos deben ser tolerantes a fallos; por esta razón, los protocolos tradicionales no son los adecuados para las WSN. En los últimos años, se han propuesto varios protocolos para optimizar una métrica de fidelidad (por ejemplo, las mediciones) basado en las limitaciones de funcionamiento (como la energía), siendo los protocolos más conocidos los que incluyen el enrutamiento centrado en los datos (SPIN: *Sensor Protocols for Information via Negotiation*) [26], la emisión de mensajes de acuerdo a la eficiencia energética de los nodos (GEAR: *Geographic and Energy Aware Routing*) [27], o el enrutamiento jerárquico (LEACH: *Low-Energy Adaptive Clustering Hierarchy*) [28].

- Seguridad: tiene como objetivo la confidencialidad; es decir, un intruso no debe ser capaz de robar o leer los datos; la integridad, que implica que los datos no deben manipulados o eliminados; y la disponibilidad de los datos, es decir, que el flujo de datos no debe ser interrumpido. Las WSN pueden ser vulnerables a algunos ataques como de denegación de servicio (DoS, *Denial-of-Service*), espionaje, análisis de tráfico, replicación de nodos y ataques físicos [29].

En términos generales, un sistema complejo consta de un gran número de componentes heterogéneos que interactúan entre sí. Un ejemplo de un sistema complejo es el clima, puesto que para entender su comportamiento es necesario analizar varios factores como la temperatura, la humedad, la presión atmosférica o el viento, entre otros. Dadas las características de las redes de sensores inalámbricas, estas pueden considerarse como sistemas complejos. Los principales elementos que caracterizan estos sistemas son los siguientes [30]:

- Complejidad de los estados del entorno: viene dada por el número posible de estados del sistema, incluyendo el número y rango de valores de los parámetros que definen el modelo.
- Complejidad de la estructura: está formada por las relaciones y dependencias de los componentes del sistema.
- Complejidad del comportamiento: se refiere al comportamiento y los patrones de interacción de los componentes del sistema.
- Complejidad temporal: alude al comportamiento dependiente del tiempo así como al estado de los componentes del sistema.

### 2.1.2. Aplicaciones

Las redes de sensores inalámbricas tienen aplicaciones en diferentes áreas, con diversas funcionalidades. Estas aplicaciones, pueden clasificarse en dos categorías de acuerdo a su objetivo: aplicaciones de monitorización remota y aplicaciones de seguimiento de objetos móviles. Además, según la ubicación que se considere, pueden ser aplicaciones de interior o aplicaciones de exterior. La Figura 2.1 muestra una clasificación de posibles aplicaciones de las WSN.

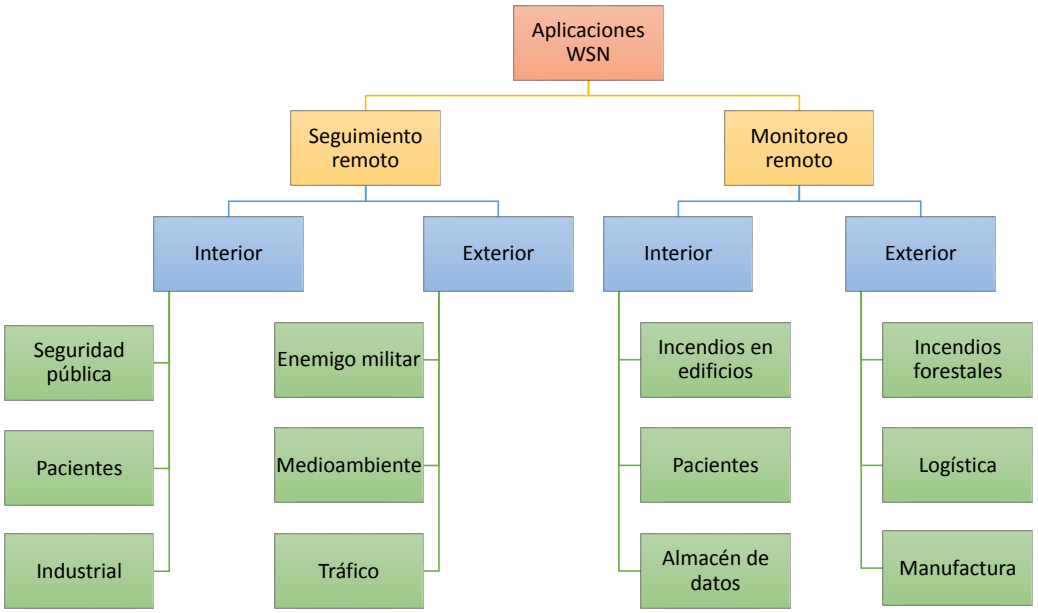



Figura 2.1 Posibles aplicaciones de las WSN.

Las aplicaciones de seguimiento remoto actualizan los resultados en tiempo real. Sin embargo, las aplicaciones de monitorización remota, miden las condiciones del entorno periódicamente y envían los resultados de diferentes modos: en un intervalo de tiempo preestablecido, cuando se alcanza un umbral predefinido en un evento específico o por solicitud del usuario [5].

Por otra parte, las aplicaciones de interior incluyen la monitorización de edificios y oficinas, realizando mediciones de la humedad y calidad del aire o la detección de incendios. Por el contrario, las aplicaciones de exterior incluyen la vigilancia en lugares hostiles, como pueden ser la detección de terremotos, la monitorización de gases en las fábricas de productos químicos, la detección de erupciones de volcanes, la monitorización de las condiciones medioambientales en los cultivos agrícolas o la detección de deslizamientos de tierra e inundaciones [25].

La investigación de las redes de sensores tiene su origen en la agencia militar de Estados Unidos, DARPA (*Defense Advanced Research Projects Agency*), con el proyecto *Distributed Sensor Networks* (DSN), que estudió la utilización de Arpanet (predecesor del actual Internet) para la comunicación entre redes de sensores. Se suponía que la red tenía muchos nodos de detección de bajo coste distribuidos espacialmente, que colaboran entre sí, pero que operaban de forma autónoma, y que la información se enrutaba a cualquier nodo que pudiera usarla mejor [31]. Posteriormente, se ha implementado en múltiples áreas con gran utilidad. A continuación se relacionan algunas de las áreas donde se utilizan las WSN:

- Militar: las WSN se utilizan en el área militar como sistema de contramedidas para las guerras urbanas, para la localización de tiradores y clasificación de armas mediante sensores utilizables por los soldados, o para localizar un tirador usando sistemas de detección de disparos de los soldados, entre otras cosas. Las aplicaciones en esta área requieren un rápido despliegue de las redes, que incluyan además, mediciones robustas en entornos hostiles, con larga vida útil, manejo eficiente de la energía y capacidad de procesamiento confiable [23].

- Industrial: las aplicaciones de WSN en el área industrial son muy amplias debido a los beneficios que brindan como la instalación y mantenimiento flexible de los dispositivos en el campo, la reducción de los costes, además de evitar los problemas que conlleva una red cableada. Se ha creado, por ejemplo, unas WSN para la monitorización del estado y la evaluación de la energía de máquinas eléctricas, y también para monitorizar la red de abastecimiento de agua para la industria del petróleo y del gas [25].

- Medioambiental: las WSN facilitan el estudio de los procesos de la naturaleza, algunos de los cuales pueden ser peligrosos para las personas. Por lo general, se realizan moni-

torizaciones a gran escala, como la exploración de las masas de tierra y agua, vigilancia de la contaminación, volcanes o movimientos sísmicos, detección de la erosión del suelo, inundaciones o incendios forestales; además, permiten la monitorización de hábitats para el seguimiento de la migración de aves [22].

- Atención médica: las aplicaciones médicas que utilizan WSN tienen como objetivo mejorar la atención sanitaria y los servicios de vigilancia de los pacientes. En este entorno se consideran diferentes actores, como los niños, los ancianos, los enfermos crónicos, los cuidadores, los profesionales de la salud, los administradores y los informáticos. Algunas de las ventajas que ofrecen estos sistemas son la capacidad de monitorización remota o la medición de signos vitales en tiempo real. Para desarrollar estas aplicaciones se encuentran disponibles cinco subsistemas, a saber, (1) las redes de área corporal (BAN, *Body Area Network*), (2) las redes de área personal (PAN, *Personal Area Network*), (3) las redes de área amplia (WAN *Wide Area Network*), (4) las puertas de enlace permiten la conexión de las BAN y PAN con las WAN, y (5) las aplicaciones de monitorización de la salud del usuario final [32].

- Vida diaria: las aplicaciones en esta área tienen como finalidad facilitar la vida a los usuarios, tanto en sus hogares, como en los lugares de trabajo, los supermercados, comercios y otros espacios. Algunos ejemplos de estas aplicaciones son los sistemas de gestión de los aparcamientos inteligentes, o las redes de objetos cotidianos en un entorno doméstico inteligente (domótica) [33].

- Multimedia: se habla de aplicaciones multimedia de las WSN cuando su objetivo es la transmisión de audio y vídeo, o de imágenes fijas y o datos escalares. Estas aplicaciones pueden dividirse en las siguientes categorías: de seguimiento, en las que los sensores de vídeo y audio mejoran y complementan los sistemas de seguimiento existentes contra la delincuencia y los ataques terroristas; de vigilancia del tráfico y aplicación de la ley, que permiten vigilar el tráfico para brindar rutas alternativas y evitar atascos; personal y de salud, que pueden utilizarse como complemento a las aplicaciones de salud para estudiar el comportamiento de algunas personas; de juegos, que sirven para mejorar el entorno de los jugadores; y ambiental e industrial, para la vigilancia de entornos que utilizan fuentes acústicas y de vídeo, y que deben transmitirse en tiempo real [34].

### 2.1.3. Arquitectura

La arquitectura de una red de sensores inalámbrica se describe a partir de los elementos que la componen, de su funcionamiento y su organización. Una WSN está organizada con

cuatro elementos básicos: los nodos, la estación base, el fenómeno de interés y los usuarios. Los nodos, también denominados sensores, pueden ser de tres tipos: los nodos sensores, que recopilan los datos directamente del entorno; el nodo receptor (*cluster-head node*), que se encarga de la transmisión de los datos; y finalmente, el nodo enrutador (*sink node*), que recibe los datos y tiene la función de puerta de enlace de la red. La Figura 2.2 muestra la arquitectura de una WSN con diferentes distribuciones posibles de los nodos.

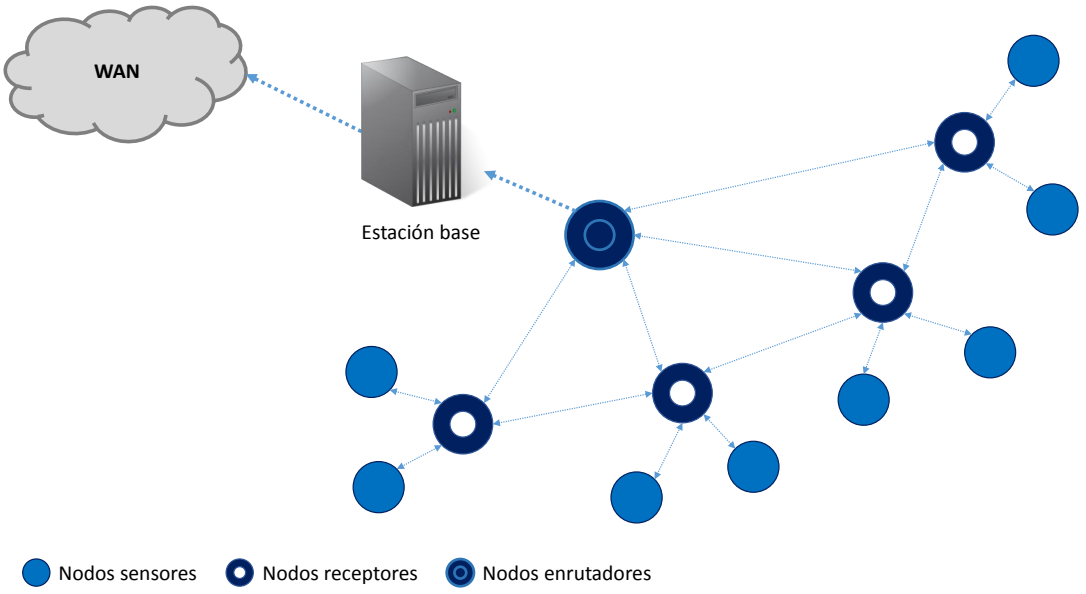



Figura 2.2 Arquitectura de una WSN [35].

La estación base recibe los datos desde el nodo enrutador y proporciona la comunicación con el usuario a través de Internet. El fenómeno de interés es el entorno donde los nodos sensores realizan mediciones. Finalmente, el usuario es el que obtiene la información, la analiza y toma las decisiones necesarias.

La arquitectura que presenta una red de sensores inalámbrica puede ser de dos tipos: centralizada o distribuida. En la arquitectura centralizada el ciclo de trabajo de los nodos está dividido en cuatro fases: despertar, hacer las mediciones, transmitir los datos y dormir. Las principales desventajas de esta arquitectura son los excesos de tráfico en las puertas de enlace, el tiempo de vida de la red suele ser más corto y tienen un mayor consumo de energía. La arquitectura distribuida es más utilizada en las WSN, está dividida en clústeres, donde los nodos solo pueden comunicarse con otros nodos que forman parte del mismo clúster; además, evitan los cuellos de botella y gestionan mejor el consumo de energía [25].

A continuación se detallan algunas características importantes de los nodos, las topologías y los protocolos de comunicación que intervienen en una red se sensores inalámbrica.

### Nodos sensores

Una red de sensores inalámbrica puede estar formada por miles de nodos, donde cada uno puede realizar diferentes funciones como medición, transmisión o intercambio de datos con redes externas [5]. No obstante, los nodos sensores son el elemento básico de estas redes, normalmente tienen la capacidad de recolectar datos del entorno y, en algunos casos, son capaces de analizar o enrutar los datos a su destino. Específicamente, un nodo sensor está formado por los siguientes componentes [25]:

- La unidad de medición, que está compuesta por dos subunidades: los sensores y los conversores analógico-digital (ADC, *analogue-digital converter*). Las señales analógicas producidas por los sensores al realizar mediciones, se tranforman en señales digitales a través del ADC, y posteriormente son enviadas a la unidad de procesamiento.
- La unidad de procesamiento está formada por microprocesadores de propósito general, que se encargan de gestionar las acciones que realiza un nodo, de acuerdo a su función en la red.
- La unidad de transmisión y recepción: se encarga de conectar el nodo a la red, a través de una antena interna, que suele estar integrada en el nodo. Cuando es necesario mayor rango de transmisión se pueden utilizar antenas resonantes externas.
- La fuente de energía la componen generalmente baterías de ión de litio, sin embargo, se han comenzado a utilizar células solares para alargar el periodo de vida de los nodos.
- Componentes adicionales: un nodo puede tener subunidades de acuerdo a su función en la aplicación, como puede ser un sistema de geolocalización, un generador de energía o un movilizador, para los casos en que es necesario mover el nodo a otro lugar.

Los nodos sensores se caracterizan por un consumo de energía extremadamente bajo, gestionan grandes cantidades de datos con bajo coste de producción, son autónomos y operan desatendidos, y por su reducido tamaño, se adaptan al entorno.

### Topologías

La topología hace referencia a la distribución de los nodos dentro de una red. Las topologías básicas en las redes inalámbricas de sensores son las siguientes (Figura 2.3) [22]:

- Estrella: la estación base o el nodo enrutador ocupan la posición central de la estrella, todos los demás nodos sensores se conectan a ese nodo central. Esto quiere decir que los nodos sensores no se pueden comunicar directamente entre sí, y deben hacerlo a través del nodo central. Esta topología ofrece un bajo consumo de energía; sin embargo, la principal desventaja es que toda la comunicación de la red depende únicamente del nodo central, lo que genera un punto único de fallo, es decir, que la red puede dejar de funcionar si el nodo central falla.

- Malla: en esta topología todos los nodos pueden comunicarse con sus vecinos próximos. La principal ventaja de esta topología es que si un nodo falla, la comunicación entre dos nodos lejanos puede realizarse a través de diferentes caminos. Como desventaja, se puede considerar que el consumo de potencia es mayor debido a la transmisión redundante de datos.

- Híbrida: es una combinación de la topología en estrella con la topología en malla. Se puede decir que los nodos se agrupan en clústeres, donde los nodos sensores se comunican con el nodo central, que corresponde a un nodo receptor, y estos nodos se comunican a su vez con los nodos enrutadores. Esta topología reúne las ventajas de las topologías estrella y malla, ya que el consumo de energía de los nodos es bajo y el nodo central de la malla tiene mayor potencia.

- Existen otras topologías que pueden ser utilizadas en las WSN, como las topologías en árbol (serie de redes en estrella conectadas a un nodo troncal, que generalmente es un *switch*), la topología punto a punto (comunicación bidireccional entre dos nodos), la topología en anillo (cada nodo tiene una única conexión de entrada y salida), y la topología en bus (los nodos están conectados a un único canal de comunicaciones).

Las redes complejas como las redes de sensores, redes computacionales o redes sociales, están representados por un grafo. Un grafo es una estructura matemática que consiste en vértices, nodos o puntos, que están conectados por aristas, enlaces o líneas. Algunos tipos de redes complejas que pueden darse en las WSN son los siguientes [36]:

- Redes de escala libre: estas redes tienen un grado de distribución basado en la ley de la potencia, donde existen nodos con un gran número de conexiones a otros nodos. Además, alguna de las ventajas de estas redes es que se pueden añadir nuevos nodos de manera dinámica y que existe preferencia por los enlaces a nodos con gran cantidad de conexiones.

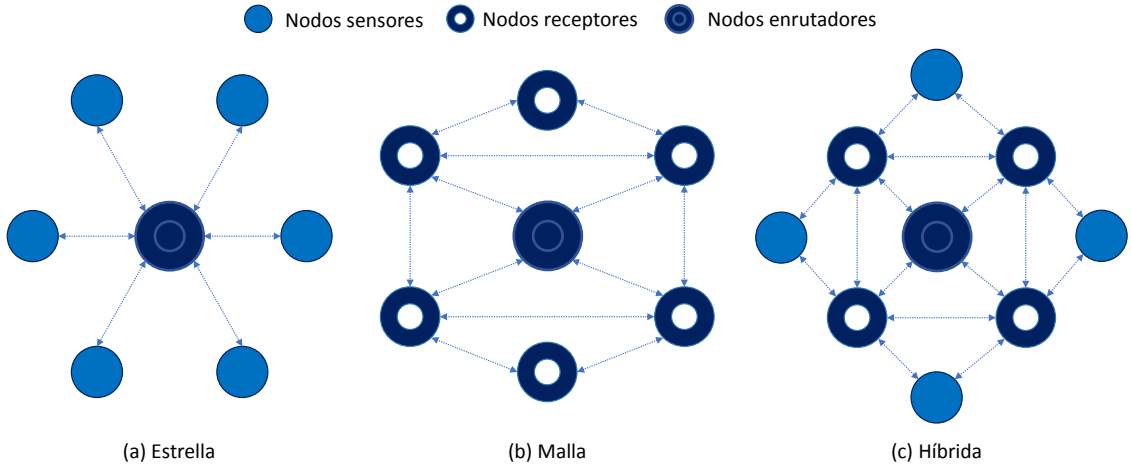



Figura 2.3 Topologías en WSN: (a) en forma de estrella, (b) de malla, e (c) híbrida [22]

- Redes de mundo pequeño: se construyen a partir de una red en forma de anillo reconectando aleatoriamente sus enlaces, además, gran parte de los nodos de la red no son vecinos entre sí, es decir, no tienen un enlace directo. La principal característica de estas redes es que la transmisión de información entre dos nodos se realiza en poco tiempo. Las ventajas de estas redes son que ofrecen menor latencia en la transmisión de los datos y mayor robustez en la conexión de la red.

No obstante, las redes de sensores inalámbricas también pueden ser desplegadas de forma aleatoria o dinámica, es decir, se crea una red de tipo ad-hoc, donde los nodos deben ser capaces de autoconfigurarse para establecer la comunicación entre nodos vecinos y definir el enrutamiento de los datos; para esto las WSN utilizan protocolos especializados, como el protocolo de autoorganización (SOP) [37]. Se describen a continuación los protocolos de enrutamiento que han sido creados con esta finalidad en las WSN.

### 2.1.4. Protocolos de enrutamiento

Los protocolos de enrutamiento describen la forma de transmitir los datos a través de la red y se clasifican de forma general en protocolos proactivos y protocolos reactivos. Los protocolos proactivos generan rutas para el envío de datos automáticamente sin que sean solicitadas y se actualizan constantemente. Posteriormente se podrá buscar la ruta en una tabla de enrutamiento cuando sea solicitada. Este tipo de protocolos suele utilizarse cuando las redes son estáticas. Por el contrario, los protocolos reactivos calculan las rutas por petición; esto incrementa el tiempo de latencia en la entrega de los datos. Estos protocolos son

utilizados en las redes dinámicas, como las redes de sensores inalámbricas que se estudian en esta memoria.

Los protocolos para las WSN pueden clasificarse en dos grandes grupos, según su función y según la estructura de la red. Cabe señalar que algunos protocolos pueden pertenecer a varias categorías. Los protocolos según su función pueden dividirse en las cinco subcategorías siguientes: basados en solicitud, basados en la negociación, basados en multirutas, basados en la calidad de servicio y basados en la coherencia [5].

Además, los protocolos de una red de sensores inalámbrica se pueden clasificar atendiendo a su estructura de red en los siguientes [22]:

- Protocolos centrados en los datos: estos protocolos agregan información basada en las propiedades de los datos para eliminar las redundancias mientras son enviados a través de la red. Algunos ejemplos de estos protocolos son:
  - SPIN: este protocolo establece un modelo de negociación, que se inicia preguntando a los nodos si tienen interés en los datos, que son enviados solo a aquellos nodos que han hecho la solicitud. el protocolo SPIN utiliza tres tipos de paquetes: ADV para dar el aviso de nuevos datos, REQ es una solicitud de datos y DATA son los propios datos [26].
  - *Flooding and gossiping*: en los protocolos de inundación (*flooding*) los nodos envían los paquetes a todos sus vecinos hasta que llega a su destino, estableciendo un número máximo de saltos. Los protocolos *gossiping* buscan mejorar este enfoque, enviando el paquete solo a un vecino seleccionada al azar [22].
  - Difusión directa: el proceso de recolección de los datos es iniciado por un nodo enrutador o por la estación base. Primero, el nodo enrutador envía un paquete a todos sus vecinos, y estos nodos lo envían a su vez a todos sus vecinos y así sucesivamente hasta que se encuentra el nodo que contiene los datos. El paquete inicial incluye un valor llamado gradiente. Como consecuencia de ello, el nodo que tiene los datos solicitados, envía el paquete a través de diferentes rutas de acuerdo a dicho gradiente. Por último, se selecciona la mejor ruta [5].
- Protocolos jerárquicos: estos protocolos organizan los nodos en grupos llamados clústers, donde se elige un nodo cabecero. Este nodo cabecero o principal se encarga de procesar y enviar los datos, mientras que los demás nodos se encargan de realizar las mediciones de los datos en el entorno. Algunos de los protocolos jerárquicos más conocidos son:

- LEACH: este protocolo utiliza una rotación aleatoria del nodo cabecero; este nodo es seleccionado a partir de un enfoque probabilístico donde cada nodo puede nominarse como nodo cabecero, independientemente de la decisión de los demás nodos. Con esto se logra minimizar la sobrecarga en las comunicaciones y una mejor coordinación entre los nodos, disminuyendo la perdida de energía [28].
- PEGASIS (*Power-Efficient GAthering in Sensor Information Systems*): el principal objetivo de este protocolo es mejorar el tiempo de vida de la red. Un único nodo, diferente en cada ocasión, se encarga de la trasmisión de los datos a la estación base, con esto se consigue minimizar la energía consumida por cada nodo [22].
- TEEN (*Threshold sensitive Energy Efficient sensor Network protocol*): este protocolo ha sido diseñado para las aplicaciones donde el tiempo de reacción es crítico. Por ejemplo, TEEN es utilizado para las aplicaciones que miden temperatura y presión, o para realizar mediciones en tiempo real como las alarmas de incendio [5].
- SOP (*Self-Organizing Protocol*): la organización de los nodos y la creación de las tablas de enrutamiento se realizan en cuatro etapas, descubrimiento, organización, reorganización y mantenimiento. En la etapa de descubrimiento se identifican los nodos; en la etapa de organización se forman los clústeres; posteriormente, en la etapa de reorganización, se pueden reagrupar los clústeres; y finalmente, la etapa de mantenimiento se encarga de dar la respuesta a posibles fallos. La desventaja de este protocolo es que la organización de los nodos conlleva una sobrecarga adicional de energía [38].
- EECR (*Energy efficient clustering and routing*): el objetivo de este algoritmo es brindar una solución eficiente en el uso de la energía, basada en la creación de los clústeres y las tablas de enrutamiento. Este protocolo funciona en dos fases, primero se realiza la configuración, donde la selección de los nodos cabeceros se realiza entre todos los nodos que pertenecen al clúster y la estación base; luego en la fase de comunicación, se miden los datos, se procesan y se transmiten a la estación base. El nodo cabecero cambia cada cierto tiempo, y los nodos pasan a estado dormido para ahorrar energía, siempre que no estén realizando sus funciones [39].

- Protocolos basados en localización: estos protocolos crean rutas basadas en la información de la ubicación de los nodos. La distancia entre los nodos se calcula para estimar

la energía necesaria para la transmisión de los datos, con la finalidad de gestionar la energía de forma eficiente.

- MECN (*Minimum Energy Communication Network*): este protocolo utiliza la información de la geolocalización de los nodos, para identificar las rutas que minimizan el consumo de energía durante la transmisión de los datos [22].
- GAF (*Geographic Adaptive Fidelity*): utiliza la información de geolocalización para encender y apagar los nodos, según las necesidades de la red, con el fin de conservar energía [40].
- GEAR: en este protocolo el proceso de consulta de datos se realiza a los nodos que se encuentran geolocalizados en una región seleccionada, con la finalidad de mejorar el uso de la energía mediante el proceso de difusión dirigida [27].

Debido a las características y singularidades de las WSN ha sido necesario adaptar la pila de capas sobre la que operan estos protocolos. Así, la pila de protocolos de las redes de sensores inalámbricas está organizada en cinco capas, representadas en la Figura 2.4, a diferencia del modelo OSI (Interconexión de Sistemas Abiertos) que consta de siete capas. El modelo OSI ha sido la base de la pila de protocolos de las WSN. En estas redes, cada una de las capas de la pila tiene una función específica e independiente del resto de capas. Las capas de la pila física y de enlace de datos funcionan bajo el estándar 802.15.4, que se encarga de la comunicación inalámbrica de corto alcance; y las capas de red, transporte y aplicación utilizan el protocolo Zigbee para establecer la topología, las características de autenticación, el cifrado de los datos y los servicios de la aplicación. Las funciones de estas capas son las siguientes [5]:

- Capa física: define y gestiona las conexiones entre los dispositivos y el medio de comunicación, genera la frecuencia, detecta y modula las señales, además de cifrar los datos.
- Capa de enlace de datos: es la responsable de que los nodos puedan acceder al medio de comunicación y transmitir los datos con éxito, para esto incluye servicios de control de acceso al medio, detección y corrección de errores.
- Capa de red: establece las rutas de comunicaciones entre los nodos de la red, con el fin de enrutar correctamente los paquetes de datos. Algunos protocolos de enrutamiento son más eficientes y permiten enviar un paquete de un punto a otro con el menor número posible de saltos entre nodos.

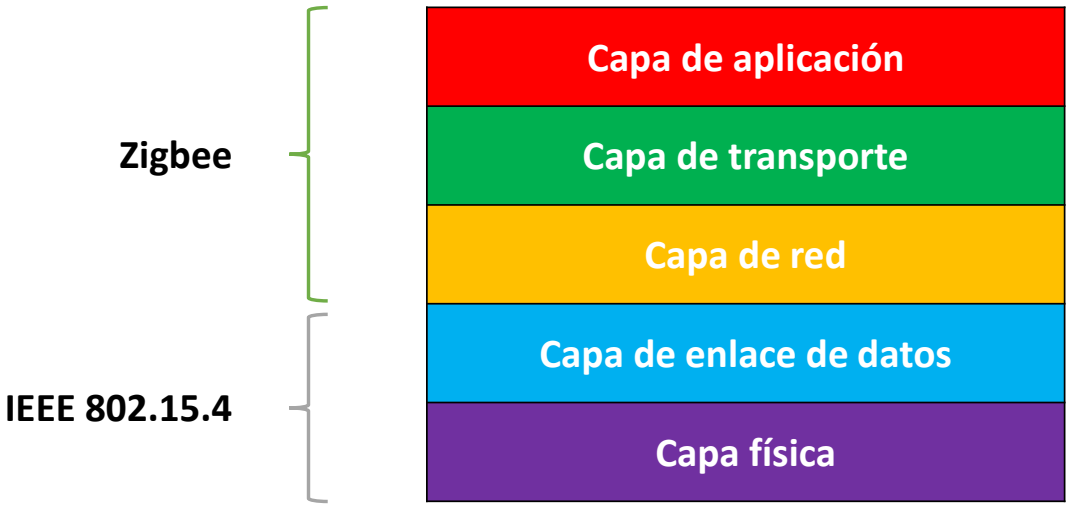



Figura 2.4 Pila de protocolos de una WSN.

- Capa de transporte: existen varios protocolos para esta capa, los más utilizados son el protocolo de control de transmisión (TCP) y el protocolo de datagramas de usuario (UDP). El protocolo TCP ofrece una mayor confiabilidad, ya que incluye gestión de errores, así como el control de transmisión y flujo. Sin embargo, el protocolo UDP, ofrece un servicio no confiable, con menor control de errores, transmisión y flujo de datos.

- Capa de aplicación: se ocupa principalmente del procesamiento de la información recopilada, del cifrado y almacenamiento de datos. Además, la capa de aplicación explora las capas subyacentes para detectar si hay suficientes recursos de red y servicios disponibles para satisfacer las solicitudes de la red.

### 2.1.5. Estándares

Un estándar hace referencia al nivel de calidad requerido para un proceso; pueden dividirse en dos categorías según el propósito del diseño: estándares públicos y estándares privados. Actualmente, no existe un estándar unificado para las WSN, sin embargo, para que un estándar pueda ser aplicado a estas redes debe tener, como requisito de diseño, un consumo de energía bajo. Algunos de los estándares más utilizados en WSN son los siguientes [25]:

- Estándar IEEE 802.15.4: fue diseñado en 2003 para las redes de área personal con tasas bajas de transmisión de datos, donde las aplicaciones tienen recursos de energía

y hardware limitados. La arquitectura de este estándar consta de dos capas: la capa física y la capa de control de acceso al medio (MAC, *Media Access Control*). La capa física es utilizada por el transceptor de radio (dispositivo formado por un transmisor y un receptor), para sincronizar la comunicación; y la capa MAC proporciona las reglas para la transferencia de los datos a través de la capa física. Este estándar permite diseñar aplicaciones a bajo nivel e interactuar directamente con la transferencia de los datos. Los tipos de dispositivos que participan en el sistema son el dispositivo de función completa (FFD) y el dispositivo de función reducida (RFD). Las topologías que permite este estándar son la de estrella, punto a punto e híbrida.

- WirelessHART: proporciona un protocolo de comunicación para aplicaciones de medición y control de procesos, basado también en estándar IEEE 802.15.4. El WirelessHART brinda soluciones de cifrado, verificación, autenticación y gestión de claves; además, soporta las topologías estrella, malla e híbrida. Una de red basada en este estándar está formada por los sensores, la puerta de enlace, ordenadores y el administrador de red.

- ISA100.11a: es un estándar para aplicaciones de monitorización inalámbrica segura, confiable y de baja velocidad de transmisión de datos. Sus principales características son el bajo consumo de energía, la escalabilidad, la robustez y la interoperabilidad con otros dispositivos inalámbricas. Ofrece funciones de seguridad sencillas, flexibles y escalables. Este estándar define perfiles de roles para los dispositivos de acuerdo a sus capacidades y funciones y las topologías que utiliza son de malla y de estrella.

- 6LoWPAN (*IPv6 over Low power Wireless Personal Area Networks*): este protocolo permite el uso de IPv6 a través de una red basada en el estándar IEEE 802.15.4; esto permite que los nodos puedan comunicarse directamente con otros dispositivos con dirección IP. Para realizar esta comunicación, el estándar proporciona una capa de adaptación, un nuevo formato del paquete y también la gestión de las direcciones. Un LoWPAN es la colección de nodos 6LoWPAN, estos nodos comparten un prefijo de dirección IPv6 común (los primeros 64 bits de una dirección IPv6).

Otros estándares utilizados en las WSN son Z-Wave, *Impulse Radio Ultra Wide Bandwidth Technology* 802.15.4a (IR UWB), Wibree BLE, INSTEON, Wavenis, ANT, MyriaNed o EnOcean.

### 2.1.6. Seguridad

La implementación de medidas de seguridad en las redes de sensores inalámbricas supone un desafío para los administradores de la red debido a las características computacionales de estas redes. Estas medidas deben ser establecidas durante el diseño de la red, considerando la finalidad para la que se construye la red, así como el fenómeno de interés y la fuente de alimentación energética que utilizará. Además, la mayoría de las técnicas de seguridad informática tradicionales no son compatibles con las WSN [41].

Generalmente, el motivo para llevar a cabo un ataque a una WSN es el beneficio que puede obtener el atacante, como interferir en el propósito de la red o acceder a datos sensibles que están siendo monitorizados o transmitidos por la red; sin embargo, los atacantes pueden incurrir en vandalismo sin que esto conlleve un beneficio para ellos. Consecuentemente, el desarrollo de los mecanismos de seguridad especializados debe adaptarse a las siguientes características y limitaciones de las WSN [42]:

- Recursos muy limitados: los mecanismos de seguridad necesitan unos recursos mínimos para su implementación, como la memoria para la ejecución del algoritmo y el espacio de almacenamiento del código fuente; además, se debe considerar el impacto energético que conlleva cada ejecución del mecanismo en el sensor.

- Comunicación poco fiable: el enrutamiento de las WSN suele ser poco fiable debido a que los paquetes de datos pueden dañarse o perderse por errores en el canal de comunicación, por pérdida de conexión o porque la capacidad de procesamiento de los nodos es insuficiente. Por otro lado, la latencia en la red provocada por el enrutamiento multisalto, la congestión de la red y el procesamiento de un nodo, pueden dificultar la sincronización entre los nodos, generando fallos críticos en los mecanismos de seguridad.

- Operación desatendida: puede tener como consecuencia varios problemas como la exposición a ataques físicos, ya sea por personas malintencionadas o fenómenos naturales como la lluvia; la gestión remota de la red que entorpece la detección de anomalías físicas en los nodos; y la organización de una red distribuida sin punto central de gestión, que puede afectar al funcionamiento de la red, si esta no ha sido diseñada correctamente.

Además de estas características, comunes a las redes de sensores inalámbricas, debe tenerse en cuenta que cualquier sistema que recoge y transmite información, se considera seguro si cumple con los siguientes requisitos [5, 42, 43]:

- Confidencialidad: los datos gestionados por las WSN pueden ser datos críticos que no deben ser interceptados por terceros. Para evitar esto, es importante crear un canal de comunicación seguro y cifrar los datos que se transmiten.
- Integridad: el objetivo de la integridad de la información es evitar la pérdida o manipulación de los paquetes de datos durante la transmisión entre nodos y hacia la estación base.
- Datos recientes: se busca que los datos que se transmitan por la red estén actualizados y no hayan sido manipulados con datos antiguos. Para asegurar la transmisión de los últimos datos, se puede agregar al paquete un contador de tiempo.
- Disponibilidad: los datos siempre deben estar disponibles cuando sean solicitados; sin embargo, puede haber problemas como que un nodo agote su energía antes de transmitir los datos. Por esto es importante considerar la energía extra que consumen los mecanismos de seguridad como puede ser un sistema de cifrado, disminuir las redundancias y evitar los puntos de fallos únicos.
- Autoorganización: se deben utilizar protocolos de organización eficientes, tanto para el envío de datos como en el manejo de la energía. Además, se debe considerar la gestión de claves y la construcción de relaciones de confianza entre nodos.
- Sincronización: para ahorrar energía, los sensores pueden estar en modo dormido o apagados por periodos de tiempo, es posible también programar las mediciones o calcular el tiempo de transmisión de un punto a otro.
- Localización segura: la geolocalización de los nodos permite detectar los fallos en la red de manera precisa; sin embargo, es posible que se pueda manipular la ubicación de los nodos a través de señales falsas.
- Autenticación: permite verificar que los datos transmitidos son auténticos y no han sido manipulados por un atacante. Para garantizar la autenticidad de los datos se pueden usar mecanismos criptográficos, que permitan calcular el código de verificación del paquete.
- No repudio: es una protección contra el rechazo de alguna de las partes involucradas en el envío de los datos, es decir, que es posible demostrar quien ha enviado y recibido dichos datos.

### Mecanismos de seguridad

Los mecanismos de seguridad son técnicas y métodos para implementar servicios de seguridad, utilizados para detectar y prevenir ataques, además de recuperar el funcionamiento normal de la red. Se han desarrollado una gran variedad de esquemas o mecanismos de seguridad que pueden ser implementados en una WSN, estos esquemas se pueden clasificar de la siguiente forma [44]:

- Mecanismos de alto nivel, que tienen las siguientes características:
  - Administración segura de la gestión de grupos: se requieren protocolos seguros para gestionar los grupos de nodos, de modo que permitan añadir nuevos nodos al grupo y mantener la comunicación segura. Generalmente, las claves de acceso son transmitidas por la estación base, por lo que es necesario autenticar que la información proceda de un grupo válido.
  - Detección de intrusos: el uso de grupos seguros puede ser un enfoque útil para la detección de intrusos descentralizada, permitiendo el ahorro de energía, menor requerimiento de memoria y una comunicación eficiente.
  - Envío seguro de datos: los valores detectados deben enviarse a la estación base para evitar grandes cantidades de tráfico. El envío de los datos puede realizarse desde diferentes puntos de la red, por lo que es necesario proteger los nodos y el canal de comunicación durante la transmisión de los datos.
- Mecanismos de bajo nivel, que incluyen lo siguiente:
  - Establecimiento de claves y configuración de la confianza: el establecimiento de claves criptográficas es un requerimiento para la configuración de una WSN; sin embargo, hay que considerar que la capacidad computacional de los nodos es limitada y las claves pueden requerir más capacidad de la disponible.
  - Secreto y autenticación: la criptografía de clave secreta en la capa de enlace puede simplificar la implementación, pero existe riesgo de que los nodos intermedios puedan escuchar o alterar los datos.
  - Privacidad: se debe considerar que se pueden agregar nuevos nodos a la red inicialmente desplegada, por lo que se hace necesario establecer un proceso de reconocimiento legítimo de los nuevos nodos.
  - Robustez ante la denegación de servicio: un ataque DoS de denegación de servicio puede producirse a través de una señal de alta energía, interrumpiendo la operación normal de la red, lo cual puede ocasionar el colapso del sistema.

- Enrutamiento seguro: la mayoría de los protocolos de enrutamiento pueden sufrir vulnerabilidades de seguridad, por ejemplo, la inyección de información de enrutamiento malicioso en la red; esto se podría evitar con métodos de autenticación.
- Resistencia a la captura de nodos: la ubicación de la red puede exponer a los nodos a ser capturados, lo que tendría como consecuencia que pudieran ser manipulados con códigos maliciosos o reemplazados para extraer información de la red. Algunas soluciones para evitar este problema son embalajes resistentes a la manipulación o soluciones algorítmicas, por ejemplo, un esquema de cifrado dinámico basado en cadenas de bloques (*Blockchain*) [45].

### Criptografía

Como es bien sabido, la criptografía tiene como objetivo proporcionar técnicas de cifrado y descifrado de la información que se transmite, para que no pueda ser legible por un tercero no autorizado. La criptografía es la base de gran parte de los mecanismos de seguridad para las WSN, que han sido diseñados para garantizar la confidencialidad, integridad, disponibilidad, autenticación, autoorganización, sincronización y localización segura de los datos. La criptografía utilizada en las redes de sensores inalámbricas ha sido la criptografía ligera, tanto la de clave secreta o criptografía simétrica como la de clave pública o criptografía asimétrica.

La criptografía ligera (en inglés, *Lightweight Cryptography*), está diseñada para garantizar la seguridad en los sistemas que tienen recursos limitados para el procesamiento y la capacidad de energía. Este tipo de criptografía puede desarrollarse tanto a nivel de hardware como a nivel de software. Para medir el grado de optimización se utilizan a nivel de hardware, el tamaño del chip y el nivel de energía; y en el nivel de software, el tamaño del código y la complejidad de la memoria RAM. El desarrollo de un algoritmo de criptografía ligera debe considerar tres aspectos fundamentales: la seguridad versus el desempeño, la seguridad versus el coste, y finalmente, el desempeño versus el coste [46].

En la criptografía de clave secreta, el remitente utiliza una clave para cifrar el mensaje, que es la misma que conoce el receptor y le permite a este descifrar el mensaje. Existen varios enfoques que pueden aplicarse a WSN, el enfoque más sencillo consiste en el establecimiento de una única clave conocida por todos los nodos de la red. Otra posibilidad consiste en la distribución centralizada de claves, donde cada nodo comparte su clave solo con la estación base. Finalmente, un tercer enfoque sería pre-distribuir las claves, es decir, precargar en cada par de nodos una clave secreta antes del despliegue de la red, para esto se pueden utilizar esquemas como la pre-distribución de clave aleatoria (RKP, *Random Key Pre-Distribution*) o la distribución de claves determinista [22].

Por otro lado, en los sistemas de clave pública, el remitente utiliza la clave pública del destinatario para cifrar el mensaje, mientras este utiliza su clave privada para descifrarlo. El sistema asimétrico es más robusto contra ataques de captura de nodos, ya que tendría que descubrirse la clave privada, puesto que la pública ya es conocida. Algunos protocolos criptográficos que se están utilizando en WSN son los algoritmos de NtruEncrypt y Rabin, o los métodos basados en RSA [22].

### Protocolos de seguridad

Los protocolos de seguridad tienen la función de proteger la confidencialidad, integridad, disponibilidad, autenticidad y no repudio de los datos que se transmiten a través de una red, evitando que usuarios o dispositivos no autorizados puedan acceder a ellos. Algunos de estos protocolos que pueden implementarse en las WSN son los siguientes:

- TinySec: es un protocolo de seguridad en la capa de enlace para el sistema operativo TinyOS. Ha sido diseñado para generar paquetes de tamaño pequeño en la red. Este protocolo admite una opción que garantiza la seguridad: el cifrado con autenticación de identidad, donde se cifran los datos y se añadide un código de autenticación de mensaje (MAC, *Message Authentication Code*) al paquete; y adminte también el método de solo autenticación, donde los datos no son cifrados, pero la autenticación del paquete se realiza con el MAC [47].

- SPINS: está formado por el protocolo $\mu$TESLA, el protocolo SNEP y un tercer protocolo de enrutamiento basado en los dos anteriores. $\mu$TESLA se utiliza para la autenticación de identidad de radiodifusión y SNEP proporciona confidencialidad, autenticación de identidad entre dos nodos, actualización de los datos y baja sobrecarga en la comunicación. Este protocolo utiliza el algoritmo de cifrado en bloques RC5 para el cifrado de los datos; el remitente crea un MAC para que los paquetes sean difundidos unidireccionalmente, utilizando una clave conocida solo por ellos mismos, después de cierto tiempo se transmite la clave del paquete para que el receptor pueda autentificar el mensaje [48].

- LISP: utiliza el mecanismo de conmutación mediante el uso de nodos receptores y servidores de claves. Algunas de las ventajas de este protocolo que se pueden considerar son que utiliza una emisión de claves eficaz que no necesita que se envíen acuses de recibido o ACK (*acknowledgement*) y que usa bits de comprobación creados sin agregarlos al mensaje de datos. Además, debido a que no necesita ACK ni otros paquetes de control, es bastante fuerte contra ataques DoS, de Denegación de Servicio [49].

- Zigbee: define los protocolos de comunicación de alto nivel basado en el estándar IEEE 802.15.4. Los dispositivos ZigBee necesitan para su funcionamiento poca energía, por lo que la batería podría durar un largo periodo de tiempo; además, pasan gran parte del tiempo en estado dormido. Este protocolo utiliza el cifrado fuerte AIG-128. Zigbee proporciona datos actualizados a través de un contador que se restablece cuando se crea una nueva clave. Además, proporciona integridad y evita que un atacante cambie el mensaje. Por otro lado, la autenticación comprueba si se alcanza o no al remitente adecuado y evita que el atacante reemplace el dispositivo. La autenticación a nivel de red se logra utilizando una clave de red pública; y a nivel de dispositivo se logra utilizando una clave de enlace única entre dispositivos [50].

- LSec: proporciona autenticación de identidad y autorización, intercambio de clave seguro, mecanismo de defensa contra infracciones, privacidad de datos y uso de codificación asimétrica y simétrica al mismo tiempo [51].

- LISA: incluye las siguientes soluciones de seguridad: (1) la seguridad semántica, es decir, que el atacante solo puede extraer del texto cifrado información sin valor del mensaje sin cifrar. (2) La autenticación de identidad garantiza que los datos provienen del nodo correcto. (3) La protección contra ataques de repetición evita que se repitan mensajes antiguos. Por último, (4) la actualización de los datos, verifica en la estación base que el mensaje generado es posterior al anterior [52].

- MiniSec: proporciona alta seguridad a bajo consumo de energía. Se utilizan tres técnicas para lograrlo: primero, el método de cifrado de bloques para proporcionar privacidad y autenticación; luego, el vector de inicialización (IV), que utiliza pocos bits; y por último, los saltos básicos durante los modos de operación *unicast* y *broadcast*. En el modo *unicast*, el consumo de energía se reduce haciendo cálculos adicionales y utilizando contadores sincronizados, y en el modo *broadcast* se utiliza el mecanismo de filtro de Bloom. Skipjack se utiliza como algoritmo de cifrado y OCB (*Offset Codebook Mode*) como el modo de cifrado [53].

- LLSP: proporciona autenticación de identidad con un bajo consumo de energía; además, facilita la integridad de datos y la seguridad semántica utilizando algoritmos criptográficos [54].

**Amenazas, ataques y medidas de defensa**

Las redes de sensores inalámbricas están expuestas a múltiples ataques, al igual que ocurre en otros tipos de redes. La calidad y complejidad de los ataques ha ido en aumento

y estos ataques pueden clasificarse de diversas formas. Si se considera la capacidad del atacante, las amenazas de seguridad en WSN se pueden clasificar en las siguientes categorías [44]:

- Ataques internos o externos: los ataques internos provienen de un nodo legítimo de la red, que puede haber sido manipulado para interrumpir el funcionamiento de la red o explotar vulnerabilidades; sin embargo, un ataque externo proviene de nodos que no pertenecen a la red, aunque no pueden tener acceso a la información cifrada.
- Ataques activos o pasivos: los ataques activos implican alguna modificación del flujo de paquetes o la inyección de datos falsos; por otro lado, los ataques pasivos incluyen el espionaje o la observación de los paquetes de datos.
- Ataques por tipo de dispositivo, nodo o portátil: en los ataques por nodos, el enemigo utiliza un nodo con características similares a los nodos de la red, para realizar el ataque; en los ataques por portátil, se utilizan dispositivos con mayor capacidad computacional para causar daño a la red.

Un sistema o red informática debe estar siempre disponible, y debe garantizar la confidencialidad e integridad de los datos que maneja, además de requerir algún tipo de autenticación para que un usuario pueda acceder a la información; sin embargo, estos requerimientos de seguridad pueden ser vulnerados a través de diferentes ataques. Dependiendo del tipo de requerimiento de seguridad que se va a vulnerar, los ataques pueden ser de los siguientes tipos [55]:

- Ataque de interrupción: es un ataque contra la disponibilidad de la red, suele ser un ataque DoS.
- Ataque de interceptación: es un ataque a la confidencialidad de la red en el que se busca comprometer un nodo de la red para lograr acceso no autorizado a los datos.
- Ataque de modificación: es un ataque contra la integridad de la red en el que se modifican los datos.
- Ataque de fabricación: es un ataque a la autenticación de la red en el que el enemigo inyecta información falsa en los paquetes de datos.

Los ataques pueden tener como objetivo un dispositivo computacional (como un ordenador o un servidor, que se denominan host), o los dispositivos de red (incluyen los router, switch o el canal de comunicación cableado o inalámbrico). Atendiendo al tipo de dispositivo que el ataque tiene como objetivo, se pueden clasificar los ataques de la siguiente forma [56]:

- Ataques basados en host: se pueden realizar de tres formas diferentes, comprometiendo el software utilizado en los nodos, manipulando el hardware de los nodos o engañando a los usuarios para que revelen información sensible.
- Ataques basados en la red: se pueden realizar a través de una capa de la pila o de un protocolo de enrutamiento.

La forma más utilizada para clasificar los ataques en una red de sensores inalámbrica distingue los ataques basados en cada capa de la arquitectura de la pila. A continuación, se detallan estos ataques específicamente para cada capa y las medidas de defensa que se pueden aplicar [22, 29]:

- Capa física: esta capa es vulnerable a los siguientes ataques:
  - *Tampering*: el atacante puede reprogramar el nodo, comprometer los datos almacenados o destruir físicamente el nodo.
  - *Jamming*: el atacante envía señales que interfieren con las frecuencias de radio utilizadas por la red, lo que puede ocasionar que los nodos no sean capaces de enviar o recibir datos.
  - Espionaje y análisis de tráfico: la transmisión de los datos en una WSN es inalámbrica, por lo que un atacante podría escuchar las transmisiones utilizando antenas especializadas.

  La medida más eficiente que se puede aplicar para evitar estos ataques es el cifrado de datos, ya sea cifrar directamente los datos o cifrar los paquetes que son transmitidos por el canal de comunicación. Otras soluciones que se pueden considerar son cajas para proteger los nodos o realizar la comunicación a través de una canal de espectro ancho, pero estas pueden resultar costosas y generar mayores gastos de energía.
- Capa de enlace: los ataques que puede sufrir esta capa son:
  - Intrusiones al protocolo MAC (Media Access Control): este protocolo se encarga de asegurar el uso compartido del canal de comunicación. Si el atacante logra que un nodo envíe datos mientras otro nodo debe ser el que esté enviando los datos, puede producir colisiones de paquetes o producirse una denegación de servicio.
  - MAC spoofing: este ataque consiste en la falsificación de la dirección MAC de la tarjeta de red, para que un nodo malicioso pueda acceder a la red y enmascararse como un nodo legítimo.

Una solución para evitar estos ataques es asociar la MAC a una clave secreta, para evitar que la atacante pueda usar la dirección MAC en un nodo falso sin tener la clave asociada.

- Capa de red: los ataques que pueden vulnerar esta capa son:
  - Ataques de manipulación de enrutamiento: la información de las rutas utilizadas para el envío de paquetes en la red pueden manipularse a través de ataques como sinkhole (la información de la tabla de enrutamiento es manipulada para que todo el tráfico de red sea enviado a un nodo manipulado por el atacante), wormhole (conocido como agujero de gusano, crea una ruta directa entre dos nodos lejanos a través de un nodo malicioso), confirmación spoofing (un nodo malicioso envía una confirmación de recibido al nodo emisor haciéndose pasar por el nodo receptor), ataques de inundación HELLO (el nodo malicioso envía mensajes de HELLO a los nodos para que lo identifiquen como un nodo vecino), o de reenvío selectivo (un nodo maliciosos descarta de manera selectiva algunos paquetes).
  - Desbordamiento de la tabla de enrutamiento: el envío constante de la información de enrutamiento de la red puede provocar que las tablas de enrutamiento se desborden.

  Para evitar estos ataques se pueden utilizar técnicas como el uso de múltiples rutas para la transmisión de los datos, adjuntar un código de autenticación de mensaje a cada paquete para determinar que no ha sido alterada la información de la ruta, sincronizar el reloj de los nodos para estimar la distancia de recorrido del paquete, o utilizar protocolos de enrutamiento geográfico.

- Capa de transporte: esta capa puede ser vulnerable a ataques como:
  - Denegación de servicio: la capa de transporte se encarga de la información sobre el estado de las conexiones activas; si un atacante genera un gran número de conexiones activas en un corto periodo de tiempo puede agotar la memoria de los nodos.
  - Desincronización de la conexión: el enemigo puede enviar mensajes falsos a uno o ambos nodos implicados en una conexión para solicitar la retransmisión de paquetes perdidos, afectando a la memoria y energía de los nodos.

Una posible defensa a estos ataques es la autenticación de todos los paquetes que se transmiten durante la conexión. Otra solución para los ataques DoS es obligar a los nodos a dedicarse únicamente a una conexión iniciada.

- Capa de aplicación: esta capa es vulnerable a los siguientes ataques:
  - Ataques al proceso de agregación de datos: el atacante puede modificar los datos antes de que se envíen a la estación base para generar la interrupción del servicio de agregación de datos.
  - Desincronización del reloj: para provocar que los nodos tengan que reajustar sus relojes el atacante puede iniciar el envío de paquetes maliciosos por la red.
  - Reenvío selectivo: el atacante reenvía los mensajes recibidos basados en el contenido de los paquetes.

  Estos ataques afectan la integridad de los datos, por lo que se deben proteger a través de la autenticación; en el ataque de reenvío selectivo se puede utilizar esquemas de cifrado que oculten los datos.

## 2.2. Malware y modelos matemáticos

Un código o software malicioso, comúnmente abreviado como malware, es un programa o código fuente diseñado para realizar funciones de intrusión en sistemas, evadir políticas de seguridad y dañar o causar mal funcionamiento en un sistema. En los últimos años, se han desarrollado diferentes malware que pueden afectar a cualquier plataforma informática y sistema operativo, lo que hace de estos programas una amenaza latente para los ordenadores, dispositivos móviles, archivos y especialmente para los datos críticos e información personal. Un malware puede codificarse y modificarse para realizar diferentes funciones de acuerdo a su objetivo [57].

Los ciberdelincuentes utilizan el malware como arma principal para llevar a cabo ataques a la seguridad, provocando daños y pérdidas significativas a los usuarios de Internet. Los principales objetivos de los delitos cibernéticos son las vulnerabilidades del software, los dispositivos o redes inseguras conectados a Internet, los usuarios y sus datos [58].

Una analogía, entre el código malicioso y los virus biológicos, es que, al igual que la evolución de los mecanismos biológicos, los mecanismos de ataque de código malicioso dependen de la recopilación de información a lo largo del tiempo. Sin embargo, la velocidad del flujo de información en Internet es extremadamente más rápida que los métodos biológicos, por lo que esta amenaza de seguridad está en constante cambio [59].

El malware puede clasificarse, según su forma de propagación, en varios tipos que se detallan en la Tabla 2.1. Las características del malware que se han tenido en cuenta para esta clasificación de los diferentes tipos de malware han sido las siguientes [60]:

- Autorreplicación: el malware intenta propagarse creando nuevas copias o instancias de sí mismo en el host infectado.
- Crecimiento demográfico: describe el número de instancias de malware creadas debido a la autorreplicación.
- Malware parásito: en este caso, el malware requiere de algún código ejecutable para existir.

Tabla 2.1 Características de los tipos de malware.

| Tipos de malware | Características | | |
|---|---|---|---|
| | Autorreplicación | Crecimiento demográfico | Parásito |
| Virus | Sí | Positivo | Sí |
| Gusanos | Sí | Positivo | No |
| Troyanos | No | Cero | Sí |
| Backdoors | No | Cero | Posible |
| Bombas lógicas | No | Cero | Posible |
| Spyware | No | Cero | No |
| Adware | No | Cero | No |
| Bots (Botnets) | No | Cero | Posible |
| Rootkits/Bootkits | No | Cero | Posible |
| Ransomware | Sí | Positivo | Posible |
| Spam and Phishing e-mails | No | Cero | Posible |
| Rabbit | Sí | Positivo | No |
| Mobile malware | No | Cero | Posible |

### 2.2.1. Tipos de malware y sus características

Los malware que han sido estudiados y se ha comprobado que pueden atacar a las WSN son los virus informáticos [61], los gusanos informáticos [62], los troyanos [63], backdoors [64], spyware [65], botnet [66], rootkits [67] y spam [68].

Además de estos, existen otros tipos de malware, que no se ha comprobado que puedan atacar a las WSN: las bombas lógicas, el sdware y el ransomware. Sin embargo, como en los casos anteriores, existe la posibilidad de que los atacantes busquen la forma de adaptar estos tipos de malware para las WSN, tal es el caso del ransomware que ha sido adaptado para las redes IoT, recibiendo el nombre de Ransomware of Things (RoT) [69].

Se describen a continuación estos tipos de malware y sus características.

### Virus informáticos

Existe una gran variedad de definiciones para virus informáticos, una de las más destacadas corresponde a Cohen en 1987 [70], que lo define como un programa que infecta otros programas para obtener una posible versión alterada de sí mismo, otra definición es que se trata de un programa creado con la función e intención de copiarse y propagarse sin el conocimiento y la cooperación del propietario o usuario del ordenador [59]. El objetivo general de un virus es modificar el comportamiento del dispositivo en el que reside, sin que el usuario se dé cuenta [71].

La analogía que existe entre el virus computacional y el virus biológico viene dada por sus características: la capacidad de producir enfermedades (causar daño) en las personas y que actúan por sí solos; es decir, se reproducen y por lo tanto se propagan (contagian) [59]. Como cualquier virus, los virus informáticos necesitan de un anfitrión o huésped donde alojarse, y este puede ser muy variable: un archivo ejecutable, dispositivos de almacenamiento externos, archivos de programas, el sector de arranque o incluso la memoria del ordenador [57].

Algunos virus son inofensivos, otros pueden causar daños en el software, hardware o en la información almacenada; sin embargo, todos ellos dependen de los recursos del dispositivo infectado, y pueden realizar las acciones aprovechando alguna vulnerabilidad o utilizando permisos del sistema. El ciclo de vida de un virus consta de cinco fases: contagio, incubación, reproducción, ataque y defensa [71].

### Gusanos informáticos

Los gusanos informáticos se definen como un código malicioso que se propaga por la red, con o sin ayuda humana. Los gusanos son programas que se autopropagan a través de la red explotando las políticas de seguridad de los servicios utilizados. Poseen la capacidad de autorreplicarse a sí mismos, sin dañar o modificar otros ficheros. Los gusanos utilizan la ingeniería social para engañar a los usuarios y explotar las vulnerabilidades de los dispositivos; además, pueden crear puertas traseras para que el atacante pueda tomar el control remoto del dispositivo infectado.

El ciclo de vida de un gusano tiene cuatro fases: latencia, reproducción, descomposición y persistencia. Los gusanos pueden clasificarse según su mecanismo de propagación en gusanos de correo electrónico u otras aplicaciones cliente, gusanos de archivos compartidos de Windows, y gusanos tradicionales; también pueden clasificarse según las técnicas de descubrimiento de objetivos en gusano de escaneo, lista de objetivos pre-generados, listas de objetivos generadas internamente y de vigilancia pasiva [72].

### Troyanos

Los troyanos son programas que permanecen ocultos en el dispositivo (como en la leyenda del caballo de Troya), pero que al abrir el archivo que lo contiene, ejecutable o no ejecutable, son capaces de capturar datos que son enviados a otro dispositivo, o permiten el acceso remoto de un atacante, a través de una puerta trasera. El ciclo de vida de un troyano está dividido en las siguientes fases: introducción en la víctima, activación, realización de acciones maliciosas y autodestrucción.

Los troyanos suelen aparecer como códecs de audio o vídeo, un plugin de navegador web, un juego, una aplicación pirata, un instalador de imágenes de disco ISO para un sistema operativo o un documento. No tienen la capacidad de autorreplicarse; sin embargo, al finalizar su objetivo pueden autodestruirse, para no dejar rastrs en el equipo infectado [57, 71].

### Backdoors

Los backdoors o puertas traseras son programas que se introducen en un dispositivo de manera encubierta, son troyanos de acceso remoto que pueden estar en un código legítimo o ser programas independientes. Al ejecutarse permiten el uso de una contraseña o evitan el proceso de autenticación, para tomar el control remoto del dispositivo infectado; esto da acceso a carpetas, archivos, documentos, recursos compartidos, así como a otros equipos y servidores de la red.

Según la forma de infectar, las backdoors se pueden clasificar en dos tipos, el primer tipo es similar a los troyanos, ya que son insertados dentro de otro programa, que al ser ejecutado infecta el dispositivo, donde se instala permanentemente. El segundo tipo, es similar a los gusanos, ya que son transportadas dentro de un gusano [60].

### Spyware

El spyware se encarga de recopilar información acerca de lo que un usuario hace en su dispositivo, esté o no conectado a Internet, para luego enviar esa información a los atacantes. La información que el spyware recopila puede incluir cualquier dato que tenga un valor potencialmente importante como nombres de usuario y contraseñas, direcciones de correo electrónico, números de cuenta bancaria y tarjetas de crédito o claves de licencia de software, para facilitar la piratería de software.

El spyware puede llegar a una máquina de varias formas, mediante un virus, un troyano propagado por correo electrónico, con otro software que el usuario instala o aprovechando fallos técnicos en los navegadores web. Este último método hace que el spyware se instale simplemente visitando una página web, y a veces se denomina descarga directa.

La detección de un spyware no suele ser sencilla, ya que utiliza técnicas avanzadas para convivir con el resto de programas y sin que el usuario se dé cuenta; por ejemplo, no son detectados por los antivirus, utilizan sistemas de camuflaje casi perfectos, como el servicio de clientes P2P o utilidades del disco duro, e incluso puede ocultarse en software legales; además, el nombre de los ficheros puede pasar desapercibido y no provocan ningún efecto visible en el dispositivo [73].

### Botnet

Una botnet es un conjunto o red de robots informáticos o bots que se ejecutan de manera autónoma y automática, para que trabajen de forma conjunta y distribuida. Las bots suelen estar a la venta en la web profunda, normalmente utilizan la conexión a Internet del equipo infectado para lanzar un ataque DDoS o de denegación de servicio distribuido, prolongado, a una empresa o sitio web. Todo lo que el usuario del ordenador infectado notará es una desaceleración en su velocidad de Internet.

El artífice de la botnet puede controlar todos los ordenadores/servidores infectados de forma remota denominados robots o zombis. Las redes de bots pueden ser una manera efectiva para que los delincuentes realicen actividades fraudulentas en empresas, o ataquen las infraestructuras críticas de un país. Además, pueden incluir registradores de teclado y software de control que ofrece puertas traseras de los dispositivos infectados. También se puede utilizar una red de bots para investigaciones científicas, existiendo botnets legales e ilegales. El ciclo de vida de una botnet cuenta con las fases de infección, comando y control, compra de la botnet por parte de terceros para ejecutar acciones maliciosas, y ejecución del ataque distribuido [74].

### Rootkits

El rootkit se instala en las particiones de arranque en un ordenador y se ha diseñado para ocultar objetos como procesos, archivos o entradas del Registro del sistema. El rootkit inicia el sistema operativo en un entorno de tipo hipervisor (o monitor de máquina virtual que permite ejecutar varios sistemas operativos al mismo tiempo), lo que le dará un control total del sistema operativo, mientras que al mismo tiempo se oculta y protege desde cualquier software de seguridad que tenga instalado, o cualquier característica de seguridad en el propio sistema operativo.

Un rootkit suele explotar las características del sistema operativo, como la capacidad de instalar extensiones en una aplicación, conectar o parchear una interfaz de programación de aplicaciones que contiene una vulnerabilidad explotable. Una vez ejecutado, el rootkit puede

obtener el control del flujo de ejecución de la aplicación, usar sus permisos y privilegios para atacar los sistemas de seguridad y arranque del ordenador. Los rootkits y bootkits pueden ser extremadamente difíciles de eliminar y detectar, dado que residen en particiones ocultas y protegidas del disco duro [75].

### Spam y phishing

Spam son todos los correos que se reciben sin ser solicitados y que han sido enviados de forma masiva. No es código malicioso, pero generalmente sirve como medio de propagación de malware. El phishing o redireccionamiento, se envía por lo general a través del spam, suelen ser comunicados de bancos o tiendas online para el cambio de contraseña o instrucciones para confirmar el número de tarjeta de crédito, que al seguir el link recibido redirigen al usuario a una página falsa que es copia de la original, donde los datos personales son capturados, para posteriormente ser utilizados para fines maliciosos [71].

### Bombas lógicas

Las bombas lógicas es un software que se activa después de un tiempo predeterminado o en una fecha concreta, esto permite que el usuario no relacione el problema en el dispositivo con la instalación de un programa en concreto. Pueden ser insertadas en el código de otro programa o ser independientes.

Una bomba lógica es un código que consta de dos partes: la carga útil, es decir, la acción a realizar, que puede ser cualquiera, pero con un efecto malicioso; y un disparador, que es una condición booleana que se evalúa y controla cuándo se ejecuta la carga útil. Los disparadores pueden ser diseñados para ser ejecutados remotamente o ser activados por la ausencia de un evento.

Una bomba lógica puede realizar diferentes acciones en el dispositivo, desde el consumo excesivo del procesador, la destrucción de ficheros, violaciones de seguridad, borrado de memoria CMOS o destrucción de los cabezales de las impresoras [57].

### Adware

El adware tiene similitudes con el spyware ya que ambos pueden recopilar información sobre el usuario. Sin embargo, esta más centrado en el marketing, se puede desplegar como anuncios o redirigir el navegador web de un usuario a ciertos sitios web con la esperanza de hacer una venta. El anuncio suele adaptarse a las preferencias del usuario, y se muestran en forma de ventanas emergentes, en un navegador o por separado. Un adware puede recopilar

y transmitir información sobre los usuarios que puede ser utilizada para fines de marketing, y no es capaz de replicarse automáticamente [60].

### Ransomware

Este malware cifra los archivos y documentos para exigir el pago por un rescate, generalmente en bitcoins. El ransomware no solo es capaz de cifrar copias de seguridad de ficheros, sino que también puede extenderse a otros dispositivos en la red y emplear los permisos de acceso del usuario en esos dispositivos para acceder a más áreas de almacenamiento. Algunos ransomware incluso pueden cifrar el disco duro entero de un ordenador.

Generalmente, la clave de descifrado, si es pagado el rescate, puede contener malware adicional. Sin embargo, los precios de los rescates no son demasiado altos, de esta forma los atacantes consiguen que los usuarios paguen el rescate para recuperar su información.

El ciclo de vida de un ransomware incluye las siguientes fases: distribución, infección, comunicación, búsqueda de archivos, cifrado de archivos y exigencia del rescate [76].

### Rabbit

Rabbit es el término utilizado para describir el malware que se multiplica rápidamente. Existen dos tipos, el primero es un programa que intenta consumir todos los recursos del sistema, como el espacio en disco, por ejemplo, y crea nuevos procesos en un bucle infinito. Este malware tiende a dejar rastros obvios que apuntan hacia el atacante.

El segundo tipo de rabbit es un caso especial de gusano, es un programa autónomo que se replica a través de una red de máquina a máquina, pero elimina la copia original de sí mismo después de la replicación. En otras palabras, solo hay una copia de un rabbit en una red, es decir que solo salta de un ordenador a otro [60].

### Mobile malware

El Mobile malware tiene como objetivo principal los dispositivos móviles, como los teléfonos inteligentes y tabletas. Los vectores de propagación de este malware son conexiones Bluetooth, mensajes multimedia, páginas web, mensajes de texto y a través de tarjetas de memoria externa. Los posibles riesgos a los que nos enfrentamos con este tipo de malware son la perdida de dinero o información privada y los problemas asociados a la exposición de datos privados de una empresa. Si el dispositivo deja de funcionar se podrían perder todos los datos guardados, incluyendo fotos personales, contactos y correos electrónicos [77].

### 2.2.2. Modelos matemáticos de propagación de agentes biológicos

Los modelos científicos de simulación de sistemas reales pueden ser de dos tipos: formales y no formales. Dentro de los modelos formales, se encuentran los modelos matemáticos y los modelos computacionales. Por otra parte, los modelos no formales son normalmente verbales; es decir, que expresan de manera oral o escrita la relación entre variables. Para determinar un modelo es necesario seguir una serie de pasos que definen el denominado ciclo de modelización (ver Figura 2.5), enumerados a continuación [78]:

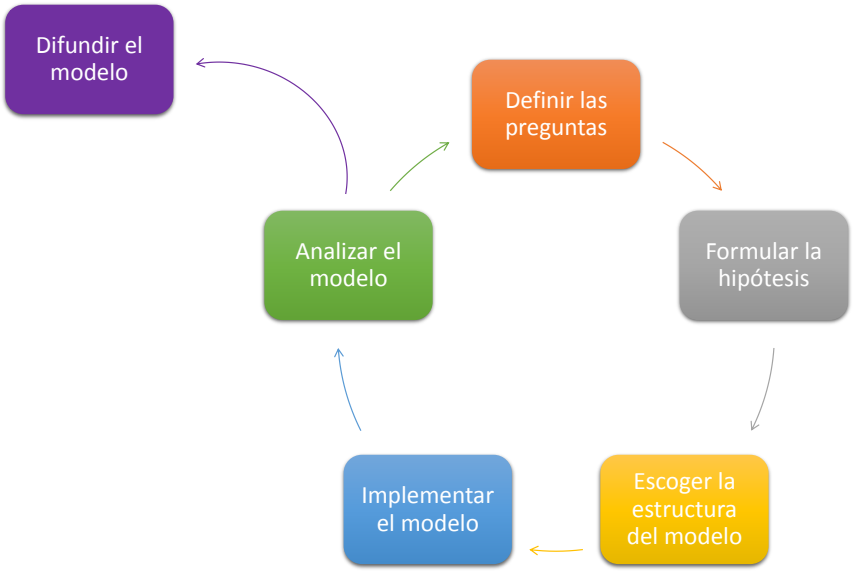



Figura 2.5 Ciclo de modelización

1. Definir las preguntas: el proceso del modelo inicia con preguntas claras y concretas sobre la investigación, ya que todo el ciclo se desarrollará en base a estas preguntas.

2. Formular la hipótesis: permite dar el enfoque correcto al modelo, pues se desea comprobar que la hipótesis es verdadera o falsa.

3. Escoger la estructura del modelo: en base a las preguntas y la hipótesis formuladas, se definen los procesos, parámetros y variables que se utilizarán en el modelo.

4. Implementar el modelo: en esta fase del modelo se utiliza un software especializado para transcribir el modelo verbal y matemático en un modelo computacional gráfico.

5. Analizar el modelo: al llegar a esta fase, se analiza, prueba y revisa que el modelo cumple adecuadamente con su propósito, en caso contrario se debe reiniciar el ciclo y comprobar en cada una de las fases los posibles fallos y corregirlos hasta que obtener el resultado esperado.

6. Difundir el modelo: cuando el modelo funciona correctamente, se pueden comenzar a difundir los resultados obtenidos.

El término simulación suele estar relacionado a dos conceptos distintos, por una parte hace referencia a la metodología del uso de las técnicas de simulación para resolver un problema específico [30], y por otra, se refiere a la simulación como el acto de ejecutar en tiempo real un modelo de simulación. Además, el concepto de simulación computacional se refiere a un método informático que se utiliza para construir los modelos computacionales mediante un lenguaje de programación [79].

Tal como se detalla en la Figura 2.6, la infraestructura de simulación está compuesta por el hardware y el software necesarios para ejecutar la simulación del modelo.

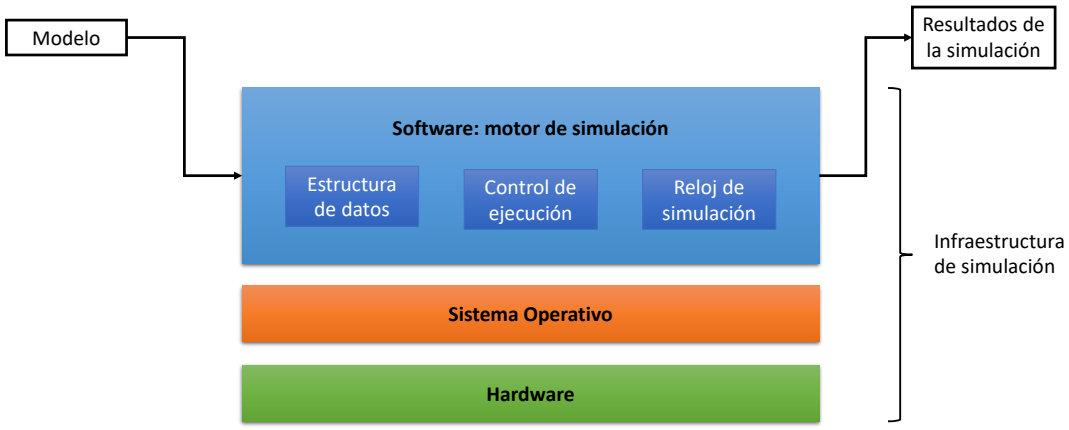



Figura 2.6 Infraestructura de simulación [30].

Finalmente, la diferencia entre los términos modelizar y simular es que el primero se refiere a la construcción de modelos en general, y la simulación consiste en desarrollar modelos computacionales [79].

Como ya se ha mencionado en la Sección 1, el estudio de los modelos matemáticos de propagación de malware tiene su origen en la epidemiología matemática. Una infección puede definirse como la invasión de un individuo por uno o más organismos pequeños. Según

[80], existen diversos tipos de agentes infecciosos y muchas maneras en las que se pueden transmitir. Durante una infección, se pueden distinguir tres periodos de tiempo:

1. Pre-infección: corresponde al tiempo transcurrido desde la infección hasta el momento en que un huésped es capaz de transmitir el agente infeccioso a otro huésped.

2. Incubación: es el tiempo transcurrido desde la infección hasta el inicio de la enfermedad clínica.

3. Periodo infeccioso: es el tiempo en el que el huésped es capaz de transmitir la infección, este periodo esta comprendido entre el final del periodo pre-infeccioso y el momento en que un huésped deja de ser infeccioso.

Cuando un individuo está expuesto a un virus y se infecta, se habla en general, de tres tipos de resultados inmunitarios de la infección: la inmunidad sólida, que se da cuando los individuos ya no pueden infectarse; los individuos que pueden librarse de la infección pero pueden permanecer susceptibles en cierta medida a nuevas infecciones; y el individuo que desarrolla poca o ninguna inmunidad y puede ser infectado e infeccioso de por vida [80].

La transmisibilidad de la infección o capacidad de pasar de un huésped a otro, está relacionada con la tasa de ataques secundarios; esta tasa se define como el número de individuos susceptibles que pueden ser infectados por un caso primario, es decir, el individuo que ha traído la infección a la población. Para esto, se definen dos medidas de transmisibilidad: el número de reproducción básica ($R_0$) y el número de reproducción efectiva ($R_n$) [81].

El número de reproducción básica se define como el número medio de personas infectadas, resultantes de la introducción de una persona contagiosa en una población totalmente susceptible. Por otro lado, el número de reproducción efectiva se define como el número promedio de personas secundarias infectadas, que resultan de cada persona infecciosa en una población dada, en la que algunos individuos pueden ser inmunes, debido a una infección previa.

La propagación de agentes biológicos se estudia a través de la epidemiología matemática, teniendo en cuenta las diferentes características de la enfermedad, que permiten analizar la forma de transmisión de un organismo a otro.

La clasificación de los modelos matemáticos en estocásticos y deterministas puede generar desconcierto porque algunos modelos deterministas incorporan elementos estocásticos y viceversa. Los modelos estocásticos tienden a utilizarse cuando se modeliza la transmisión en poblaciones pequeñas o cuando es importante proporcionar estimaciones del rango en el que podría ocurrir un resultado, como por ejemplo, el número de casos; debido a que las variables y parámetros utilizados en estos modelos son aleatorios [82].

Otra clasificación de los modelos puede ser en continuos y discretos. Las variables de los modelos continuos adquieren un valor infinito dentro de un rango determinado, por el contrario, en los modelos discretos los valores de las variables son finitos. Además, los modelos pueden ser globales e individuales, donde los modelos globales consideran un sistema complejo como un todo y no se consideran las interacciones entre los elementos que forman el sistema; sin embargo, los modelos individuales son capaces de considerar dichas interacciones de los elementos y sus características individuales [83].

Existen diferentes modelos matemáticos de propagación de agentes biológicos. A continuación, se describen algunos de los modelos más utilizados y las características de cada uno de ellos [84]:

1. Modelo compartimental: los individuos de la población se subdividen en amplios subgrupos, el modelo rastrea el proceso de infección para estos individuos colectivamente. Estos modelos pueden ser deterministas o estocásticos.

2. Modelo individual o modelo de micro-simulación: este modelo rastrea el proceso de infección para cada individuo en la población. Además, muchos modelos individuales también son estocásticos.

3. Modelo dinámico de transmisión: este modelo incorpora el contacto entre individuos, por lo que el riesgo de infección depende del número de individuos contagiosos en la población y, por lo tanto, variará con el tiempo si el número de individuos infectados cambia.

4. Modelo estático: este modelo no describe explícitamente el contacto entre individuos. Por lo tanto, el riesgo de infección se establece con valores predeterminados. Estos modelos se utilizan habitualmente cuando se conoce el riesgo de infección, como fue el caso de la tuberculosis en algunos entornos occidentales durante el siglo XX. Sin embargo, los modelos estáticos no son fiables para analizar el efecto de las intervenciones que implican reducciones en la prevalencia de individuos infecciosos, como puede ser la vacunación o los tratamientos, que pueden reducir el riesgo de infección.

5. Modelo en red: es un modelo en el que la red de contacto entre individuos está explícitamente modelizada. El riesgo de infección para un individuo depende de la persona a la que está conectado. Estos modelos se han utilizado para describir las enfermedades de transmisión sexual.

El primer modelo compartimental fue propuesto por Kermack y McKendrick [8], en 1927, para simular la propagación de la peste bubónica ocurrida en Londres entre 1665 y

1666, que produjo la muerte del 20 % de la población. Este modelo divide la población en tres grupos: Susceptibles $S(t)$, Infectados $I(t)$ y Recuperados $R(t)$. Se considera que la población es constante en cualquier instante de tiempo $t$, es decir, $S(t)+I(t)+R(t)=N$, donde $N$ es el número total de la población. Además, se tienen en cuenta la tasa de transmisión $a$ y la tasa de recuperación $b$. Este modelo se describe mediante el siguiente sistema de ecuaciones diferenciales ordinarias [83]:

$$\begin{aligned} S'(t) &= -\frac{a}{N}\cdot S(t)\cdot I(t) \\ I'(t) &= \frac{a}{N}\cdot S(t)\cdot I(t) - b\cdot I(t) \\ R'(t) &= b\cdot I(t) \end{aligned}$$

A partir de entonces, la mayoría de los modelos deterministas propuestos basados en ecuaciones diferenciales ordinarias tienen como base este modelo de Kermack y McKendrick. La Figura 2.7 muestra las estructuras compartimentales más utilizadas para estudiar la transmisión de enfermedades.

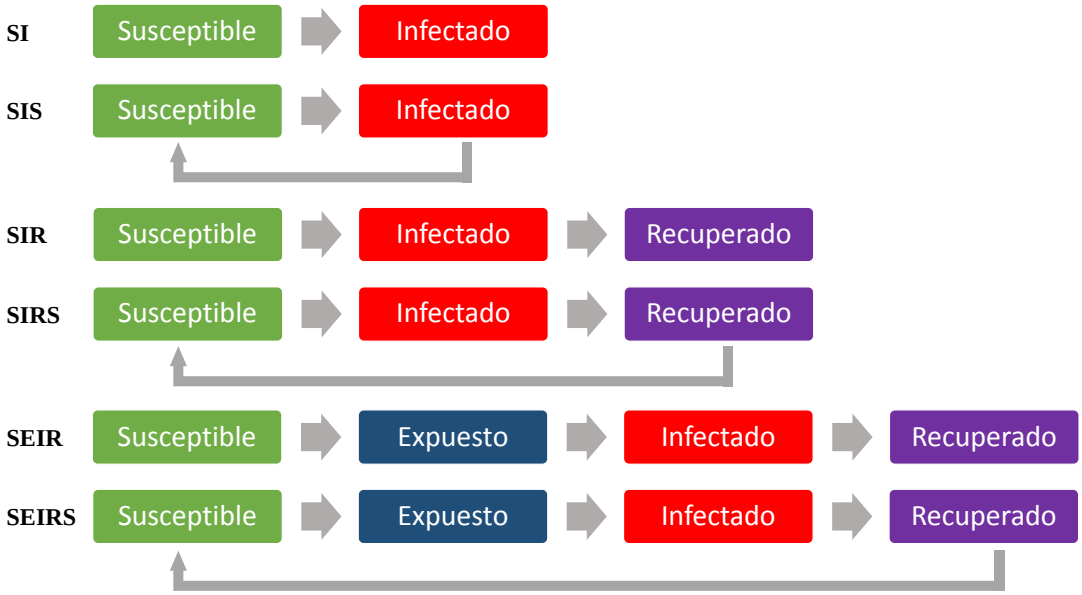



Figura 2.7 Estructuras utilizadas para la modelización de enfermedades infecciosas.

En los modelos matemáticos se pueden definir coeficientes para agrupar las características de los individuos y el comportamiento del sistema con el entorno. En un sistema complejo,

los individuos no controlan el sistema, sin embargo, el comportamiento del sistema es el resultado de las interacciones de los individuos [30].

### 2.2.3. Modelos matemáticos de propagación de malware

Los modelos matemáticos propuestos para estudiar la propagación del malware en las WSN pueden ser modelos globales o individuales. Además, estos modelos pueden ser clasificados en función del compartimento, el tipo de modelo (continuo o discreto, determinístico o estocástico), y la herramienta matemática utilizada, como las ecuaciones en derivadas parciales (EDP), ecuaciones diferenciales ordinarias (EDO), autómatas celulares (AC) o cadenas de Markov. La Tabla 2.2 incluye un estudio detallado de los modelos matemáticos que están siendo utilizados para modelizar la propagación de malware en redes de sensores inalámbricas.

Tabla 2.2 Clasificación de los modelos matemáticos.

| **Trabajo** | **Compatimento** | **Tipo de modelo** | | | **Herramienta** |
|---|---|---|---|---|---|
| [85] | SEIR-V | Global | Continuo | Determinístico | EDO |
| [86] | SIR | Individual | Discreto | Estocástico | AC |
| [87] | SI | Global | Discreto | Estocástico | Cadena de Markov |
| [88] | SIRC, SIQRC, SIRVC | Global | Continuo | Determinístico | EDO |
| [89] | SEIR | Global | Continuo | Determinístico | EDO |
| [90] | SIRS | Global | Continuo | Determinístico | EDO |
| [91] | SIR | Individual | Discreto | Determinístico | AC |
| [92] | SIRS | Global | Continuo | Determinístico | EDO |
| [93] | SIR | Global | Continuo | Determinístico | EDP |
| [94] | SIR | Global | Discreto | Estocástico | Cadena de Markov |
| [20] | SIR | Individual | Discreto | Determinístico | AC |
| [95] | SIS | Global | Discreto | Estocástico | Cadena de Markov |
| [96] | SI | Individual | Discreto | Estocástico | Cadena de Markov |
| [97] | SEIR | Individual | Discreto | Determinístico | AC |
| [12] | SIRD | Individual | Discreto | Determinístico | AC |
| [9] | SI | Global | Discreto | Determinístico | EDO |
| [11] | SEIR-V | Individual | Continuo | Determinístico | EDO y MBA |
| [10] | SIRD | Global | Discreto | Estocástico | Cadena de Markov |
| [66] | SIS | Global | Discreto | Determinístico | EDO |
| [98] | SNIRD | Global | Discreto | Estocástico | EDP |

En los últimos años, se han propuesto varios modelos globales para modelizar la propagación del malware en las WSN (véase [99] y las referencias correspondientes). En [93], por ejemplo, se describe un modelo SIR que incluye un retardo discreto para predecir eficazmente tanto el comportamiento dinámico temporal, como la distribución espacial de la propaga-

ción del malware en WSN de tipo móvil. Esta propuesta es un modelo global, continuo y determinista que ha utilizado EDP.

En [90] se ha propuesto un modelo SIRS mejorado, para caracterizar el proceso de propagación de los gusanos basado en tres factores: el consumo de energía, el radio de comunicación y la densidad distribuida de los nodos. Este modelo se basa en un EDO. Además, es un modelo global, continuo y determinista.

En [95] los autores han propuesto un modelo SIS heterogéneo de tiempo discreto. Esta propuesta, predice el comportamiento de la infección del malware desarrollando un juego no cooperativo de suma no cero, para describir las interacciones entre un sistema heterogéneo de WSN y el malware. Este modelo se basa en una cadena de Markov; es por lo tanto, es un modelo global, discreto y estocástico.

Otros autores han estudiado la propagación del malware en redes WANET (*Wireless Ad Hoc Network*) [9]. El objetivo en dicho trabajo ha sido identificar la velocidad de propagación del malware a través de dos esquemas de difusión (*unicast* y *broadcast*) y dos modos de red (difusión y comunicación), basados en el modelo clásico de propagación epidemiológico SI. Se trata de un modelo global, discreto y determinista, que ha utilizado EDO.

El modelo SIRD propuesto en [10] ha evaluado la propagación de malware en el IoT de banda estrecha de las WSN heterogéneas basados en objetos (NBIOT-HWSN). Se ha analizado la disponibilidad de nodos basados en la distribución de nodos heterogéneos y vulnerabilidades. Este modelo ha utilizado las cadenas de Markov y es un modelo global, discreto y estocástico.

Los autores de [66] propusieron un modelo de propagación de botnets IoT-SIS basado en redes de sensores inalámbricas. Analizaron el impacto del consumo de la energía de los dispositivos y la densidad de la red frente a los diferentes métodos de exploración de las botnets. Este modelo se describe con EDO; además, es un modelo global, discreto y determinista.

En [98] se propuso un modelo heterogéneo que consideraba los estados susceptible, insidioso, infeccioso, recuperado, disfuncional (SNIRD) en la WSN. Este modelo incluyó el estado insidioso ($N$) para aquellos sensores infectados que no han sido detectados por el sistema de detección de intrusos (IDS), y el estado disfuncional ($D$), que se refiere a los nodos que han dejado de funcionar porque han sido destruidos por la acción del malware, el agotamiento de la energía o por haber sufrido daños físicos. Este modelo utilizó ecuaciones en derivadas parciales, y es un modelo global, discreto y estocástico.

Posteriormente, se han tenido en cuenta las características específicas de las WSN para desarrollar modelos individuales, por ejemplo, en [96], los autores propusieron un modelo que sigue el esquema de transición de estados de un modelo típico de infección SI, pero que

puede calcular microscópicamente la probabilidad previa de que cada sensor sea infectado por un gusano, utilizando varias ecuaciones iterativas de estados de seguridad individuales. Esta investigación es un modelo individual, discreto y estocástico, que se basa en una cadena de Markov.

En [20] se propuso un modelo mejorado basado en individuos, que utiliza características particulares de tres tipos de nodos y una topología compleja. Los compartimentos que se consideran en el modelo son los de susceptibles, infectados, recuperados, dañados y fuera de servicio. En este modelo se utilizan autómatas celulares. Además, es un modelo individual, discreto y determinístico.

En [97] se propuso un modelo SEIR para simular la propagación de un virus informático a través de una red informática. Como herramienta matemática se utilizaron los autómatas celulares en grafos. En este modelo, los parámetros que se consideran están relacionados con el ciclo de vida de un virus informático, las contramedidas aplicadas en los hosts y el comportamiento de los usuarios. Además, es un modelo individual, discreto y determinista.

En [12], se planteó un modelo SIRD basado en autómatas celulares bidimensionales (2D). Este modelo consideraba tres aspectos (tasas de infección, inmunidad y mortalidad), en dos tipos diferentes de nodos (cabeceros y sensores), con la finalidad de analizar la propagación del malware en WSN. Además, se construyó un modelo de juego evolutivo multijugador para encontrar la estrategia evolutiva y estable óptima. Se trata en este caso de un modelo individual, discreto y determinista.

El paradigma MBA se ha considerado en el diseño de otros modelos para simular no solo la propagación de agentes biológicos [100, 101, 102], sino también para la propagación de malware [103, 104]. En [103], los autores han introducido un novedoso entorno de simulación basado en agentes para el estudio de malware complejo. Consideran diferentes aplicaciones, estructuras de red, coordinación de redes y movilidad de los dispositivos. Específicamente, la atención se centra en la propagación de malware entre los usuarios de una red celular, considerando como vector de transmisión las conexiones Bluetooth; y un gusano informático híbrido, es decir, que puede tener múltiples vectores de transmisión, particularmente el correo electrónico y el intercambio de archivos a través de la red.

Por otra parte, en [104] se propone un modelo MBA para la propagación de malware basado en el proceso de difusión de rumores. Este modelo considera un conjunto de agentes (dispositivos) heterogéneos e interacciones entre ellos. Se basa en un modelo de red cuya dinámica se rige por un sistema de ecuaciones diferenciales ordinarias.

Los autores en [11] han propuesto un método de análisis y diseño de sistemas dinámicos de ciber-agentes analíticos (A2CDSADM). Este método se ha dividido en dos partes; primeramente, se realiza un análisis del modelo basado en un sistema de ecuaciones diferenciales,

donde se buscan los puntos de equilibrio y se utiliza Matlab para resolver las ecuaciones. Seguidamente, se diseña un modelo SEIR-V basado en agentes, representado a través del lenguaje unificado de modelado (en inglés, UML); este modelo utiliza dos tipos de agentes, los sensores y el malware, además, el entorno es una red de clúster para representar una red de sensores inalámbricas. El objetivo es estudiar la propagación de un gusano en una WSN, la simulación es realizada en Netlogo. Posteriormente, son evaluados los resultados de ambos enfoques, a través de una alineación de modelos y pruebas de equivalencia. Finalmente, los resultados son comparados con modelos de otros autores. En cuanto a la representación matemática, este modelo puede clasificarse como individual, continuo y determinístico, que utiliza un sistema de ecuaciones diferenciales ordinarios.

El modelo que se propone en este trabajo, se ha desarrollado sobre la base del paradigma MBA, que permite asignar características y acciones a cada individuo autónomo, y establecer la interacción entre los individuos dentro de un entorno.

Las características que se han tenido en cuenta con más frecuencia para definir los modelos basado en agentes han sido el tipo de sensor y el malware. Algunos estudios más recientes, han incluido la topología y el medio ambiente o entorno. Además de estas particularidades, el modelo propuesto en esta memoria incluye otras características individuales de los sensores, como la capacidad computacional, la capacidad de transmisión y recepción de información, el ciclo de trabajo y los métodos de recopilación de datos. Por otra parte, se han introducido la acción humana y los dispositivos externos y computacionales, como agentes que influyen en la propagación del malware.

## 2.3. Modelo Basado en Agentes

A finales de la década de los 40, la máquina autorreplicante conocida como máquina de Von Neumann [105], que se modelizaba con autómatas celulares, sirvió de base para el conocido como Juego de la Vida, introducido por el matemático John Conway [106], que operaba con reglas sencillas en un mundo virtual bidimensional, en forma de tablero de juego de damas. Durante los años 70 y 80 se publicaron los primeros MBA teóricos, incluyendo el modelo de segregación de Thomas Schelling [107]. No obstante, no fue hasta los años 90 y 2000, cuando las áreas de aplicación de los MBA comenzaron a crecer rápidamente. Fue entonces cuando se desarrolló el MBA denominado Sugarscape [108], que tenía como finalidad simular y explorar fenómenos sociales complejos como las migraciones estacionales, la contaminación, la reproducción sexual, o la prevención, control y transmisión de enfermedades [13].

Tal como ya se ha comentado anteriormente, un modelo científico es una representación teórica del comportamiento de sistemas, fenómenos o procesos reales que se desean estudiar, analizar o describir, con el fin de encontrar las características más relevantes o simular una situación real. El propósito fundamental de los modelos es permitir un mejor estudio de las propiedades específicas que ha utilizado el sistema original [30].

### 2.3.1. Concepto y estructuras de los modelos basados en agentes

Los modelos basados en agentes son modelos donde los individuos o agentes se definen como entidades únicas y autónomas, que interactúan entre sí y con su entorno local. Un MBA puede estar formado por distintos tipos de agentes que representan diferentes individuos dentro del sistema que se analiza. Generalmente, los MBA son modelos adecuados para sistemas con actores heterogéneos, autónomos y pro-activos en los que no se pueden descuidar las características individuales.

A continuación se detallan las características principales de los elementos que forman un MBA: los agentes, su entorno y las reglas por las que se rige su comportamiento.

#### Agentes

Los agentes son una entidad capaz de realizar acciones autónomas en un entorno concreto para alcanzar su objetivo. Pueden ser plantas, animales, personas, instituciones o cualquier otra entidad. Cada agente es único porque tiene características, como pueden ser el tamaño, la posición o los recursos de que dispone, entre otros, que los diferencia de otros agentes. Se dice que los agentes interactúan localmente porque no interactúan con todos los demás agentes, sino únicamente con sus vecinos, creando así una red. Otra característica de los agentes es que son autónomos; es decir, actúan independientemente para alcanzar su objetivo [30]. Además, los agentes siguen un comportamiento adaptativo para ajustar su comportamiento a su estado actual, o al estado de otro agente y también a su entorno. Un agente suele tener las siguientes características básicas [13]:

- Un agente es un individuo auto-contenido, modular e identificable; cada agente es un objeto discreto en la simulación. Se dice que uno de los requisitos de los agentes es el de modularidad; es decir, que cada agente tiene una frontera.
- Un agente es autónomo y auto-dirigido, es decir, puede funcionar independientemente en su entorno e interactuar con otros agentes.
- Los agentes tienen atributos que les permiten ser distinguidos y reconocidos por otros agentes; los atributos pueden ser estáticos, no modificables durante la simulación

(como el nombre o id), o dinámicos, que pueden modificarse a medida que la simulación progresa (como la edad o etapa del ciclo de vida). Los atributos y datos de los agentes son típicamente almacenados como sus estados internos, los cuales pueden ser representados por variables discretas o continuas.

- Un agente tiene comportamientos que relacionan la información detectada por el agente con sus decisiones y acciones; esta información se obtiene a través de interacciones con otros agentes y con el entorno. El comportamiento de un agente puede especificarse mediante reglas simples o modelos abstractos (como redes neuronales o programas genéticos) que mapean las entradas de los agentes a sus comportamientos.

- Un agente tiene un estado que varía con el tiempo y que además representa las variables esenciales asociadas con la situación actual del agente. El estado consiste en un conjunto o subconjunto de sus atributos que condicionan los comportamientos de un agente; por lo tanto, cuanto más grande sea el conjunto de posibles estados de un agente, más variado será el conjunto de comportamientos que el agente puede tener.

- Un agente es una entidad social que interacciona dinámicamente (mediante comunicación, movimiento, competencia/limitación por el espacio y los recursos, etc.) con otros agentes que influyen en su comportamiento; estas interacciones pueden involucrar protocolos de agentes y dar lugar a fenómenos emergentes auto-organizados, que son observables a escala global.

- Un agente puede adaptarse y modificar su comportamiento basándose en reglas u otros mecanismos abstractos; además, puede tener la capacidad de aprender (lo que requiere alguna forma de memoria). Los modelos basados en agentes sofisticados incorporan algunas veces redes neuronales, algoritmos evolutivos u otras técnicas de aprendizaje que permiten un aprendizaje y adaptación realistas; los agentes también pueden ser capaces de evolucionar, permitiendo que surjan comportamientos no anticipados.

- Un agente puede estar orientado a objetivos con respecto a su comportamiento y ser capaz de ajustar sus respuestas para interacciones futuras.

- Los agentes pueden ser heterogéneos; los MBA a menudo consideran la gama completa de diversidad de agentes en una población; las características y comportamientos de los agentes pueden variar en varios aspectos, incluyendo la extensión, sofisticación, cantidad de información necesaria para la toma de decisiones y extensión de la memoria de eventos pasados.

### Entorno

En los modelos basados en agentes se define el entorno como la representación del mundo virtual simulado en el que existen los agentes. El entorno puede ser concebido como el medio en el que operan los agentes y con el que interactúan. Un entorno típico puede consistir en el conjunto de variables globales u otras estructuras computacionales que definen colectivamente los comportamientos e interacciones de los agentes. Así, el estado de un modelo basado en agentes es el estado colectivo de todos los agentes junto con el estado del entorno.

El entorno en el MBA se describe a menudo por su topología, que define cómo los agentes están conectados entre sí y con su entorno. Las topologías pueden ser tan simples como un modelo abstracto no espacial denominado de "sopa" (en el que los agentes no tienen ubicación), espacios euclídeos (en los que los agentes vagan en espacios bidimensionales, tridimensionales o superiores), etc. o tan complejos como los autómatas celulares (como en el modelo Sugarscape [108]), la cuadrícula espacial o la red de nodos (como el modelo epidemiológico), o el sistema SIG de información geográfica, entre otros.

La topología también puede definir el concepto general del vecindario de un agente en un MBA, en el que los espacios del agente están bien definidos en el modelo. Los vecindarios típicos incluyen la vecindad de Von Neumann y Moore. En una celosía cuadrada bidimensional, el vecindario de Von Neumann comprende las 4 celdas que ortogonalmente rodean a la celda central, y el vecindario de Moore comprende las 8 celdas que rodean la celda central. Los MBA más avanzadas pueden incluir topología y conectividad que pueden cambiar con el tiempo [109].

### Reglas

En un MBA, los comportamientos de los agentes se modelizan por reglas que definen cómo se comportan los agentes en respuesta a los cambios en el entorno y hacia el conjunto de objetivos del agente. Estas reglas también definen un conjunto de relaciones entre agentes, donde cada relación gobierna la forma en que cada agente interactúa y está conectado con otros agentes y con su entorno.

Las reglas son el nexo de unión de los agentes con su entorno. Tanto los agentes como el medio donde se encuentran pueden tener su propio conjunto de reglas, las cuales pueden ser operadas entre el nivel de agente-entorno, entorno-entorno o agente-agente.

Una regla de movimiento a nivel de agente-entorno para el movimiento puede indicar a un agente que mire a su alrededor lo más lejos que pueda, encuentre el sitio con más recursos, vaya allí y tome el recurso.

Una regla de nivel entorno-entorno, para la actualización de recursos, puede tratar el rango de cambio de los recursos en un lugar específico, como una función de los niveles de recursos en espacios vecinos, donde el vecindario está predefinido de acuerdo con algunos criterios dados.

Finalmente, las reglas de nivel agente-agente pueden incluir cómo un agente masculino se relaciona con un agente femenino o cómo un tipo de agentes (por ejemplo, depredador) interactúan con otro tipo (por ejemplo, presa).

Todo lo asociado con un agente es un atributo del agente o una regla del agente que opera en el agente. Las reglas de los agentes incluyen comportamientos que vinculan la situación del agente con sus acciones. Una vez que las variables de estado de un agente están definidas, su comportamiento puede representarse como un autómata o máquina de estados finitos, en la que se produce una transición de estado cuando el agente interactúa con otro agente. Así, todas las interacciones y cambios de estados están regulados por reglas de comportamiento para los agentes y el medio ambiente. Ejemplos de comportamientos de los agentes podrían ser la producción, el consumo de recursos o la comunicación con otros agentes y con el entorno [110].

### 2.3.2. Arquitectura en los modelos basados en agentes

La arquitectura de los agentes en el MBA especifica cómo se puede descomponer la estructura de un agente en la construcción de un conjunto de módulos, y cómo se debe hacer para que estos módulos interactúen entre sí [111]. La estructura interna de un agente consta de tres componentes principales:

- El interfaz del sensor, que permite que el agente sea capaz de percibir el entorno en el que se encuentra.
- El interfaz de acciones, que permite al agente interactuar con el entorno, para alcanzar sus objetivos.
- El razonador, es un componente interno del agente que permite procesar los datos recibidos por los sensores para la toma decisiones.

La arquitectura de un agente define cómo los datos capturados del entorno y el estado interno del agente determinan las futuras acciones del agente y el próximo estado del agente; esta secuencia de acciones también puede denominarse función del agente. Se han propuesto varios enfoques para clasificar la arquitectura de los agentes, en este trabajo se ha seleccionado la clasificación más común, que consiste en separarlas según los dos tipos siguientes de arquitectura [112]:

- Agentes con arquitectura reactiva, también conocidos como agentes reactivos. En este caso el comportamiento de este tipo de agentes está definido por el entorno y las mediciones reales realizadas por el interfaz del sensor. Estos agentes se abstraen de los estados internos y reaccionan directamente a las mediciones de sus sensores.

- Agentes con arquitectura deliberativa, también llamados agentes deliberativos, que hacen un seguimiento de los estados y de las mediciones de la interfaz del sensor, para realizar una acción y establecer un estado interno.

### 2.3.3. Implementación del modelo basado en agentes

El análisis de un fenómeno de interés utilizando los MBA debe reunir ciertas características, como la interacción compleja entre los agentes, las poblaciones heterogéneas, la complejidad en la topología de las interacciones entre los componentes del sistema, los comportamientos complejos y el entorno [113]. Para esto, en [114] se diseñó un árbol de decisiones que permite identificar la herramienta de modelización que mejor se adapta a un fenómeno de interés. Este árbol de decisiones distingue, a partir de una serie de preguntas, con solo dos tipos de respuesta (sí o no), los diferentes tipos de modelización, que incluyen modelos estocásticos, modelos de ecuaciones, redes bayesianas, autómatas celulares y modelos basados en agentes.

Después de verificar que los MBA se adaptan al fenómeno de interés a estudiar, se deben seguir una serie de pasos, que se conocen como el ciclo de modelización de los MBA. Este ciclo es similar al ciclo de modelización tradicional o científico, pero con algunas modificaciones que se detallan a continuación [79]:

- Pregunta de investigación: al igual que en el ciclo de modelización tradicional, es importante formular la pregunta de investigación al inicio del ciclo, en base a esta pregunta se propondrá un modelo para tratar de dar respuestas.

- Diseño: en esta etapa se decide qué elementos del fenómeno estudiado deben incluirse en el modelo.

- Desarrollo: se crea el modelo computacional, basado en el diseño del modelo realizado anteriormente.

- Verificación: se comprueba que el modelo hace exactamente lo que se ha pensado que debería hacer.

- Calibración: se establecen los valores iniciales del modelo.

- Análisis de sensibilidad: se comprueba que el comportamiento y resultado del modelo es sensible a cambios en los valores iniciales.
- Validación: en esta fase se verifica que el comportamiento del modelo representa realmente al fenómeno que se está estudiando.
- Documentación: consiste en describir el modelo detalladamente en un documento. Para ello existen protocolos como ODD (*Overview, Design concept and Details*).
- Publicación: se pueden escribir artículos científicos donde se explique el modelo y sus resultados, sin olvidar los detalles técnicos de la simulación.
- Replicación: es el proceso mediante el cual el modelo que ha sido documentado y publicado, puede ser implementado y ejecutado por otros investigadores para comprobar sus resultados.

Otros aspectos que se deben tener contemplar durante el diseño de un MBA son tanto considerar el número adecuado de parámetros, como las variables del modelo. Estos aspectos dependen del grado de realismo y precisión que requiera el modelo. Sin embargo, se debe tener en cuenta además, que los modelos con un gran número de parámetros pueden ser más difíciles de calibrar, y por lo tanto sufrir sobreajustes [115]. Con respecto a estos parámetros, en [14], se definen las siguientes condiciones:

- Es preferible un modelo con parámetros significativos ante un modelo con parámetros de ajuste sin sentido.
- Es preferible un modelo con variables operativas a un modelo que contiene variables que no pueden medirse.
- Dado el mismo número de parámetros, es preferible un modelo explicativo antes que un modelo puramente descriptivo.
- En caso de un apropiado ajuste comparable, en el paso de calibración del modelo, se debe elegir el modelo con la mejor potencia predictiva.
- Dada una potencia predictiva comparable de dos modelos, se debería seleccionar el más sencillo.

### Simulación de modelos

La simulación de un modelo basado en agentes puede realizarse utilizando diferentes herramientas informáticas. Estas herramientas pueden ser lenguajes de programación basada en objetos, como Python, Java o Visual Basic; librerías de los lenguajes de programación especializadas en la simulación de MBA, como MASON (Java), Swarm (Java; Objective-C), RePast (Java), entorno de trabajo Mesa (Python); y entornos de programación especializados en MBA, como NetLogo, Matlab, SPARK o StarLogo.

En [116] los autores presentan un amplio resumen de las herramientas utilizadas para modelizar y simular los modelos basados en agentes. Además, han llevado a cabo un análisis de dichas herramientas, basándose en la facilidad que brindan al usuario para el desarrollo de modelos, frente a la capacidad de modelización computacional o el nivel de escalabilidad de los modelos.

De acuerdo a este análisis, las herramientas como Netlogo [117] tienen una capacidad de modelización media alta y un nivel de desarrollo simple o fácil; Matlab [118] brinda una capacidad de modelización media alta y un nivel de desarrollo moderado; y Mesa [119] ofrece una capacidad de modelización media baja y un nivel de desarrollo simple o moderado. Un ejemplo ilustrativo de la simulación de un modelo en el entorno Mesa se muestra en la Figura 2.8.

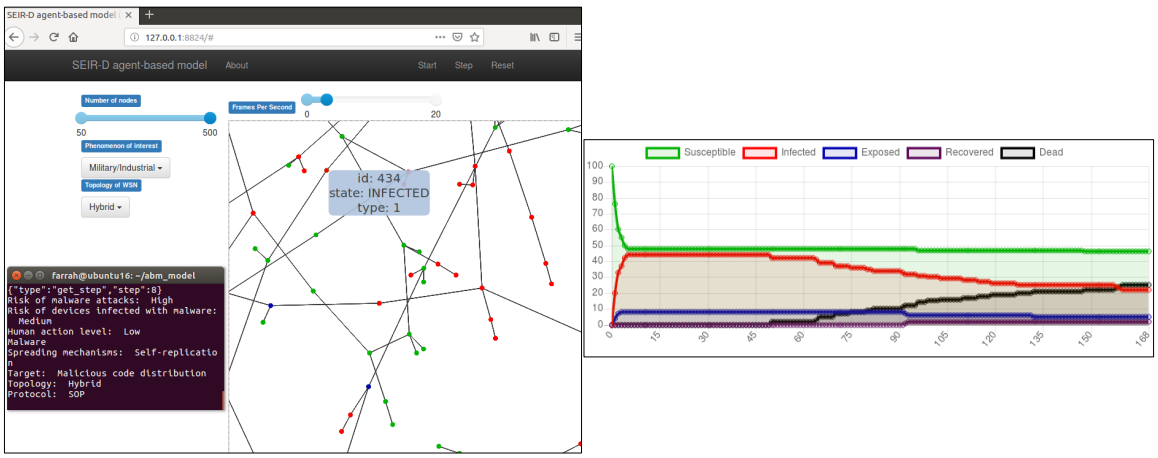



Figura 2.8 Ejemplo ilustrativo del a) entorno Mesa en la consola y el navegador y b) el gráfico de la evolución de los compartimentos.

Una vez seleccionada la herramienta de simulación, se deben considerar otros aspectos como la documentación del código, para permitir que tanto el programador como otros investigadores puedan ajustar o corregir errores. Las pruebas de factibilidad permiten la

verificación y pruebas del modelo para comprobar que los resultados obtenidos son los esperados. Por lo tanto, la aplicación de buenas prácticas a la hora de programar, como estructurar adecuadamente el código fuente, inicializar todas las variables, e incluir los parámetros previamente establecidos, pueden evitar errores en los resultados. Finalmente, se deben especificar las condiciones iniciales del modelo, como la interacción de los agentes, los valores iniciales de todas las variables, establecer los límites de las condiciones y calibrar el modelo.

Para evitar posibles problemas, es recomendable establecer limitaciones en el modelo computacional, como el rango de validez, el propósito del modelo, indicar si el modelo se ha diseñado para entender hechos específicos o científicos, aplicaciones o predicciones del mundo real. Además, se pueden indicar las ventajas y desventajas con respecto a otros modelos existentes [14].

### 2.3.4. Ventajas y desventajas de los modelos basados en agentes

Se incluyen a continuación algunas de las ventajas de los modelos basados en agentes [13, 113, 120]:

- Permiten una estructura natural e intuitiva conservando la modelización e implementación del sistema investigado. Además, resulta más sencillo describir el comportamiento de los agentes individuales a la vez que se proporcionan los datos necesarios, que describir el comportamiento del sistema general.
- Los MBA ofrecen numerosas opciones para seleccionar el entorno de los agentes y permiten la inclusión simultánea de múltiples tipos de espacios.
- Los MBA pueden simular el aprendizaje tanto a nivel individual como de población. El aprendizaje puede modelarse de varias formas, entre ellas, se puede citar el aprendizaje individual, en el que los agentes aprenden de su propia experiencia; el aprendizaje evolutivo, donde la población de agentes aprende; y el aprendizaje social, en el que algunos agentes imitan o son enseñados por otros agentes.
- Los agentes pueden ser adaptables en sus acciones e interacciones con otros agentes. Normalmente, los agentes se adaptan moviéndose, imitando y/o replicando. La adaptación puede ocurrir a nivel individual o de población.
- Son flexibles y proporcionan una alta escalabilidad.
- La heterogeneidad de los agentes suele ser muy diversa.

Sin embargo, estos modelos pueden presentar algunas desventajas, como las siguientes [13, 113, 120]:

- Demandan mayor requerimiento de características computacionales que otros tipos de modelos.
- En el ciclo de vida de la modelización, el desarrollo de un MBA requiere normalmente varios pasos críticos y lentos, como la verificación y validación, la replicación y reproducibilidad, la acreditación, el aseguramiento de calidad y la certificación. Si estos pasos no se llevan a cabo adecuadamente, puede haber poca confianza en las percepciones y predicciones proporcionadas por un estudio de simulación.
- Otros aspectos como la necesidad de eficacia en la programación informática, la reutilización, el análisis riguroso de datos y el análisis extensivo de los resultados, pueden disminuir la aplicabilidad general de los MBA.

### 2.3.5. Aplicaciones

Los modelos basados en agentes son cada vez más utilizados en diferentes disciplinas científicas, como arqueología [15], economía [16], infraestructura [121], sociedad y cultura [14], militar [122], tecnología [123], biología [124], salud [13], agricultura [125] y sistemas ecológicos [17].

Desde el punto de vista de un sistema biológico, los agentes pueden representar células o moléculas que pueden residir bajo una configuración discreta o continua. Por otro lado, en los sistemas sociales, entender un sistema político o económico requiere más que una simple comprensión de los individuos que lo componen [126]. Por lo tanto, es necesario deducir cómo interactúan los individuos entre sí y cómo los resultados pueden ser más que la suma de las partes. Por consiguiente, los MBA se adaptan muy bien a los objetivos de las ciencias sociales, y así permiten ofrecer soluciones a muchos otros problemas importantes de nuestro medio ambiente, vida silvestre, salud o finanzas [127].

Al mismo tiempo, estos modelos se han utilizado en los problemas epidemiológicos, contribuyendo así al bienestar de la humanidad, puesto que los MBA consiguen reproducir muchas características de brotes de enfermedades reales y sus predicciones resultan fáciles de percibir, ya que los gestores de control de enfermedades aceptan los resultados de la simulación e intervienen con éxito en las estrategias de vacunación preventiva [128].

Los MBA se han convertido en una técnica notable en la modelización y análisis de los suministros eléctricos [129]. Dentro del ámbito médico, los MBA se han centrado en las

dolencias humanas, como los tumores malignos cancerígenos, la cicatrización de heridas, la inmunología, el sistema vascular o los procesos metabólicos [130].

Existe una gran variedad de otras áreas de aplicación de los MBA en ingeniería y ciencias, que incluyen el diseño de sistemas autoorganizados, el análisis de contaminantes para formular políticas para un hábitat más ecológico [131], el transporte y la logística [132], la detección y prevención de fallas en sistemas distribuidos [133], la fabricación, la producción y el diseño de sistemas críticos de seguridad [134].

En los últimos años, científicos e ingenieros están investigando la utilización de los MBA en la ciencia computacional, particularmente en términos de biología de sistemas [116].

# Capítulo 3

# Modelo basado en agentes SEIRS-D

El modelo basado en agentes, SEIRS-D, que se propone en este trabajo, es un modelo individual, discreto y estocástico. Este novedoso modelo ha permitido analizar el comportamiento del malware desde una nueva perspectiva, mediante la incorporación de nuevos elementos que permiten ajustar las características del modelo con valores más realistas, de acuerdo al entorno. Tanto el entorno como la acción humana y los dispositivos que interactúan con la red se consideren agentes en el modelo propuesto. Los coeficientes agrupan las características de los diferentes agentes y el comportamiento de la red de sensores inalámbrica en el medio ambiente. Al mismo tiempo, las reglas de transición se han ajustado en base a en estos coeficientes.

Además, el comportamiento y las características de un agente pueden evaluarse individualmente en un intervalo de tiempo $t$. Por último, este modelo añade las ventajas de los modelos basados en agentes como un nuevo paradigma para la propagación del malware en las redes de sensores inalámbricas.

Para la creación del modelo SEIRS-D se han seguido los pasos del ciclo de modelización de los MBA, detallado en la Sección 2.3.3; estos pasos se describen a continuación:

- Pregunta de investigación: para este modelo se ha formulado la siguiente pregunta: ¿Cómo se propaga el malware en las redes de sensores inalámbricas a partir de las características individuales de los elementos involucrados en el proceso?

- Diseño: se han definido los seis agentes, los siete coeficientes y las seis reglas de transición, en los cuales se fundamenta el modelo teórico SEIRS-D basado en agentes.

- Desarrollo: se ha utilizado el entorno de trabajo Mesa para crear el modelo computacional que se apoya en el diseño previo. Este entorno utiliza Python como entorno de programación.

- Verificación: se han realizado las pruebas con diferentes valores para comprobar que el modelo funciona correctamente. Esto se ha realizado de forma manual en el código del modelo computacional.
- Calibración: se han establecido los valores iniciales a partir de varios escenarios que han sido definidos, a partir de los fenómenos de interés y las topologías. Estos valores pueden ser escogidos en la interfaz gráfica del modelo computacional.
- Análisis de sensibilidad: se han analizado las gráficas que permiten visualizar los resultados de la ejecución del modelo computacional, para comprobar que los resultados obtenidos varían de acuerdo a los valores iniciales.
- Validación: se ha verificado que el comportamiento del modelo representa la propagación del malware en redes de sensores inalámbricas, a través de la comparación con modelos propuestos previamente por otros autores, que han utilizado otras técnicas de modelado.
- Documentación: se ha detallado el modelo SEIRS-D en el manuscrito actual, incluyendo el modelo teórico y la implementación computacional. Además, se ha documentado el análisis de los resultados obtenidos tanto por escenario como por topología; finalmente, se ha proporcionado el código en Python del modelo computacional.

En el modelo SEIRS-D propuesto, los nodos sensores pueden adoptar, en cada instante de tiempo $t$, uno de los siguientes estados, detallados en la Figura 3.1:

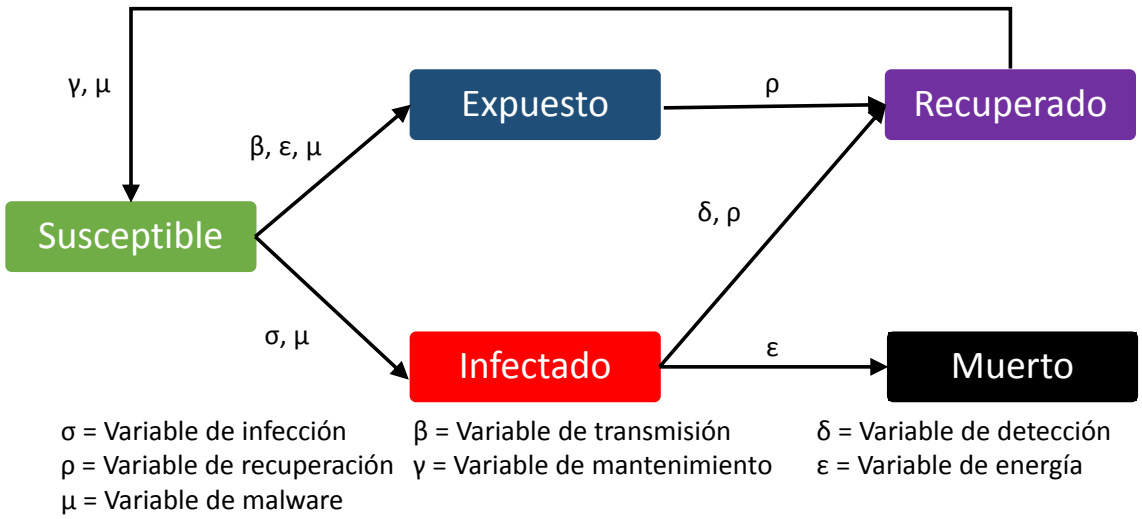



Figura 3.1 Esquema del modelo SEIRS-D.

- Susceptible ($S$): el sensor no ha sido infectado por el malware, pero tiene las características computacionales para ser infectado.
- Expuesto ($E$): el sensor ha sido atacado por el malware, pero no es capaz de transmitir el malware a los sensores vecinos debido a las características de estos sensores y del malware.
- Infectado ($I$): el sensor ha sido infectado por el malware. Este sensor infectado puede tener la capacidad de realizar intentos de infección a sus vecinos.
- Recuperado ($R$): el sensor adquiere inmunidad temporal cuando el malware ha sido eliminado con éxito o se han instalado parches de seguridad.
- Muerto ($D$): el sensor queda inutilizado cuando su energía se ha agotado, esto puede suceder rápidamente si el sensor ha sido infectado por malware, o en otros casos cuando la vida útil de la batería llega a su fin.

Se supone que la población de nodos permanece constante, por consiguiente, $S(t) + E(t) + I(t) + R(t) + D(t) = N$, en cada instante de tiempo $t$. Para un tiempo $t$ dado, $N$ es el número total de agentes, $S(t)$ el número de agentes susceptibles, $E(t)$ el número de agentes en estado expuesto, $I(t)$ los agentes en estado infectado, $R(t)$ los agentes en estado recuperado y finalmente, $D(t)$ es el número total de agentes en estado muerto.

Los agentes son entidades autónomas y heterogéneas que pueden interactuar entre sí o con el medio ambiente, de acuerdo con las reglas de transición. En el modelo propuesto en este estudio, estos agentes se definirán en la Sección 3.1. El entorno de los agentes representa el mundo virtual simulado en el que existen los agentes. Además, los coeficientes determinan las características asociadas a cada agente; estos coeficientes se describirán en la Sección 3.2. Por último, el comportamiento de los agentes se establece mediante normas que definen la respuesta de los agentes a los cambios del entorno y las relaciones con otros agentes. En la Sección 3.3 se detallarán estas reglas de transición.

## 3.1. Agentes en el modelo SEIRS-D

El modelo SEIRS-D está compuesto por seis agentes principales representados en la Figura 3.2: los sensores, el malware, la topología de la red, el fenómeno de interés, la acción humana y los dispositivos. Estos agentes han sido seleccionados después de analizar los diferentes entornos y características que pueden estar presentes mientras funciona una red de sensores inalámbrica. En este caso, los sensores son el componente esencial de la red, que

para comunicarse entre ellos utilizan una topología, mientras se encuentran desplegados en un entorno que se considera el fenómeno de interés a monitorizar. Dependiendo del entorno los humanos pueden realizar diferentes acciones sobre la red. Por otro lado, los datos que generan las WSN pueden ser consultados desde diferentes dispositivos computacionales o pueden almacenarse en dispositivos externos, que pueden ser una de las posibles fuentes vulnerable para que los códigos maliciosos obtengan acceso a la red. Las principales características consideradas para cada agente son la importancia del nivel de participación en la red y la contribución al proceso de infección del malware.

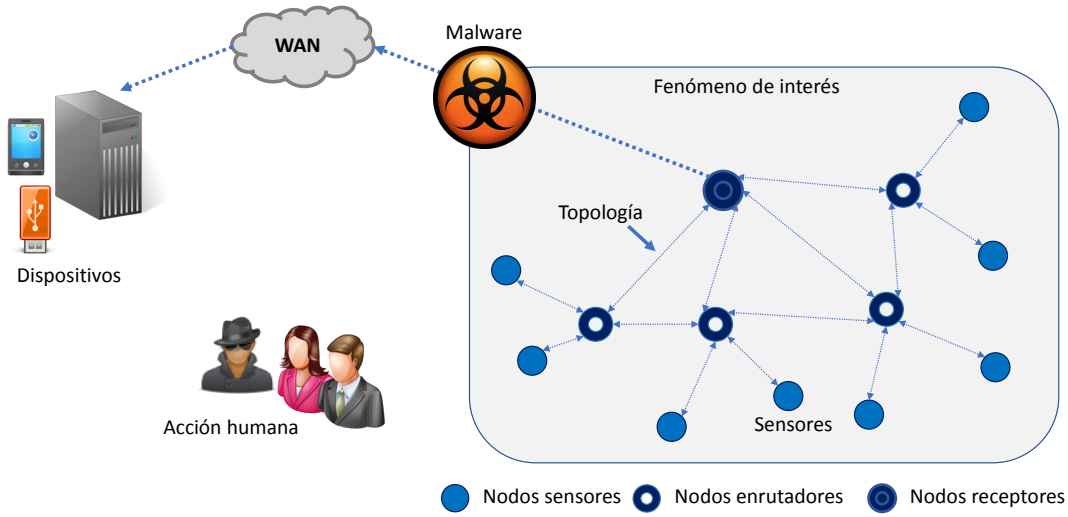



Figura 3.2 Esquema de los agentes que intervienen en el modelo modelo SEIRS-D propuesto.

En la Tabla 3.1 se detalla cada una de las 15 características asociadas a estos 6 agentes. Cada $j$-ésimo agente (con $1 \leq j \leq 6$) tiene asociadas unas características específicas denotadas por $k$ (con $k \in \{1,\ldots,15\}$), que toman valores $q$ que son diferentes para cada agente y cada característica de este. Las probabilidades asociadas a estos valores han sido expresadas por $P_{k,q}$ y verifican que $\sum_q P_{k,q} = 1$ para cada $k$. Así por ejemplo, la probabilidad de que un determinado sensor ($q = 1$) tenga una capacidad computacional baja ($k = 2$) es $P_{k,q} = P_{2,1}$, verificando que $P_{2,1} + P_{2,2} = 1$.

Se describen a continuación las características específicas de los agentes del modelo, sus valores y las probabilidades correspondientes a cada agente:

1º) Nodos sensores: son responsables de recoger los datos directamente del entorno, es el elemento principal dentro de la WSN. Las 7 características de los nodos sensores son las siguientes:

Tabla 3.1 Agentes y sus características.

| **Tipo de agentes** | | **Características** | | **Valores** | | **Probabilidades** |
|---|---|---|---|---|---|---|
| $j$ | **Nombre** | $k$ | **Nombre** | $q$ | **Nombre** | $P$ |
| 1 | Sensores | 1 | Tipo | 1 | Nodos sensores | $P_{1,1}$ |
| | | | | 2 | Nodos enrutadores | $P_{1,2}$ |
| | | | | 3 | Nodos receptores | $P_{1,3}$ |
| | | 2 | Capacidad computacional | 1 | Bajo | $P_{2,1}$ |
| | | | | 2 | Alto | $P_{2,2}$ |
| | | 3 | Consumo de energía | 1 | Muy bajo | $P_{3,1}$ |
| | | | | 2 | Bajo | $P_{3,2}$ |
| | | | | 3 | Medio | $P_{3,3}$ |
| | | | | 4 | Alto | $P_{3,4}$ |
| | | | | 5 | Muy alto | $P_{3,5}$ |
| | | 4 | Capacidad de transmisión y recepción de información | 1 | Bajo | $P_{4,1}$ |
| | | | | 2 | Alto | $P_{4,2}$ |
| | | 5 | Nivel de seguridad de los nodos | 1 | Bajo | $P_{5,1}$ |
| | | | | 2 | Medio | $P_{5,2}$ |
| | | | | 3 | Alto | $P_{5,3}$ |
| | | 6 | Método de recolección de datos | 1 | Periódicamente | $P_{6,1}$ |
| | | | | 2 | Estímulo externo | $P_{6,2}$ |
| | | | | 3 | Por petición | $P_{6,3}$ |
| | | 7 | Ciclo de trabajo | 1 | Activo | $P_{7,1}$ |
| | | | | 2 | Inactivo | $P_{7,2}$ |
| 2 | Malware | 8 | Tipo | 1 | Malware diseñado para WSN | $P_{8,1}$ |
| | | 9 | Mecanismo de propagación | 1 | Autorreplicación | $P_{9,1}$ |
| | | | | 2 | Explotación | $P_{9,2}$ |
| | | | | 3 | Interacción del usuario | $P_{9,3}$ |
| | | 10 | Objetivo | 1 | Distribución de código malicioso | $P_{10,1}$ |
| | | | | 2 | Extracción de información | $P_{10,2}$ |
| | | | | 3 | Denegación de servicios | $P_{10,3}$ |
| 3 | Topología de red | 11 | Tipo | 1 | Estrella | $P_{11,1}$ |
| | | | | 2 | Malla | $P_{11,2}$ |
| | | | | 3 | Híbrido | $P_{11,3}$ |
| | | | | 4 | Red libre de escala | $P_{11,2}$ |
| | | | | 5 | Mundo pequeño | $P_{11,2}$ |
| | | 12 | Protocolos de enrutamiento | 1 | SOP (*Self-Organizing Protocol*) | $P_{12,1}$ |
| | | | | 2 | EECR (*Energy Efficient Clustering and Routing*) | $P_{12,2}$ |
| 4 | Fenómeno de interés | 13 | Riesgo de ataque de malware | 1 | Bajo | $P_{13,1}$ |
| | | | | 2 | Medio | $P_{13,2}$ |
| | | | | 3 | Alto | $P_{13,3}$ |
| 5 | Acción humana | 14 | Nivel | 1 | Bajo | $P_{14,1}$ |
| | | | | 2 | Medio | $P_{14,2}$ |
| | | | | 3 | Alto | $P_{14,3}$ |
| 6 | Dispositivos | 15 | Riesgo de dispositivos infectados con malware | 1 | Bajo | $P_{15,1}$ |
| | | | | 2 | Medio | $P_{15,2}$ |
| | | | | 3 | Alto | $P_{15,3}$ |

1. Tipo de sensor: los sensores que pueden formar parte de la red son los nodos sensores, los enrutadores y los receptores, así como las especificaciones técnicas que permiten a cada nodo realizar diferentes funciones.
2. Capacidad computacional: se ha clasificado como alta y baja. Un nodo con baja capacidad puede tener un procesador de uso general, memoria estática, baterías como fuente de energía, un sensor y una antena inalámbrica interna. Un nodo con alta capacidad puede tener un procesador con funciones especiales, memoria dinámica, células solares como fuente de energía, múltiples sensores y una antena inalámbrica externa.
3. Consumo de energía: puede variar entre muy alto, alto, medio, bajo y muy bajo.
4. Capacidad de transmisión y recepción de información: se establece un rango alto cuando el nodo tiene antenas externas; y un rango bajo cuando el nodo tiene antenas internas.
5. Nivel de seguridad: se ha clasificado como de alto nivel cuando tiene métodos de seguridad avanzados; de nivel medio si utiliza claves criptográficas, y de nivel bajo si dispone de medidas de seguridad.
6. Método de recopilación de datos: se ha clasificado en mediciones periódicas, estímulos externos o por solicitudes.
7. Ciclo de trabajo: un nodo se encuentra en estado activo cuando el nodo toma medidas ambientales o transmite datos, y en estado inactivo mientras el nodo se despierta o duerme.

2º) Malware: es el código malicioso diseñado para realizar intrusiones en los sistemas, quebrantar políticas de seguridad o transportar archivos dañinos. En este modelo se consideran las siguientes 3 características del malware:

8. Tipo de malware: en este modelo no se hace referencia a un malware específico, se considera un malware genérico diseñado para WSN; sin embargo, pueden proponerse otros modelos para virus, gusanos, troyanos, etc.
9. Mecanismo de propagación: puede ser por autorreplicación, explotación o por la interacción con el usuario.
10. Objetivo: puede ser la distribución de códigos maliciosos, la extracción de información o la denegación de servicio.

3º) Topología de red: se refiere a las interconexiones de los nodos dentro de la red. Se han considerado las siguientes dos características de este agente:

11. Tipo de topología, que puede ser en estrella, malla o híbrida.
12. Protocolos de enrutamiento: los más utilizados en las WSN son SOP y EECR.

4º) Fenómeno de interés: está relacionado con el entorno donde se ha desplegado la red. En este trabajo, el fenómeno se ha dividido de acuerdo al riesgo de que ocurra un ataque de malware, como se detalla a continuación:

13. Riesgo de ataque de malware: riesgo alto, cuando se trata de fenómenos militares e industriales; riesgo medio, cuando se trata de fenómenos sanitarios y medioambientales; y riesgo bajo, cuando el fenómeno es de actividades cotidianas y multimedia.

5º) Acción humana: está relacionada con el nivel de actividad que los técnicos, administradores, usuarios o atacantes pueden tener dentro de la WSN. La única característica que se considera es:

14. El nivel de acción humana en la red: se ha clasificado en nivel alto cuando está relacionado con las actividades cotidianas o los fenómenos multimedia; nivel medio a los fenómenos de la medicina o el medio ambiente; y nivel bajo a los fenómenos militares o industriales.

6º) Dispositivos: incluyen todos los dispositivos externos y dispositivos computacionales que forman parte o interactúan con la red. Por ejemplo, un dispositivo externo puede ser una memoria USB u otro tipo de memoria externa, un CD/DVD o un disco duro. Asimismo, los dispositivos computacionales pueden ser ordenadores, dispositivos móviles, servidores y la estación base, que está conectada a la misma red o a otras redes que tienen comunicación directa con la WSN. La única característica que se considera es la siguiente:

15. Riesgo de dispositivos infectados con malware: se ha clasificado en alto riesgo cuando el dispositivo está infectado por malware diseñado para atacar la WSN, riesgo medio cuando los dispositivos están infectados con malware que no puede atacar la WSN, y riesgo bajo cuando el dispositivo no está infectado con malware.

## 3.2. Coeficientes asociados a los agentes

En esta sección, se describen los coeficientes asociados a los agentes que participan en el proceso de propagación del malware. Los sensores son los agentes que intervienen en

todos los coeficientes, excepto en el coeficiente de malware, debido a que los sensores son los únicos que pueden ser infectados por el malware. Se definen a continuación los siete coeficientes asociados a los agentes, representados por valores de probabilidad: infección, transmisión, detección, recuperación, mantenimiento, energía y malware.

### 3.2.1. Coeficiente de infección

El coeficiente de infección define la probabilidad de que el malware pueda comprometer un sensor susceptible. En este caso, las características computacionales de un sensor definirán la probabilidad de que sea infectado por el malware diseñado para atacar a la WSN. Este coeficiente depende de las características de los agentes definidos en el modelo; sin embargo, solo puede afectar a los agentes de tipo sensor, por lo que se utilizará en las reglas de transición para definir el estado del compartimento de cada sensor.

El coeficiente de infección del agente sensor $i$-ésimo en el instante de tiempo $t$ se representa matemáticamente por $a[i,t]$ donde $1 \leq i \leq n$, siendo $n$ el número total de sensores. Por consiguiente, la probabilidad de que el agente sensor $i$-ésimo susceptible se infecte en el momento $t+1$ viene dada por el siguiente coeficiente:

$$a[i,t] = \prod_{1 \leq j \leq 6, j \neq 3} X_j\left(\vec{k}_j\right), \quad 0 \leq a[i,t] \leq 1, \tag{3.1}$$

donde $\vec{k}_j$ es un vector que define las características de las que depende la variable $0 \leq X_j \leq 1$. Por consiguiente, la probabilidad de infección depende de los 5 factores: $X_1, X_2, X_4, X_5$ y $X_6$. $X_1$ representa las características de los sensores, $X_2$ considera las características del malware, $X_3$ son las características de la topología de la red, $X_4$ refleja las características del fenómeno de interés, $X_5$ representa las características de la acción humana, y $X_6$ considera las características de los dispositivos. La variable $X_3$ no se ha considerado porque la topología de la red puede verse afectada directamente por la transmisión del malware de un sensor a sus vecinos, debido a las interconexiones entre los nodos vecinos.

El riesgo de infección puede calcularse utilizando las probabilidades $P_{k,q}$ asociadas a cada posible valor de las características de los agentes (tal como se detallan en la Tabla 3.1). Como consecuencia, si $\vec{k}_j = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ con $1 \leq \alpha_1 < \alpha_2 < \ldots < \alpha_m \leq 15$, se puede afirmar lo siguiente:

$$X_j\left(\vec{k}_j\right) = \prod_{l=1}^{m} P_{\alpha_l,q}, \tag{3.2}$$

donde $q$ es el valor asociado a la característica $\alpha_l$ en cada caso.

En este sentido, para cada variable $X_k\left(\vec{k}_j\right)$ se han seleccionado las características que más influyen en el proceso de infección. Para ello, se han seleccionado diferentes escenarios, con la finalidad de determinar los posibles valores que hacen que el riesgo de infección sea alto, medio o bajo. Así por ejemplo, en el caso de un entorno militar o industrial (en el que la acción humana en la red es baja), se puede tener un alto riesgo de infección si se cumplen las siguientes condiciones:

- En los sensores: el consumo de energía es alto, el nivel de seguridad de los nodos es bajo, el método de recolección de datos se realiza periódicamente y el ciclo de trabajo es activo. Es decir, la variable $X_1\left(\vec{k}_1\right)$ depende de $\vec{k}_1 = (\alpha_1 = 2, \alpha_2 = 5, \alpha_3 = 6, \alpha_4 = 7)$ y por lo tanto
$$X_1\left(\vec{k}_1\right) = P_{2,2} \cdot P_{5,1} \cdot P_{6,1} \cdot P_{7,1}, \tag{3.3}$$
ya que los valores considerados para $q = 2$, $q = 1$, $q = 1$ y $q = 1$ corresponden a $k = 2$, $k = 5$, $k = 6$ y $k = 7$, respectivamente.

- En el malware: el mecanismo de propagación es por autorreplicación y el objetivo del malware es la denegación de servicios. Es decir, la variable $X_2\left(\vec{k}_2\right)$ depende de $\vec{k}_2 = (\alpha_1 = 9, \alpha_2 = 10)$ de manera que
$$X_2\left(\vec{k}_2\right) = P_{9,1} \cdot P_{10,3}, \tag{3.4}$$
ya que para $k = 9$ y $k = 10$ se consideran los valores $q = 1$ y $q = 3$, respectivamente.

- En el fenómeno de interés: el riesgo de sufrir un ataque por malware es alto. Es decir, la variable $X_4\left(\vec{k}_4\right)$ depende de $\vec{k}_4 = (\alpha_1 = 13)$ de manera que
$$X_4\left(\vec{k}_4\right) = P_{13,3}, \tag{3.5}$$
ya que para $k = 13$ se considera el valor $q = 3$ .

- En la acción humana: el nivel de acción humana sobre la red es bajo. Es decir, la variable $X_5\left(\vec{k}_5\right)$ depende de $\vec{k}_5 = (\alpha_1 = 14)$ de manera que
$$X_5\left(\vec{k}_5\right) = P_{14,1}, \tag{3.6}$$
ya que se considera para $k = 14$ el valor $q = 1$.

- Para los dispositivos: el riesgo de conectar dispositivos a la red, infectados por malware, es alto. Es decir, la variable $X_6\left(\vec{k}_6\right)$ depende de $\vec{k}_6 = (\alpha_1 = 15)$ de manera que

$$X_6\left(\vec{k}_6\right) = P_{15,3}, \tag{3.7}$$

  ya que para $k = 15$ se considera el valor $q = 3$.

### 3.2.2. Coeficiente de transmisión

Este coeficiente se refiere a la probabilidad de que un sensor infectado transmita malware a sus sensores vecinos susceptibles. La propagación de malware en una red depende fundamentalmente de la capacidad de transmitir el malware de un nodo a otro. Las características computacionales de cada sensor determinan esta capacidad. Este coeficiente resulta útil para estudiar las condiciones que deben cumplirse para que el malware se propague.

El coeficiente de transmisión del $i$-ésimo agente sensor en el instante de tiempo $t$ se representa matemáticamente por $b[i,t]$ donde $1 \leq i \leq n$, siendo $n$ el número total de sensores. Por consiguiente, la probabilidad de que el $i$-ésimo agente sensor infectado transmita (propague) el malware en el momento $t+1$ viene dada por el siguiente coeficiente:

$$b[i,t] = \prod_{1 \leq j \leq 3} X_j\left(\vec{k}_j\right), \quad 0 \leq b[i,t] \leq 1,$$

donde $\vec{k}_j$, igual que ocurría para el coeficiente de infección, es un vector que representa las características de las que depende la variable $j$, con $0 \leq X_j \leq 1$. La probabilidad de transmisión depende de los agentes sensores ($X_1$), del malware ($X_2$) y la topología de red ($X_3$). En este caso no se han considerado las características del fenómeno de interés ($X_4$), la acción humana ($X_5$) ni los dispositivos ($X_6$), puesto que no influyen directamente en la transmisión del malware de un sensor a otro, ya que estos factores no son parte física ni lógica de la red.

El riesgo de transmisión se determina utilizando la ecuación (3.2). Si se considera por ejemplo, el caso de una red con nodos de alta capacidad computacional, con un nivel de seguridad bajo, una topología de malla que permita que todos los nodos se comuniquen entre sí y que también utilice el protocolo EECR, puede tener un mayor riesgo de que los sensores puedan transmitir programas maliciosos a otros sensores, si se cumplen las siguientes condiciones:

- La variable $X_1\left(\vec{k}_1\right)$ depende de $\vec{k}_1 = (\alpha_1 = 2, \alpha_2 = 3, \alpha_3 = 4, \alpha_4 = 5, \alpha_5 = 7)$ de tal forma que

$$X_1\left(\vec{k}_1\right) = P_{2,2} \cdot P_{3,5} \cdot P_{4,2} \cdot P_{5,1} \cdot P_{7,1}, \tag{3.8}$$

dado que para $k = 2$, $k = 3$, $k = 4$, $k = 5$, y $k = 7$ se consideran los valores $q = 2$, $q = 5$, $q = 2$, $q = 1$ y $q = 1$, respectivamente.

- La variable $X_2\left(\vec{k}_2\right)$ depende de $\vec{k}_2 = (\alpha_1 = 9, \alpha_2 = 10)$ de tal forma que

$$X_2\left(\vec{k}_2\right) = P_{9,1} \cdot P_{10,3}, \tag{3.9}$$

dado que se consideran para $k = 9$ y $k = 10$ los valores $q = 1$ y $q = 3$, respectivamente.

- La variable $X_3\left(\vec{k}_3\right)$ depende de $\vec{k}_3 = (\alpha_1 = 12)$ de manera que

$$X_3\left(\vec{k}_3\right) = P_{12,2}, \tag{3.10}$$

dado que para $k = 12$ se considera el valor $q = 2$.

### 3.2.3. Coeficiente de detección

Este coeficiente se refiere a la probabilidad de que se detecte el malware en el sensor infectado. El tiempo de detección del malware puede ser mayor o menor, dependiendo de los efectos del fenómeno de interés en el ciclo de trabajo y la vida útil de la batería. Además, un sensor infectado puede recibir mantenimiento, que podría aumentar la probabilidad de que se detecte el malware.

El coeficiente de detección del $i$-ésimo agente sensor en el instante de tiempo $t$ se representa matemáticamente por $d[i,t]$ donde $1 \leq i \leq n$ y $n$ es el número total de sensores. Como resultado, la probabilidad de que el malware sea detectado en un agente sensor $i$-ésimo infectado, en el momento $t+1$, viene dada por el siguiente coeficiente:

$$d[i,t] = \prod_{1 \leq j \leq 5, j \neq 2,3} X_j\left(\vec{k}_j\right), \quad 0 \leq d[i,t] \leq 1,$$

donde $\vec{k}_j$ es un vector que describe las características de las que depende la variable $0 \leq X_j \leq 1$. Los factores $X_2, X_3$ y $X_6$ no han sido considerados en este coeficiente debido a que el malware, la topología y los dispositivos no influyen directamente en la detección de malware en un sensor infectado, puesto que el malware se detecta por alertas generadas por los mecanismos de seguridad o por los administradores de la red en el momento del mantenimiento de los sensores. El riesgo de detección puede evaluarse mediante la ecuación (3.2).

Por ejemplo, en el caso de una red con sensores de gran capacidad computacional, alto nivel de seguridad, y en la que el fenómeno de interés permite que el sistema sea supervisado

con frecuencia por los administradores de la red, es más probable que se detecte un malware que haya infectado un sensor cuando se cumplen las siguientes condiciones:

- La variable $X_1\left(\vec{k}_1\right)$ depende de $\vec{k}_1 = (\alpha_1 = 2, \alpha_2 = 3, \alpha_3 = 5)$ tal que

$$X_1\left(\vec{k}_1\right) = P_{2,2} \cdot P_{3,5} \cdot P_{5,3}, \tag{3.11}$$

  para $k = 2$, $k = 3$ y $k = 5$ se consideran los valores $q = 2$, $q = 5$ y $q = 3$, respectivamente.

- La variable $X_4\left(\vec{k}_4\right)$ depende de $\vec{k}_4 = (\alpha_1 = 13)$ tal que

$$X_4\left(\vec{k}_4\right) = P_{13,1}, \tag{3.12}$$

  puesto que para $k = 13$ se considera el valor $q = 1$.

- La variable $X_5\left(\vec{k}_5\right)$ depende de $\vec{k}_5 = (\alpha_1 = 14)$ tal que

$$X_5\left(\vec{k}_5\right) = P_{14,3}, \tag{3.13}$$

  para $k = 14$ se considera el valor $q = 3$.

### 3.2.4. Coeficiente de recuperación

Este coeficiente indica la probabilidad de que un sensor adquiera inmunidad temporal, después de que el malware se haya eliminado adecuadamente o que el sensor haya sido reparado. Cuando un sensor se ha recuperado, es probable que vuelva a su estado de funcionamiento normal. Por último, cuando el malware ha sido eliminado de toda la red, el sensor recuperado vuelve a ser susceptible.

El coeficiente de recuperación del $i$-ésimo agente sensor se representa matemáticamente por $r[i,t]$ donde $1 \leq i \leq n$, siendo $n$ el número total de sensores, en el instante de tiempo $t$. Por consiguiente, el siguiente coeficiente da la probabilidad de que el $i$-ésimo agente sensor infectado se recupere en el instante de tiempo $t+1$:

$$r[i,t] = \prod_{1 \leq j \leq 5, j \neq 2,4} X_j\left(\vec{k}_j\right), \quad 0 \leq r[i,t] \leq 1,$$

donde, como en casos anteriores, $\vec{k}_j$ es un vector que especifica las características de las que depende la variable $X_j$ ($0 \leq X_j \leq 1$). No se han considerado para el coeficiente de recuperación, las características: $X_2$, porque el malware debe haber sido eliminado para

que el sensor pase al estado recuperado, ni las $X_4$ y $X_6$ porque el fenómeno de interés y los dispositivos no influyen en el proceso de recuperación.

La probabilidad de recuperación se determina a partir de la ecuación 3.2. Para el caso de una red con sensores de alta capacidad computacional, por ejemplo, con una topología híbrida y mayor acción humana, es más probable que los sensores puedan recuperarse de la infección si se verifican las siguientes condiciones:

- La variable $X_1\left(\vec{k}_1\right)$ depende de $\vec{k}_1 = (\alpha_1 = 2)$ de tal manera que

$$X_1\left(\vec{k}_1\right) = P_{2,2}, \tag{3.14}$$

  ya que se considera para $k = 2$ el valor $q = 2$.

- La variable $X_3\left(\vec{k}_3\right)$ depende de $\vec{k}_3 = (\alpha_1 = 11, \alpha_2 = 12)$ de tal manera que

$$X_3\left(\vec{k}_3\right) = P_{11,3} \cdot P_{12,2}, \tag{3.15}$$

  puesto que para $k = 11$ y $k = 12$ se consideran los valores $q = 3$ y $q = 2$, respectivamente.

- La variable $X_5\left(\vec{k}_5\right)$ depende de $\vec{k}_5 = (\alpha_1 = 14)$ de tal manera que

$$X_5\left(\vec{k}_5\right) = P_{14,3} \tag{3.16}$$

  ya que se considera para $k = 14$ el valor $q = 3$.

### 3.2.5. Coeficiente de mantenimiento

El coeficiente de mantenimiento indica la probabilidad de que los administradores de la red realicen el mantenimiento de un sensor. Este mantenimiento puede ser tanto de software (por ejemplo, actualizaciones de los sistemas operativos, antivirus u otras medidas de seguridad), como de hardware (como el reemplazo de la fuente de energía).

El coeficiente de mantenimiento del $i$-ésimo agente sensor en el instante de tiempo $t$ se representa matemáticamente por $c[i,t]$, donde $1 \leq i \leq n$ y $n$ es el número total de sensores. Por consiguiente, la probabilidad de que un agente sensor $i$-ésimo infectado reciba el mantenimiento en el momento $t+1$, viene dada por el siguiente coeficiente:

$$c[i,t] = \prod_{1 \leq j \leq 5, j \neq 2,4} X_j\left(\vec{k}_j\right), \quad 0 \leq c[i,t] \leq 1,$$

donde $\vec{k}_j$ es el vector que determina las características de las que depende la variable $X_j$ ($0 \leq X_j \leq 1$). Los factores que no se han considerado en este coeficiente son: $X_2$, porque el objetivo del mantenimiento es evitar que la red se infecte con un malware; $X_4$, ya que el fenómeno de interés puede recibir mantenimiento en mayor o menor grado, dependiendo de la acción humana; y $X_6$ porque los dispositivos externos no pertenecen directamente a la red.

El riesgo de mantenimiento puede calcularse a partir de la ecuación (3.2); en particular, una red en la que los sensores tienen un consumo de energía alto y la intervención humana en el sistema es alta, es más probable que se realice el mantenimiento de los sensores si se cumplen las siguientes condiciones:

- La variable $X_1\left(\vec{k}_1\right)$ depende de $\vec{k}_1 = (\alpha_1 = 3)$ tal que

$$X_1\left(\vec{k}_1\right) = P_{3,5}, \tag{3.17}$$

  puesto que para $k = 3$ se considera el valor $q = 5$.

- La variable $X_2\left(\vec{k}_2\right)$ depende de $\vec{k}_2 = (\alpha_1 = 11)$ tal que

$$X_2\left(\vec{k}_2\right) = P_{11,3}, \tag{3.18}$$

  puesto que para $k = 11$ se considera el valor $q = 3$.

- La variable $X_3\left(\vec{k}_3\right)$ depende de $\vec{k}_3 = (\alpha_1 = 14)$ tal que

$$X_4\left(\vec{k}_3\right) = P_{14,3}, \tag{3.19}$$

  puesto que para $k = 14$ se considera el valor $q = 3$.

### 3.2.6. Coeficiente de energía

El coeficiente de energía indica si un sensor tiene la potencia mínima necesaria para continuar sus funciones operativas normales. Por lo general, el nivel de energía de un sensor disminuye a medida que pasa el tiempo, y puede aumentar cuando los administradores o técnicos de la red realizan procedimientos de mantenimiento. Sin embargo, el nivel de energía de un nodo puede disminuir drásticamente cuando el nodo ha sido infectado por un malware.

El coeficiente de energía del $i$-ésimo agente sensor se representa matemáticamente por $e[i,t]$, donde $1 \leq i \leq n$ y $n$ es el número total de sensores, en el instante de tiempo $t$. Consecuentemente, la probabilidad de que un agente sensor $i$-ésimo tenga la energía para

continuar su operación regular en el tiempo $t+1$, está dada por el siguiente coeficiente:

$$e[i,t] = X_1\left(\vec{k}_1\right), \quad 0 \leq e[i,t] \leq 1,$$

donde $\vec{k}_1$ es un vector que denota las características de las que depende la variable $0 \leq X_j \leq 1$. Se ha determinado que en el factor $X_1$ el nivel de energía de los sensores está relacionado con las funciones realizadas por cada nodo de la red.

Por ejemplo, un sensor con un bajo consumo de energía en cada tiempo $t$, es más probable que conserve un nivel óptimo de energía durante un largo período de tiempo. La probabilidad del nivel de energía se puede obtener utilizando la ecuación (3.2). Además, se debe verificar la siguiente condición:

- La variable $X_1\left(\vec{k}_1\right)$ depende de $\vec{k}_1 = (\alpha_1 = 3)$ de tal manera que

$$X_1\left(\vec{k}_1\right) = P_{3,2}, \tag{3.20}$$

  puesto que para $k = 3$ se considera el valor $q = 2$.

### 3.2.7. Coeficiente de malware

Este coeficiente indica si el malware ha sido diseñado para atacar una WSN. El malware que ataca la red puede tener diferentes objetivos y tipos de ataque; sin embargo, sería capaz de infectar un sensor teniendo en cuenta el sistema operativo y la capacidad computacional de dicho sensor, que por lo general es inferior a la capacidad de un ordenador.

El coeficiente de malware para el $i$-ésimo agente sensor, en el instante de tiempo $t$, se representa matemáticamente por $m[i,t]$, donde $1 \leq i \leq n$ y en este caso $n$ es el número de tipos diferentes de malware. Como resultado, la probabilidad de que un malware diseñado para WSN ataque al $i$-ésimo agente sensor, en el instante $t+1$, viene dada por el siguiente coeficiente:

$$m[i,t] = X_2\left(\vec{k}_2\right), \quad 0 \leq m[i,t] \leq 1,$$

donde $\vec{k}_2$ es un vector que define las características de las que depende la variable $X_j$ ($0 \leq X_j \leq 1$). En este coeficiente se ha seleccionado el factor $X_2$ porque identifica las características esenciales que puede tener un malware diseñado para WSN. La probabilidad de este malware especializado en atacar a las WSN puede calcularse mediante la ecuación (3.2). La probabilidad de infección de las WSN aumenta cuando el malware ha sido diseñado específicamente para ataques a redes de tipo WSN, para lo cual debe darse la siguiente condición:

- La variable $X_2\left(\vec{k}_2\right)$ depende de $\vec{k}_2 = (\alpha_1 = 8)$ de tal manera que

$$X_2\left(\vec{k}_2\right) = P_{8,1}, \tag{3.21}$$

ya que para $k = 8$ se considera el valor $q = 1$.

## 3.3. Reglas de transición

Las reglas de transición del modelo SEIRS-D propuesto para modelizar la propagación de malware en redes de sensores inalámbricas, definen las condiciones que debe cumplir un sensor $x_i$, para cambiar de un estado a otro en un instante de tiempo $t$, donde el estado $x_i \in \{S,E,I,R,D\}$. Las condiciones de estas reglas se basan en los coeficientes definidos en la Sección 3.2.

Los coeficientes que pueden afectar a los agentes sensores son los de infección, transmisión, detección, recuperación, mantenimiento y energía. El coeficiente de malware, en cambio, se utiliza para los agentes de malware. A continuación se detallan las definiciones de las variables aleatorias de infección, transmisión, detección, recuperación, mantenimiento, energía y malware, que se obtienen a partir de cada uno de los coeficientes respectivos:

- Variable de infección

$$\sigma[x_i] = \begin{cases} 0, & \text{si el nodo no ha sido infectado por malware con probabilidad } 1 - a[i,t]. \\ 1, & \text{si el nodo ha sido infectado por malware con probabilidad } a[i,t]. \end{cases}$$

- Variable de transmisión:

$$\beta[x_i] = \begin{cases} 0, & \text{si el nodo no puede transmitir el malware a sus vecinos con probabilidad } 1 - b[i,t]. \\ 1, & \text{si el nodo puede transmitir el malware a sus vecinos con probabilidad } b[i,t]. \end{cases}$$

- Variable de detección:

$$\delta[x_i] = \begin{cases} 0, & \text{si no se ha detectado la infección por un malware en el nodo con probabilidad } 1 - d[i,t]. \\ 1, & \text{si se ha detectado la infección por malware en el nodo con probabilidad } d[i,t]. \end{cases}$$

- Variable de recuperación:

$$\rho[x_i] = \begin{cases} 0, & \text{si el nodo no se ha recuperado de la infección por un malware con probabilidad } 1 - r[i,t]. \\ 1, & \text{si el nodo se ha recuperado de la infección por un malware con probabilidad } r[i,t]. \end{cases}$$

- Variable de mantenimiento:

$$\gamma[x_i] = \begin{cases} 0, & \text{si el nodo no ha recibido mantenimiento con probabilidad } 1 - c[i,t]. \\ 1, & \text{si el nodo ha recibido mantenimiento con probabilidad } c[i,t]. \end{cases}$$

- Variable de energía:

$$\varepsilon[x_i] = \begin{cases} 0, & \text{si el nodo no tiene un nivel de energía óptimo con probabilidad } 1 - e[i,t] \\ 1, & \text{si el nodo tiene un nivel de energía óptimo con probabilidad } e[i,t] \end{cases}$$

- Variable de malware:

$$\mu[x_i] = \begin{cases} 0, & \text{si el malware no ha sido diseñado para atacar WSN con probabilidad } 1 - m[i,t] \\ 1, & \text{si el malware ha sido diseñado para atacar WSN con probabilidad } m[i,t] \end{cases}$$

A continuación se detallan las reglas de transición, que definen cómo los agentes interactúan entre sí y con su entorno.

### 3.3.1. Transición Susceptible a Infectado

Un sensor puede pasar del estado susceptible en el tiempo $t$, al estado infectado en el instante de tiempo $t+1$, si durante un ataque se cumplen las condiciones para que las variables de infección y malware puedan activarse; es decir, cuando esas variables toman el valor 1; mientras que el resto de variables permanece con valor 0; entonces puede decirse que el sensor ha sido infectado por un malware. Esto puede representarse con la siguiente expresión matemática:

$$x_i(t) = S(t) \Rightarrow x_i(t+1) = I(t+1), \text{ cuando } \sigma[x_i] = 1 \text{ AND } \mu[x_i] = 1.$$

### 3.3.2. Transición Susceptible a Expuesto

Un sensor en estado susceptible en el instante de tiempo $t$, puede pasar al estado expuesto en el tiempo $t+1$, cuando no tiene la capacidad computacional para transmitir a sus vecinos el malware que lo ha infectado. Para que esto se cumpla las variables de transmisión, energía y malware deben tener valor 1, y el resto de variables debe mantener su valor en 0. Esto, se puede describir de la siguiente manera:

$$x_i(t) = S(t) \Rightarrow x_i(t+1) = E(t+1), \text{ cuando } \beta[x_i] = 1 \text{ AND } \varepsilon[x_i] = 1 \text{ AND } \mu[x_i] = 1.$$

### 3.3.3. Transición Infectado a Muerto

Un sensor infectado por malware en el instante de tiempo $t$, puede pasar a un estado muerto en el instante $t+1$, cuando el nivel de energía es bajo o se ha agotado la batería; la variable afectada en esta transición es la energía, que debe tener valor 1, mientras las demás variables son 0. Esto se puede representar de la siguiente forma:

$$x_i(t) = I(t) \Rightarrow x_i(t+1) = D(t+1), \text{ cuando } \varepsilon[x_i] = 0.$$

### 3.3.4. Transición Infectado a Recuperado

Un sensor puede pasar de estado infectado en el tiempo $t$, a estado recuperado en el instante de tiempo $t+1$, cuando las medidas de seguridad que posee el sensor han detectado que se encuentra infectado por un malware y se realizan las acciones necesarias para eliminar el malware del sensor, para que se pueda restablecer el funcionamiento normal del sensor. Las variables involucradas en este proceso son las de detección y recuperación, las cuales deben cambiar su valor a 1, mientras el resto de variables tiene valor 0. Esto se puede expresar matemáticamente de la siguiente manera:

$$x_i(t) = I(t) \Rightarrow x_i(t+1) = R(t+1), \text{ cuando } \delta[x_i] = 1 \text{ AND } \rho[x_i] = 1.$$

### 3.3.5. Transición Expuesto a Recuperado

Un sensor en estado expuesto en el instante de tiempo $t$ puede pasar al estado recuperado en el instante $t+1$ cuando el sensor, utilizando las medidas de seguridad que tiene instaladas, ha detectado y eliminado el malware que lo había infectado y además, sus vecinos no se encuentran en estado infectado; es posible recuperar posteriormente el funcionamiento normal del sensor. La variable relacionada con este proceso es la de recuperación, la cual

debe tomar el valor de 1. Esta transición se puede expresar como:

$$x_i(t) = E(t) \Rightarrow x_i(t+1) = R(t+1), \text{ cuando } \rho[x_i] = 1.$$

### 3.3.6. Transición Recuperado a Susceptible

Un sensor puede pasar del estado recuperado en el instante $t$, al estado susceptible en el instante $t+1$, cuando el malware ha sido eliminado de todos los sensores de la red. Sin embargo, la seguridad perfecta no existe; el sistema puede ser atacado por un nuevo malware con características y objetivos similares o diferentes. En este caso, las variables de mantenimiento y el malware deben tomar el valor 1; esto se puede expresar como:

$$x_i(t) = R(t) \Rightarrow x_i(t+1) = S(t+1) \text{ cuando } \gamma[x_i] = 1 \text{ AND } \mu[x_i] = 0.$$

# Capítulo 4

# Simulación

La simulación de un modelo basado en agentes puede realizarse utilizando diferentes programas especializados, que pueden ser gratuitos o de pago. Cada programa proporciona características útiles para diferentes áreas en las que se puede realizar un estudio. En [116] los autores presentan un amplio resumen de las herramientas utilizadas para modelizar y simular modelos basados en agentes, así como su área de aplicación, el análisis entre la facilidad de desarrollo de los modelos y la capacidad de modelización computacional; consideran además el nivel de escalabilidad de los modelos.

La simulación del modelo SEIRS-D que se propone en esta memoria ha sido desarrollada en el entorno de trabajo Mesa [119]. Se ha seleccionado este entorno puesto que es una librería de código abierto que permite crear modelos basados en agentes, a partir de componentes básicos incorporados o realizar implementaciones personalizadas. Además, puede soportar un número considerable de nodos y los resultados pueden ser visualizados de forma interactiva desde el navegador.

Otras características importantes de Mesa son que este entorno permite la creación de varias clases de agentes que pueden interactuar entre sí y con el entorno que ha sido definido al mismo tiempo, además, la forma como se despliegan los nodos puede ser a través de una cuadrícula o una red, esta red puede crearse utilizando paquetes para el análisis de redes en Python como NetworkX.

Por otro lado, la selección de los parámetros iniciales de la simulación puede realizarse desde el navegador, también, es posible controlar el avance del tiempo, ya sea paso a paso o de manera continua, detener y reanudar la simulación en cualquier momento, permitiendo el análisis del comportamiento de los agentes.

Mesa utiliza el servidor web Apache2 y el lenguaje de programación Python para el análisis de datos; siendo una alternativa a NetLogo, que resulta más eficiente y con mayores

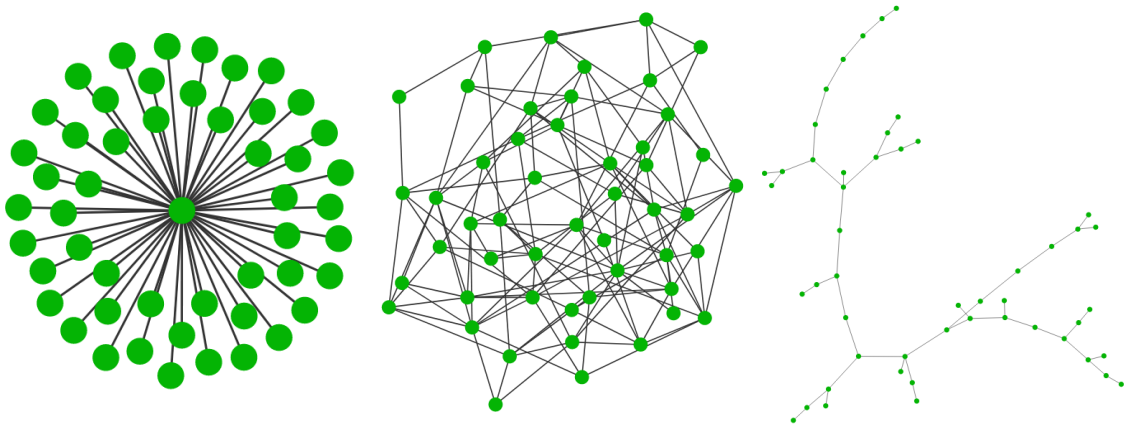
posibilidades. Algunos de los modelos simulados en NetLogo han sido replicados en el entorno Mesa [135].

Se han establecido requisitos mínimos para la selección de la herramienta de simulación, como la utilización de Python como lenguaje de programación, que permite adaptar el entorno de trabajo al modelo propuesto, donde se puedan definir los diferentes tipos de agentes, los coeficientes y las reglas de transición para definir la forma en que los agentes interactúan. Además, la herramienta debe soportar diferentes tipos de redes como las redes con topología híbrida, de malla, en estrella, y también redes complejas. Se ha considerado que la generación de los gráficos debe ser eficiente, y que la visualización de los resultados se muestre en tiempo real, para posibilitar un análisis de los resultados paso a paso. Finalmente, la interfaz de selección de los parámetros debe ser amigable para el usuario. El entorno Mesa cumple con estas características, por lo que, resulta adecuado para la simulación del modelo propuesto.

Este entorno ha sido instalado en una Máquina Virtual (VM) con distribución Ubuntu Linux 16.4. Los recursos computacionales que se han utilizado en la VM son un procesador Intel i5, una memoria RAM de 2GB, un disco duro de 10GB y conexión a Internet. La instalación del entorno se ha realizado a través de la consola de Linux, siguiendo el manual de usuario.

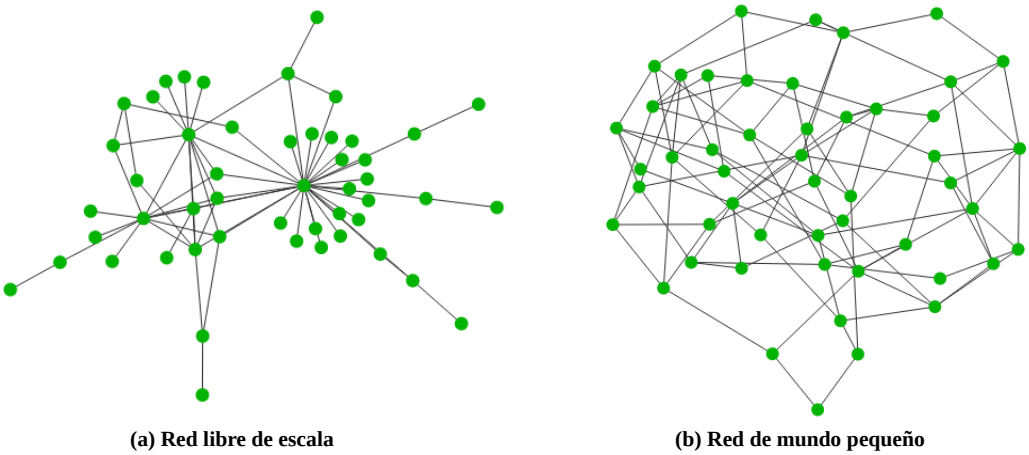
El número de nodos que soporta una red de sensores inalámbricas depende de varios factores, como el estándar y el protocolo utilizado [136], la cobertura de la antena del nodo, la densidad del área monitorizada, la topología y el tipo de aplicación de la red [137]. Por ejemplo, el estándar Zigbee/802.15.4 puede soportar alrededor de 65,000 nodos en una red mientras que el protocolo 6LoWPAN solo soporta 100 nodos [138]; para la monitorización de la actividad de un volcán se pueden utilizar 10 nodos [139], sin embargo, el despliegue de sensores para aplicaciones militares necesitan más de 100 nodos [140].

La simulación del modelo se ha realizado con 250 nodos para cada una de las topologías híbrida, de malla y en estrella, que han sido definidas para cada fenómeno de interés, y donde cada nodo corresponde a un agente sensor (Figura 4.1); además, se han simulado los fenómenos de interés militar e industrial en dos tipos de redes complejas con 800 nodos, estas han sido una red libre de escala y un mundo pequeño (ver Figura 4.2). El tiempo máximo de simulación ha sido de 168 horas. A continuación, se presenta la descripción de cada escenario y los resultados obtenidos.



**(a) Estrella** **(b) Malla** **(c) Híbrida**

Figura 4.1 Visualización de las topologías en el entorno Mesa, cuando se consideran 50 nodos.



**(a) Red libre de escala** **(b) Red de mundo pequeño**

Figura 4.2 Vista de las redes complejas en el entorno Mesa, considerando 50 nodos.

## 4.1. Escenario de simulación 1

En este escenario, el entorno es el fenómeno de interés militar e industrial, el malware se ha definido con la autorreplicación como un mecanismo de propagación, y su objetivo es la extracción de información en la red. Además, el riesgo de ataques de malware es alto, la acción humana es baja, y el riesgo de infección a través de dispositivos es medio. Al mismo tiempo, el sistema utiliza el protocolo SOP para la comunicación de los nodos, la capacidad computacional de cada nodo es alta, el alcance de la antena de los nodos es alto y el nivel de seguridad de los nodos es de medio a alto.

Las configuraciones de topología que se han utilizado, en las que la distribución de los nodos ha sido generada de forma aleatoria, son las siguientes:

a) Topología híbrida con 98 nodos sensores, 151 nodos enrutadores y un nodo receptor.

b) Topología de malla con 249 nodos enrutadores y un nodo receptor.

c) Topología en estrella con 249 nodos sensores y un nodo receptor.

Estos parámetros de simulación han dado lugar a los gráficos que se muestran en la Figura 4.3 para las topologías híbrida, de malla y en estrella. En el eje de abscisas se representa el tiempo en horas y en el eje de ordenadas se representa el número de nodos. Se observa la evolución de los compartimentos de los sensores en cada una de las topologías consideradas.

En el caso de la topología híbrida, los nodos permanecen prácticamente en el mismo estado inicial. Es posible que el malware solo haya afectado a un clúster, por lo que el resto de la red no se ha visto afectada y, por lo tanto, la propagación del malware se haya controlado rápidamente. Consecuentemente, los resultados de las curvas durante el tiempo total de la simulación muestran que los sensores en estado infectado, expuesto y recuperado mantienen un punto de equilibrio estable.

Por otro lado, los sensores susceptibles de la topología híbrida comienzan a pasar a estado muerto aproximadamente a las 120 horas. Consecuentemente, no se ha obtenido el punto de equilibrio en las curvas de los sensores susceptibles y muertos, al finalizar la simulación.

La evolución de la propagación de malware, en el caso de la topología de malla, el número de sensores expuestos se ha incrementado rápidamente, mientras que al inicio de la simulación los sensores infectados tienen un rápido crecimiento, luego se observa que estos sensores infectados comienzan a recuperarse pronto, y la infección parece detenerse.

En cuanto al punto de equilibrio de la topología de malla, los sensores susceptibles se estabilizan a partir de las 30 horas aproximadamente, con un número bajo de sensores en este estado. Los sensores expuestos y recuperados, encuentran el punto de equilibrio a partir

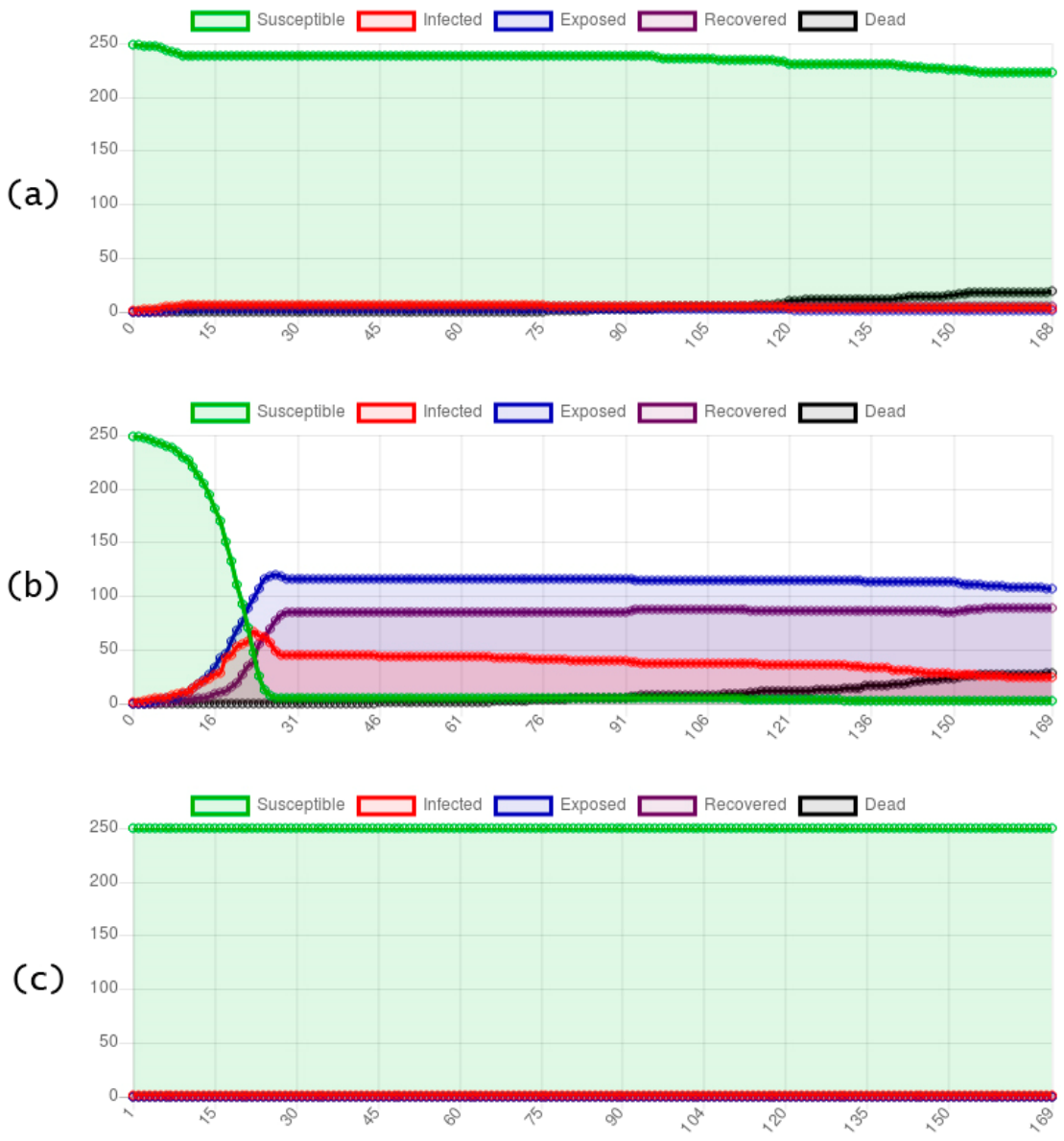



Figura 4.3 Simulación de los fenómenos militar e industrial; en las topologías a) híbrida, b) malla y c) estrella.

de las 35 horas aproximadamente, donde existe alrededor de 120 sensores expuestos y 90 sensores recuperados, al momento que finaliza la simulación.

En el caso de los sensores infectados, a partir de las 140 horas, comienzan a pasar a estado muerto, por lo que estas dos curvas no alcanzan el punto de equilibrio cuando finaliza la simulación de la topología de malla.

En la topología en estrella, solo se ha sido infectado un sensor, es posible que este sensor no haya podido infectar al enrutador, por lo tanto, la infección no se ha propagado al resto de la red. Se puede decir, que en este caso existe un punto de equilibrio libre de infección.

## 4.2. Escenario de simulación 2

Los parámetros del entorno que se consideran para este escenario son los fenómenos de atención médica y medioambiente. En la Figura 4.4 se pueden observar las gráficas de la evolución de los compartimentos de los sensores en las topologías híbrida, de malla y en estrella. Los agentes en este caso son los siguientes: un malware con la explotación como mecanismo de propagación cuyo objetivo es la denegación de servicio. Además, el riesgo de ataques de malware es de nivel medio, el nivel de acción humana es también medio y el riesgo de infección a través de dispositivos es bajo. Asimismo, la red utiliza el protocolo SOP para la comunicación de los nodos, la capacidad computacional de cada nodo es baja, el alcance de las antenas de los nodos es alto y la seguridad de los nodos es de nivel bajo a medio.

Se han definido las siguientes topologías y tipos de nodos, donde la distribución de los nodos ha sido generada de forma aleatoria:

a) Topología híbrida con 89 nodos sensores, 160 nodos enrutadores y un nodo receptor.

b) Topología de malla con 249 nodos enrutadores y un nodo receptor.

c) Topología en estrella con 249 nodos sensores y un nodo receptor.

En cuanto a la topología híbrida, la infección se ha propagado rápidamente por la red, dando como resultado un alto número de sensores infectados. Sin embargo, se observa que los sensores expuestos y recuperados son muy pocos desde el inicio de la simulación; se pueden contabilizar en unos 30 sensores susceptibles a partir de las 40 horas aproximadamente. Por esta razón, los sensores en estado susceptibles, expuestos y recuperados mantienen un punto de equilibrio en sus curvas, al final del tiempo de simulación.

Aproximadamente, a las 75 horas de la simulación de la topología híbrida, los sensores infectados comienzan a pasar a estado muerto. En este caso, estas dos curvas no han alcanzado

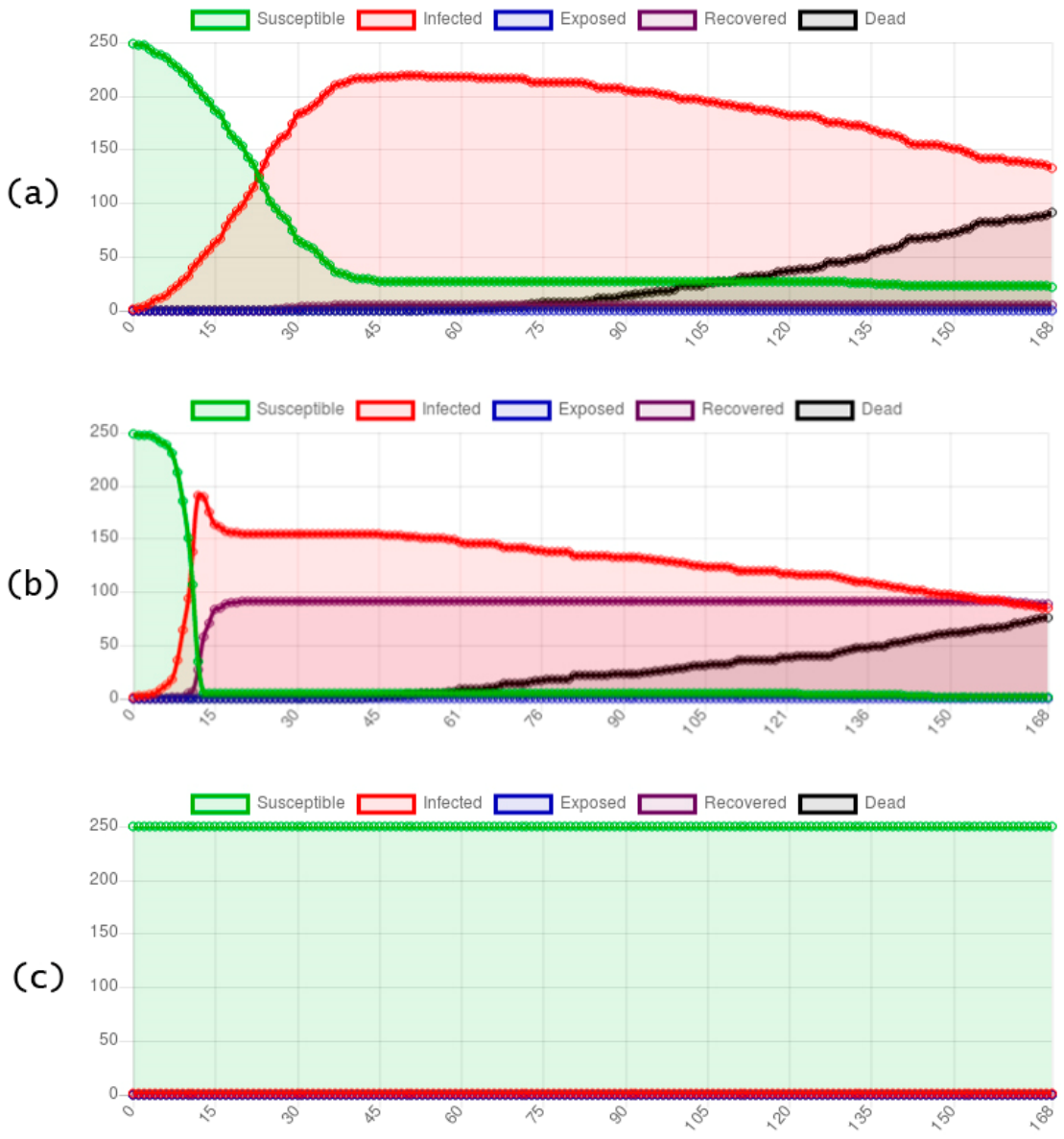



Figura 4.4 Simulación de los fenómenos de atención médica y medioambiente; en las topologías a) híbrida, b) malla y c) estrella.

el punto de equilibrio. Por otro lado, es posible que está red deje de tener un funcionamiento adecuado debido al gran número de sensores muertos.

La propagación del malware en la topología de malla ha incrementado rápidamente el número de sensores infectados, luego se observa que algunos de estos sensores infectados comienzan a recuperarse; sin embargo, al pasar el tiempo el número de sensores recuperados no incrementa, y los sensores infectados comienzan a pasar a estado muerto.

Se puede observar en la topología de malla que los sensores susceptibles bajan drásticamente, quedando muy pocos sensores en este estado, el número de nodos expuestos no incrementa durante toda la simulación, por lo que en ambos casos se ha alcanzado el punto de equilibrio; en cuanto a los nodos recuperados, mantienen un punto equilibrio estable a partir de las 20 horas de la simulación, aproximadamente.

Las curvas de los sensores en estado infectado y muerto de la topología de malla no han alcanzado el punto de equilibrio, ya que a partir de las 60 horas aproximadamente, gran parte de los sensores infectados han cambiado de estado a muerto. Por esta razón, es posible que el número de sensores en funcionamiento de la red, disminuya por debajo del 50 %, este número correspondería con los sensores recuperados.

En la topología en estrella, el malware solo ha infectado a un sensor y no se ha propagado por el resto de la red. Debido a esto, el punto de equilibrio se mantiene libre de infección.

## 4.3. Escenario de simulación 3

Por último, se ha definido un escenario que considera los fenómenos de actividades diarias y multimedia. En este caso se ha definido un malware con mecanismo de propagación a través de la interacción del usuario y su objetivo es el fraude en la red. Al mismo tiempo, el riesgo de ataques de malware es bajo, el nivel de acción humana es alto y el riesgo de infección a través de dispositivos es alto. Además, la capacidad computacional de cada nodo es de baja a alta, el alcance de la antena de los nodos está entre bajo y alto, la seguridad de los nodos es baja o alta, y la red utiliza el protocolo EECR para la comunicación de los nodos.

En la Figura 4.5, se muestran los gráficos obtenidos con los resultados de la evolución de los compartimentos de los sensores en las topologías híbrida, de malla y en estrella.

Se han definido las siguientes características, donde la distribución de los nodos ha sido generada de forma aleatoria:

a) Topología híbrida con 96 nodos sensores, 153 nodos enrutadores y un nodo receptor.

b) Topología de malla con 249 nodos enrutadores y un nodo receptor.

c) Topología en estrella con 249 nodos sensores y un nodo receptor.

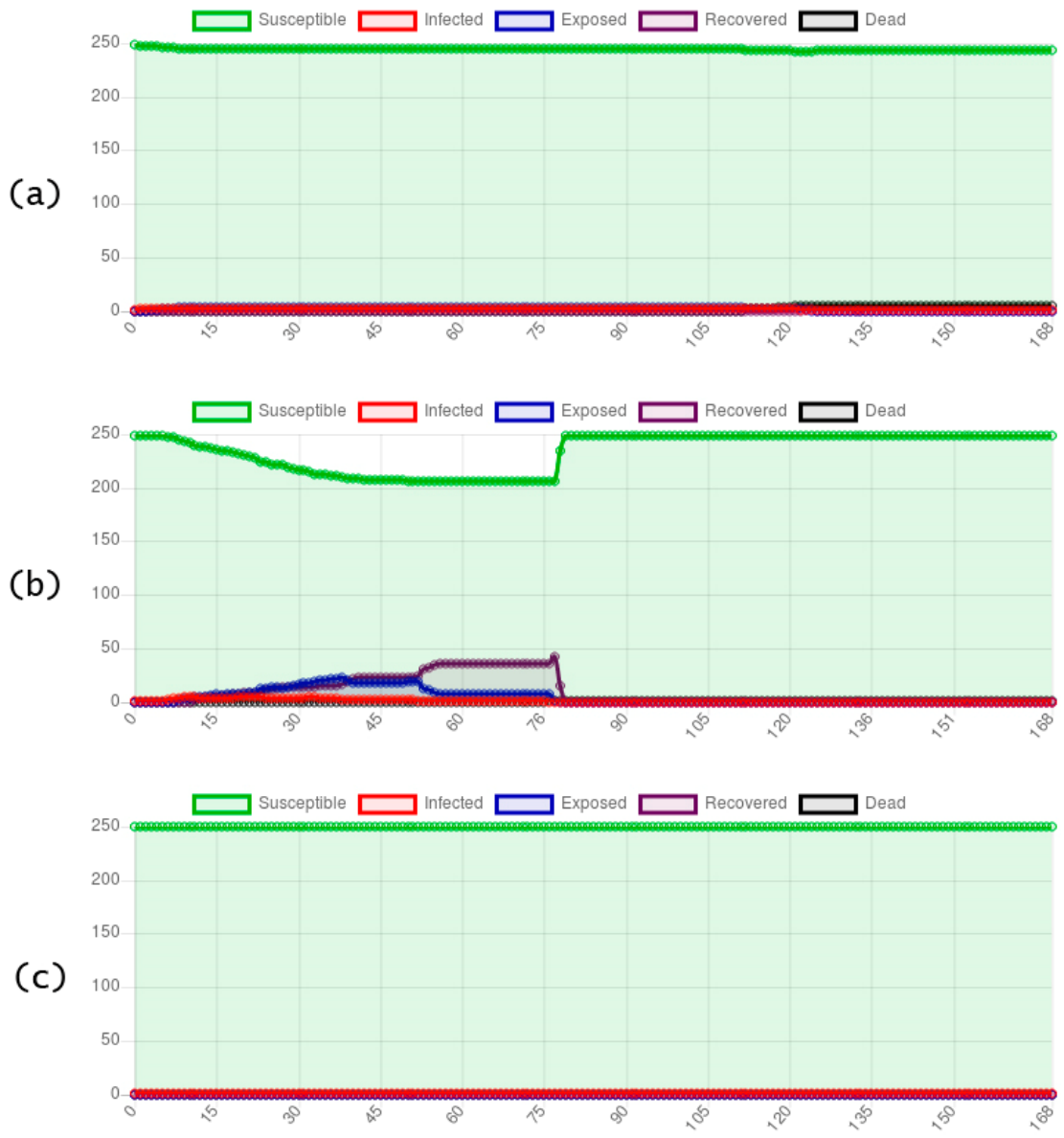



Figura 4.5 Simulación de los fenómenos de actividades diarias y multimedia; en las topologías a) híbrida, b) malla y c) estrella.

La propagación del malware en la topología híbrida no ha sido significativa, en este caso muy pocos sensores han sido infectados, y algunos sensores infectados o susceptibles han cambiado a estado muerto.

Por lo tanto, el punto de equilibrio en este caso se mantiene con una pequeña infección, que podría haber ocurrido en una pequeña parte de la red, es decir, en un clúster específico, desde donde el malware no ha podido continuar propagándose, lo que supone una ventaja de la topología híbrida.

En la topología de malla, se da una infección de malware que puede ser eliminada totalmente de la red de forma rápida y efectiva. En este caso, la propagación del malware se inicia en unos pocos sensores infectados, estos a su vez transmiten el malware a otros sensores que pasan a estado expuestos. Estos sensores expuestos no pueden transmitir el malware a otros nodos, por lo que pueden funcionar como barrera para detener la propagación.

Se observa posteriormente que tanto los sensores infectados como los nodos expuestos comienzan a recuperarse de la infección en la topología de malla; finalmente, aproximadamente a las 80 horas de la simulación, el malware es eliminado de toda la red, y todos los sensores pasan nuevamente a estado susceptible. Posteriormente, se puede decir que todas las curvas logran el punto de equilibrio libre de infección.

En la topología en estrella, el malware no ha sido capaz de propagarse por la red, por lo que la red continúa su funcionamiento normal. Además, el punto de equilibrio de las curvas es libre de infección.

## 4.4. Simulación de redes complejas

Se considera en este caso la simulación de redes complejas en el entorno de los fenómenos de interés militar e industrial, en los que el malware pretende extraer información y su mecanismo de propagación es la autorreplicación. El riesgo de ataque es alto, el nivel de acción humana es bajo y el riesgo de infección por dispositivos infectados es medio. Los nodos tienen antenas de transmisión de alto alcance, el nivel de seguridad es medio a alto y la capacidad computacional es alta.

Los nodos se han distribuido de la siguiente forma:

a) Red libre de escala con 461 nodos sensores, 338 nodos enrutadores y 1 nodo receptor.

b) Red de pequeño mundo con 799 nodos enrutadores y 1 nodo receptor.

En la red libre de escala se han definido tres probabilidades: $\alpha$, $\beta$ y $\gamma$. La suma de estas tres probabilidades debe ser igual a 1. $\alpha$ es la probabilidad de añadir un nuevo nodo conectado a un nodo existente que ha sido seleccionado al azar, según la distribución del

grado de entrada; a esta probabilidad se le ha asignado un valor de $0,41$. $\beta$ es la probabilidad de añadir un enlace entre dos nodos existentes, donde un nodo existente ha sido seleccionado al azar según la distribución del grado de entrada y el otro nodo ha sido seleccionado al azar según la distribución del grado de salida. La probabilidad $\beta$ tiene un valor de $0,54$. Finalmente, $\gamma$ es la probabilidad de conectar un nuevo nodo con un nodo existente; este nodo existente se ha elegido aleatoriamente según la distribución del grado de salida. La probabilidad $\gamma$ tiene un valor de $0,05$.

En otras palabras, en una red libre de escala, un nodo existente tiene una probabilidad del 41 % de crear un enlace de entrada con un nodo nuevo, y una probabilidad del 5 % de crear un enlace de salida con este nuevo nodo. Además, la probabilidad de crear enlaces de entrada y salida entre dos nodos existentes es del 54 %.

Las probabilidades $\alpha$, $\beta$ y $\gamma$ han sido definidas a partir de los valores por defecto que proporciona el paquete NetworkX de Python, para un grafo libre de escala directo, es decir, un grafo donde las conexiones entre nodos se crean utilizando el algoritmo de apego preferencial, que depende de la naturaleza de la distribución de los grados de entrada y salida [141].

El algoritmo de apego preferencial está basado en el modelo de Barabási-Albert [142], que consiste en crear redes aleatorias libre de escala, empleando reglas a partir de un proceso estocástico llamado conexión preferencial, donde una cantidad inicial de enlaces se distribuye entre los nodos, y consecuentemente se asignan nuevos enlaces a los nodos que más enlaces previos poseen. En este caso, se utiliza la distribución de grados, la cual se refiere al número de conexiones de entrada y salida que tiene un nodo con otros nodos.

En este caso, el sesgo para elegir los nodos en la distribución del grado de entrada es $0,2$ y en la distribución del grado de salida es 0. Los resultados obtenidos para las redes libres de escala se pueden observar en la gráfica de la Figura 4.6.

En la red libre de escala, la propagación del malware ha comenzado lentamente, luego, el número de sensores infectados y sensores expuestos ha aumentado de manera similar. Sin embargo, la recuperación de los sensores ha comenzado pronto; por lo tanto, la infección ha podido ser controlada rápidamente, y se ha propagado solo en alrededor del 20 % de la red.

En este caso, los sensores infectados, expuestos y recuperados tienen un punto de equilibrio estable, mientras que los sensores susceptibles dan la impresión de que comienzan a pasar a estado muerto a partir de las 120 horas, por lo que es posible que el punto de equilibrio de los sensores en estado susceptible y muerto no hayan alcanzado el punto de equilibrio en el momento de finalizar la simulación.

Los parámetros que se han establecido en el mundo pequeño son $k = 4$, en una topología de anillo, donde un nodo se enlaza con su vecino más cercano, utilizando el algoritmo de aprendizaje automático $k$-NN, es decir, el método de clasificación de un nuevo elemento se

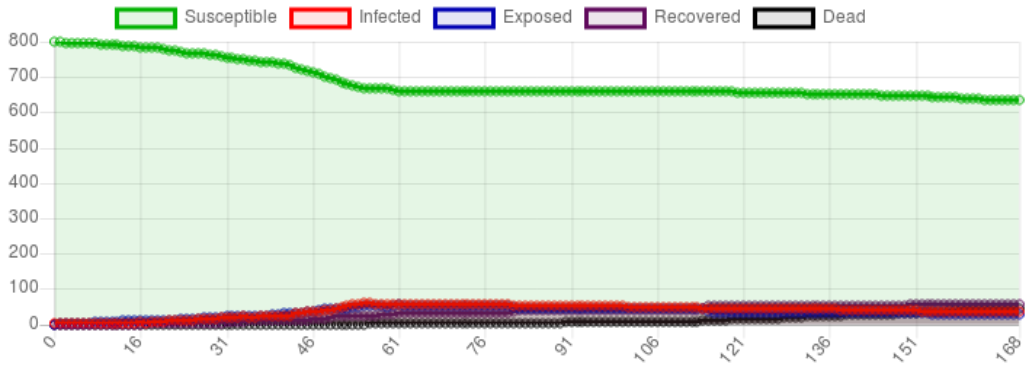



Figura 4.6 Simulación en redes libre de escala.

realiza de acuerdo a la clase más frecuente a la que pertenecen los $k$ vecinos más cercanos, y la probabilidad de volver a unir cada enlace es de $0,5$ y 20 intentos para generar un gráfico conectado; en la Figura 4.7 se muestran el resultado obtenido para esta red de mundo pequeño.

En la red de mundo pequeño, la propagación del malware se ha desarrollado de forma progresiva, distribuyéndose entre los sensores infectados y los sensores expuestos de manera muy similar. Cuando la infección ha progresado hasta las 40 horas aproximadamente, los sensores han empezado a recuperarse; sin embargo, un número importante de sensores infectados han empezado a perder su energía y, por lo tanto, a morir a partir de las 100 horas.

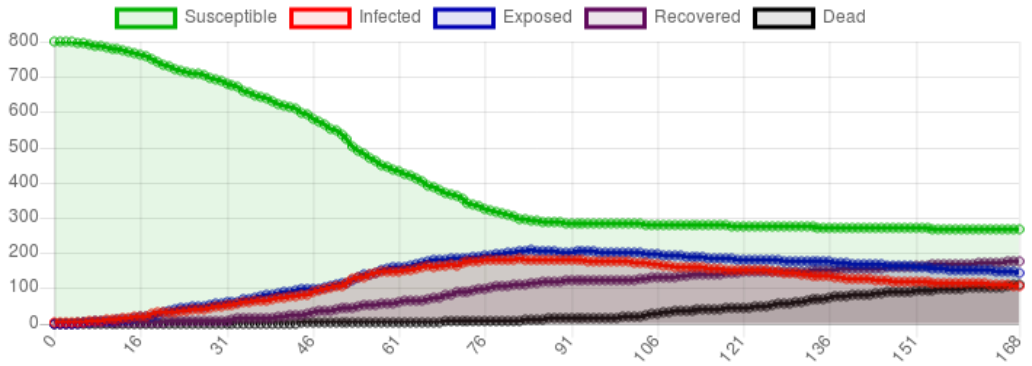



Figura 4.7 Simulación en redes de mundo pequeño.

Se puede observar que la curva de los sensores susceptibles ha alcanzado el punto de equilibrio aproximadamente a las 80 horas, sin embargo, las curvas de los sensores infectados, expuestos, recuperados y muertos no han alcanzado el punto de equilibrio en el momento que finaliza la simulación. En resumen, la red puede perder datos debido a fallos de intercomunicación entre los sensores.

## 4.5. Complejidad del modelo

La complejidad computacional se refiere a la cantidad de recursos necesarios para ejecutar un algoritmo. Los recursos que se consideran son el tiempo, es decir, el número de pasos ejecutados para completar una acción y la memoria utilizada para la ejecución. Un problema puede considerarse muy complejo si la cantidad de recursos utilizados para resolverlo es significativa, independientemente del algoritmo empleado [143].

El rendimiento del modelo propuesto no es muy costoso desde el punto de vista de la complejidad computacional. Obviamente depende del número de agentes considerados, $n$, y del tiempo total de la simulación, $T$.

Específicamente, el cálculo de los diferentes coeficientes epidemiológicos de propagación del malware (coeficientes de infección, transmisión, detección, recuperación, mantenimiento, energía y malware), realiza $255 \cdot n \cdot T$ operaciones, donde el número 255 corresponde al total de elementos de la red, es decir, 250 nodos, un malware, una topología, un fenómeno de interés, un humano y un dispositivo.

En cuanto a la complejidad lineal, para el modelo que realiza $255 \cdot n$ operaciones por tiempo de ejecución $T$, se define como $O(n)$ operaciones-bit, donde se representa la operación binaria cuando el tiempo de ejecución $T$ se incrementa linealmente con el tamaño de los bits de $n$; y $O(\log n)$ representa el tiempo máximo de ejecución proporcional al tamaño logarítmico de $n$.

Por otro lado, un simple cálculo muestra que la evaluación de las reglas de transición requiere $11 \cdot n \cdot T$ comparaciones, donde el número 11 corresponde a las veces que un sensor puede cambiar de estado, partiendo de un estado susceptible inicial.

# Capítulo 5

# Resultados de las simulaciones

En este trabajo se ha presentado el modelo SEIRS-D, un modelo basado en agentes definido para simular la propagación de malware en redes de sensores inalámbricas. La estructura de este modelo se define con agentes, coeficientes y reglas de transición. Los seis agentes que se han definido son los sensores, el malware, la topología de red, el fenómeno de interés, la acción humana y los dispositivos. Los sensores son los únicos agentes que pueden ser infectados por malware y las condiciones para que esto suceda vienen determinadas por las propias características de los sensores, el malware y la relación con el resto de agentes. Cada agente sensor ha sido clasificado en uno de los siguientes cinco compartimentos: susceptible, infectado, expuesto, recuperado y muerto.

Además, el modelo propuesto se basa en siete coeficientes significativos: infección, transmisión, detección, recuperación, mantenimiento, energía y malware. Se obtendrán resultados satisfactorios si se identifican correctamente dichos coeficientes y se establecen las reglas de transición adecuadas. En este modelo se han definido seis reglas de transición posibles para que los sensores cambien de un compartimento a otro.

La simulación del modelo propuesto se ha ejecutado en tres escenarios diferentes que corresponden al agente del fenómeno de interés. En cada situación se han establecido diferentes valores globales para simular un entorno real. Estos parámetros se han replicado en el agente correspondiente a la topología de la red. El análisis de los resultados se ha realizado en tres categorías, considerando en primer lugar los escenarios posibles y a continuación las diferentes topologías. Por último, se han detallado los resultados obtenidos en las simulaciones en redes complejas.

## 5.1. Resultados por escenario

En el primer escenario, la simulación se ha realizado en un entorno militar e industrial. En este tipo de redes, las características de los nodos y la seguridad de la red tienden a ser altas, y la red no debe estar al alcance de los atacantes. Por estas razones, es difícil que estas redes se infecten y el malware se propague en ellas. En la Figura 4.3 se observó que la tasa de infección es baja en las topologías híbrida y en estrella; sin embargo, en la topología de malla, la infección se ha extendido a un número más significativo de sensores.

En cuanto al análisis de las gráficas obtenidas para el primer escenario considerado, en la topología híbrida no se ha alcanzado el punto de equilibrio para los sensores en estados susceptibles y muertos; en la topología de malla, los sensores en estado infectado y muerto no han alcanzado el punto de equilibrio; sin embargo, en la topología en estrella, se ha obtenido un punto de equilibrio libre de infección.

Se puede concluir que para los casos de las topologías híbrida y de malla es necesario realizar más simulaciones, y prolongar el tiempo de simulación para lograr obtener el punto de equilibrio en todas las curvas. En ambas topologías, se aprecia que los sensores en estado muerto no alcanzan el punto de equilibrio. Esto puede deberse a varios factores como que la energía de los sensores susceptibles se va agotando debido a su continuo funcionamiento y que, en el caso de los sensores infectados, la energía se agota más rápidamente que cuando operan normalmente.

En todos los casos se ha observado que los sensores expuestos y recuperados han alcanzado el punto de equilibrio, esto se debe a que estos estados contribuyen a que la infección por malware no se propague al resto de la red.

En el segundo escenario, la simulación se ha realizado en el entorno de salud y medio ambiente. En estas redes, los sensores pueden tener características bajas, y los niveles de seguridad pueden considerarse entre medios y bajos, puesto que estas redes resultan más accesibles para los atacantes debido a sus vulnerabilidades, aunque puedan tener limitaciones dependiendo de su ubicación. En este caso, la infección puede tardar mucho tiempo en ser detectada. En la figura 4.4 se observó que la infección avanzó rápidamente en las topologías híbrida y de malla; sin embargo, en la topología de estrella la infección no se ha podido propagar.

Con respecto al análisis de las gráficas, tanto en la topología híbrida como en la topología de malla no se ha alcanzado el punto de equilibrio en los sensores con estados infectado y muerto; no obstante, en la topología en estrella se ha alcanzado un punto de equilibrio libre de infección.

En el caso de las topologías de malla e híbrida, donde no se ha alcanzado el punto de equilibrio en los sensores con estados infectado y muerto, es necesario prolongar el tiempo

de simulación para identificar el momento en el que se logra alcanzar el equilibrio. Esta particularidad se debe a que los nodos infectados por malware consumen mayor cantidad de energía en menor tiempo de operación, por lo tanto, pasan a estado muerto rápidamente. Es posible que en ambos casos la red deje de funcionar correctamente ya que han pasado a estado muerto alrededor de 100 sensores, después de 168 horas. Los sensores en estado susceptible, expuesto y recuperado han alcanzado el punto de equilibrio en todos los casos, esto se debe a que la propagación del malware alcanza su punto máximo y deja de propagarse; posteriormente, los sensores infectados pasan a estado muerto.

En el tercer escenario, la simulación se ha hecho en un entorno de actividades diarias y multimedia. Las características de los nodos difieren de un sensor a otro en estas redes; puede darse el caso de que la misma red tenga sensores con altas características y seguridad, y otros sensores con bajas prestaciones. Sin embargo, los usuarios interactúan con mayor frecuencia con estas redes, de modo que las actualizaciones o la detección de infecciones pueden hacerse más rápidamente. En este caso, la red de topología de malla ha sido capaz de recuperarse de una infección en un corto período de tiempo. Sin embargo, en los casos de topologías híbridas y en estrella, la infección ha alcanzando un número reducido de sensores (ver Figura 4.5).

Las curvas de las topologías híbrida y en estrella en este tercer escenario han alcanzado un punto de equilibrio prácticamente desde el inicio, esto se debe a que el malware no ha sido capaz de propagarse. Por otro lado, la topología de malla ha sido capaz de recuperarse totalmente de la infección y alcanzar el punto de equilibrio a partir de las 80 horas. Por lo tanto, en los tres casos, se ha alcanzado el punto de equilibrio al finalizar las simulaciones. Esto se puede deber al nivel alto de acción humana en estas redes.

En resumen, el tercer escenario, en el que el fenómeno de interés está relacionado con las actividades diarias y multimedia, es el que ha obtenido mejores resultados, por esta razón, la acción humana es más frecuente y el mantenimiento de la red es continuo; por lo tanto, las redes pueden evitar la propagación del malware o recuperarse de manera eficiente.

En cambio, en el primer escenario, donde el fenómeno de interés que se consideró fue el militar e industrial, el malware solo ha conseguido propagarse en la red con topología de malla. Mientras que el segundo escenario, donde el fenómeno de interés era el de atención médica y medioambiente, el malware ha logrado propagarse rápidamente en las topologías de malla e híbrida. En el caso del primero y segundo escenarios, es posible que se requiera que las simulaciones se desarrollen por un mayor tiempo para comparar los resultados.

## 5.2. Resultados por topología

En la topología híbrida, los sensores que pertenecen a un clúster se comunican directamente con el nodo receptor. Por esta razón, cuando el nodo receptor del clúster tiene las características computacionales y la seguridad altas, se evita que la propagación de la infección se extienda a sus vecinos y permanezca solo en una parte de la red hasta que sea eliminada. En otras palabras, el malware debe ser capaz de infectar a varios nodos receptores para poder propagar el malware a un mayor número de sensores.

En los escenarios con fenómeno de interés basado en el entorno militar e industrial y en el de actividades diarias y multimedia, se ha observado que la propagación de malware solo ha sido capaz de infectar a pocos sensores. Por otro lado, en el escenario con fenómeno de interés de salud y medioambiente, el malware se ha propagado rápidamente pero, de igual forma, se ha detenido rápidamente.

En cuanto al análisis de las gráficas obtenidas en las simulaciones realizadas, se puede concluir que, en el primero y segundo escenario, los sensores en estado muerto no alcanzan el punto de equilibrio; además, los sensores susceptibles e infectados tampoco alcanzan los puntos de equilibrio en los respectivos escenarios. Mientras que el tercer escenario se ha alcanzado el punto de equilibrio desde el principio.

En resumen, la propagación del malware se puede desarrollar rápidamente y afectar a pocos sensores, siendo la topología híbrida una topología que brinda capacidad de recuperación. Sin embargo, la capacidad computacional puede ser la debilidad de esta topología, tal como se ha observado en el segundo escenario, en el que las características computacionales de los sensores son bajas y por lo tanto, el malware ha logrado infectar a un número significativo de sensores.

La topología de tipo malla tiene la mayor tasa de infección y en redes con esta topología todos los nodos son capaces de comunicarse entre sí. En este caso, las características computacionales y de seguridad de los nodos pueden contribuir a evitar o recuperar el estado de funcionamiento normal, después de una infección.

En el escenario con fenómeno de interés de actividades diarias y multimedia, se puede observar la recuperación total de la red después de haber sido infectada. Sin embargo, en los escenarios con fenómeno de interés militar e industrial, de salud y medioambiente, la propagación del malware se desarrolla muy rápidamente. El problema es que los sensores no son capaces de recuperarse de la infección, por lo tanto, comienzan a pasar a estado muerto, dejando a la red con un menor número de nodos, que pueden afectar el funcionamiento normal de la red.

Con respecto a las gráficas obtenidas en los dos primeros escenarios, no se ha alcanzado el punto de equilibrio de los sensores infectados y los muertos. Sin embargo, el tercer escenario logra alcanzar un punto de equilibrio libre de infección.

En resumen, la debilidad de la topología de malla es que todos los nodos pueden comunicarse entre sí, esto contribuye a la rápida propagación del malware, y además de esto, las características computacionales de los sensores pueden permitir que la propagación alcance a un mayor número de nodos, como se ha visto que ocurrió en el caso del segundo escenario.

En la topología de tipo estrella, todos los nodos se comunican a través del nodo receptor. En este caso, la infección puede propagarse rápidamente si el nodo receptor ha sido infectado. Sin embargo, cuando se ha infectado un nodo sensor es probable que el malware no pueda propagarse a otros nodos de la red.

En los tres escenarios se ha observado que la propagación del malware no ha podido avanzar significativamente en la red. Por lo tanto, los tres escenarios han alcanzado el punto libre de equilibrio.

En resumen, en la topología en estrella es necesario estudiar el comportamiento del malware cuando el nodo receptor es infectado, debido a que, al ser un nodo específico en la red, resulta más difícil que sea el primer nodo infectado, ya que este suele escogerse al azar.

Finalmente, la topología que puede facilitar una infección rápida de la red es la topología de malla. Sin embargo, esta topología requiere muchos recursos para funcionar correctamente, debido a que un nodo está conectado con todos los demás nodos de la red, por lo que la infección con malware puede reducir drásticamente la funcionalidad de la red. La topología en estrella puede considerarse difícil de infectar, siempre que las características del nodo receptor sean altas. Por último, la topología híbrida tiene la ventaja de que la infección puede detenerse en una pequeña sección de la red, evitando comprometer la funcionalidad de todo el sistema.

## 5.3. Resultados en redes complejas

En una red libre de escala, el número de enlaces en un nodo es diferente para cada uno de ellos. Por este motivo, un nodo con un gran número de enlaces puede ser capaz de propagar el malware más rápidamente que un nodo con pocos enlaces.

En este caso, la propagación del malware se inició lentamente hasta alcanzar el máximo de infectados, de igual forma, fueron aumentando los sensores en estado expuesto y recuperado. Sin embargo, se aprecia que casi al final de la simulación, los sensores susceptibles comienzan a pasar a estado muerto (ver la Figura 4.6).

En una red de mundo pequeño, dos nodos no vecinos pueden comunicarse entre sí a través de otros nodos vecinos, con un pequeño número de saltos. En este caso, la propagación del malware se desarrolla lentamente, con un número similar de sensores en estados infectado y expuesto. Posteriormente, estos sensores van cambiando su estado a recuperado o a muerto. Al finalizar la propagación, se aprecia que el número de sensores que han quedado en estado susceptible se mantiene hasta el final de la simulación (véase la Figura 4.7).

Con respecto a las gráficas, en la red libre de escala los sensores en estado infectado, expuesto y recuperado alcanzan el punto de equilibrio estable, mientas que los sensores susceptibles y muertos no han alcanzado este punto al finalizar la simulación; mientras que en la red de mundo pequeño, solo los sensores en estado susceptible han alcanzado el punto de equilibrio.

La red libre de escala ha obtenido una menor tasa de infección con respecto a la red de mundo pequeño, esto puede deberse al modo como están interconectados los sensores con sus vecinos. Esto también puede observarse en los puntos de equilibrio, ya que en la red libre de escala los sensores susceptibles no alcanzan este punto de equilibrio, y por el contrario, en la red de mundo pequeño, solo los sensores susceptibles alcanzan este punto de equilibrio.

# Capítulo 6

# Conclusiones, aportaciones y trabajos futuros

El análisis de las vulnerabilidades de un sistema informático tiene como finalidad mitigar o reducir los riesgos a los que se encuentra expuesto dicho sistema, con el propósito de proteger la información que se almacena en los servidores o los datos que se transmiten a través de la red informática. Por esta razón, cada vez son más frecuentes los estudios de ciberseguridad para distintos entornos, como es el caso de las redes de sensores inalámbricas, formadas por microsensores cuya finalidad es la monitorización de distintos fenómenos físicos.

El objetivo de esta memoria ha sido estudiar la propagación de malware en las redes de sensores inalámbricas, con el propósito de mejorar los mecanismos de ciberseguridad que pueden ser aplicados a este tipo de redes. Para satisfacer este objetivo se han analizado diferentes modelos de propagación de malware que han sido propuestos por otros autores.

Como resultado del análisis bibliográfico realizado, se han identificado las herramientas matemáticas utilizadas en dichos modelos, que han sido principalmente los autómatas celulares, las cadenas de Markov y los sistemas de ecuaciones diferenciales ordinarias y en derivadas parciales. Entre los trabajos estudiados, el desarrollado por Nwokoye ha resultado especialmente interesante puesto que propuso un modelo que incluía un sistema de ecuaciones diferenciales ordinarias junto con un modelo basado en agentes. Los modelos basados en agentes son aquellos en los que los individuos (agentes) se consideran entidades únicas y autónomas, que interactúan entre sí y con su entorno local. Además, los sistemas estudiados plantearon modelos globales o individuales; es importante resaltar que en el caso de los modelos individuales, el enfoque se centra principalmente en los sensores y el malware, y en pocos casos se considera la topología como una característica del modelo.

Este análisis bibliográfico ha permitido también llegar a la conclusión de que los modelos basados en agentes pueden ser útiles para crear un entorno más realista, donde se pueda estudiar el comportamiento del código malicioso en las redes de sensores inalámbricas a partir de las características individuales de los agentes. También se han analizado y definido nuevos agentes que intervienen en la propagación del malware y se han detallado sus características más significativas.

Consecuentemente, se ha definido un modelo matemático utilizando el paradigma del modelo basado en agentes y teniendo como base la epidemiología matemática. Concretamente se ha propuesto el modelo basado en agentes SEIRS-D, donde la $S$ corresponde a los sensores susceptibles, $E$ son los sensores expuestos, $I$ los sensores infectados, $R$ los sensores recuperados y $D$ los sensores en estado muerto.

El modelo propuesto en esta memoria se ha planteado como un modelo de tipo individual, puesto que los individuos que forman el sistema son considerados como entidades autónomas. Además, este modelo SEIRS-D es un modelo discreto, debido a que las interacciones entre los individuos y el entorno se definen a través de reglas de comportamiento, y también como un modelo estocástico, que ha utilizado los modelos basados en agentes como herramienta matemática.

La simulación epidemiológica a través de métodos estocásticos puede utilizar diferentes herramientas matemáticas como las ecuaciones diferenciales estocásticas, las cadenas de Markov o los modelos basados en agentes. La principal diferencia entre estos métodos es la cantidad de parámetros o características que pueden manejar; tanto las ecuaciones diferenciales estocásticas como las cadenas de Markov pueden presentar limitaciones para el análisis de poblaciones heterogéneas y en red. Sin embargo, los modelos basados en agentes permiten añadir las características necesarias y específicas para cada individuo, y definir poblaciones heterogéneas y en red.

En el caso de los modelos basados en agentes es necesario definir un número de parámetros adecuado; es decir, aquellos parámetros que intervienen en el proceso a modelar, para que el resultado sea lo más preciso posible. En el modelo basado en agentes, SEIR-V, de Nwokoye, se definieron únicamente dos tipos de agentes, por un lado los sensores, que pueden estar en los estados susceptible, expuesto, recuperado y vacunado; y por otra parte el malware, que corresponde a los agentes sensores en estado infectado. Además de estos agentes y relacionado con el entorno, se utilizó en este modelo una red de topología híbrida, es decir, una red organizada en clústeres y se consideró que los factores que determinan el comportamiento de los agentes incluían movimiento, muerte, envejecimiento, duración de la vida y duración de la recuperación temporal (inmunidad). Asimismo, emplearon otros parámetros previamente definidos en los modelos basados en ecuaciones diferenciales, como

el tiempo de infección del gusano, la duración de la exposición, la tasa de recuperación y la tasa de vacunación.

En el modelo SEIRS-D propuesto se han establecido seis tipos de agentes, siete coeficientes y seis reglas de transición. Los tipos de agentes que se han considerado son los sensores, el malware, la topología de red, el fenómeno de interés, la acción humana y los dispositivos. Los coeficientes que se han utilizados son el de infección, de transmisión, de detección, de recuperación, de mantenimiento, y de energía y malware. Por último, las reglas de transición de estado que se han determinado son: de susceptible a infectado, de susceptible a expuesto, de infectado a muerto, de infectado a recuperado, de expuesto a recuperado y de recuperado a susceptible.

Por otro lado, el sistema que se ha empleado para la implementación y simulación del modelo SEIRS-D es el entorno de trabajo Mesa. Este entorno utiliza el lenguaje de programación Python, por lo que ha podido adaptarse a las necesidades de la simulación. Una ventaja importante de este entorno es que la visualización de los resultados se muestra tanto gráfica como numéricamente, permitiendo que en cada paso se pueda analizar el número de sensores que se encuentran en los diferentes compartimentos.

Todas estas características mencionadas han servido de punto de partida y han permitido establecer el modelo SEIRS-D. En concreto, en este trabajo que aquí se presenta se ha incluido una descripción y clasificación detallada de los agentes, no solo considerando los sensores y el propio código malicioso, sino también el resto de actores. Además, en este caso, se han incluido las características individuales de los componentes esenciales de la red, como es el caso del agente malware que posee características como pueden ser su tipología, el mecanismo de propagación que utiliza o el objetivo del propio malware.

La definición del entorno en el modelo SEIRS-D se ha realizado teniendo en cuenta el fenómeno de interés a monitorizar y además la topología de red, que define la forma en la que se interconectan los sensores. Asimismo, en la descripción del modelo se han detallado los coeficientes que se encargan de agrupar las características de los individuos y también las reglas de transición, que definen de forma clara el momento en el que un agente sensor debe pasar de un estado compartimental a otro.

Una de las principales ventajas del modelo propuesto en esta memoria es la capacidad de predicción, que supone una mejora con respecto a otras propuestas anteriores relacionadas con el paradigma de modelo basado en agentes, y no solo con respecto a los enfoques tradicionales de la propagación de malware, que se basaban en ecuaciones diferenciales. Además, el modelo planteado considera un número adecuado de características para evaluar eficientemente el comportamiento de la red de sensores inalámbrica y sus componentes, cuando la red ha sido infectada por código malicioso.

Para las simulaciones que se han incluido en esta memoria, se han identificado las topologías y los entornos más vulnerables en los que el malware puede propagarse con mayor facilidad; esto ha permitido ajustar los parámetros a situaciones específicas para obtener y analizar diferentes resultados. El análisis de topologías en diferentes entornos, como pueden ser el militar e industrial, el de los fenómenos de atención médica y medioambiente, o los fenómenos en actividades diarias y multimedia, ha permitido observar el comportamiento del malware en contextos reales. Otra ventaja relevante de las simulaciones realizadas es la posibilidad de visualizar en tiempo real los estados de los compartimentos de los agentes, en cada instante de tiempo.

La desventaja más significativa de las simulaciones del modelo SEIRS-D presentadas ha sido que no se ha logrado obtener datos reales de una red en funcionamiento existente, que permitiese analizar otras características que pueden influir en los resultados. Algunos estudios sugieren la implementación de herramientas se seguridad como los Honeypot para redes de sensores inalámbricas, que permiten la recolección de datos en tiempo de real de un ataque malicioso, en un entorno controlado.

Los resultados de las simulaciones del modelo han sido analizados para diferentes escenarios, para tres topologías diferentes y además en redes complejas. El escenario que consideraba como fenómeno de interés las actividades diarias y multimedia, ha sido el que ha obtenido mejores resultados, ya que la propia red de sensores inalámbrica ha logrado evitar la propagación del malware o se han recuperado rápidamente de la infección. Por el contrario, en el resto de escenarios, el malware se ha propagado rápidamente; esto puede ser consecuencia de la acción humana en la red, que juega un papel determinante para detectar tempranamente una infección de malware, y que puede realizar los mantenimientos tanto de hardware como de software para implementar las medidas de seguridad necesarias que eviten la propagación.

En cuanto a las topologías utilizadas en las simulaciones, la topología en estrella se puede considerar como la topología más difícil de infectar; sin embargo, esta topología es una de las menos utilizadas en las redes de sensores inalámbricas actuales. Contrariamente, la topología de malla puede facilitar la propagación del malware a gran parte de la red debido a la alta interconexión entre los nodos. Por último, la topología híbrida ha demostrado buenos resultados para contener la propagación en una pequeña parte de la red.

Sin embargo,las conexiones de los nodos en las redes complejas son diferentes y dependen de varios factores. En el caso de la red libre de escala, si se infecta un nodo que cuenta con un gran número de enlaces hacia otros nodos, es posible que el malware se propague fácilmente a gran parte de la red; pero no ocurre lo mismo si el que se infecta es un nodo con pocos enlaces. El caso de las redes de mundo pequeño es diferente, puesto que la propagación

puede desarrollarse lentamente y alcanzar un número importante de nodos de la red, debido a la conexión a través de saltos entre nodos no vecinos.

En todos los casos analizados se pueden realizar nuevas simulaciones modificando los diferentes parámetros del sistema, como las características computacionales de los sensores, el nivel de acción humana, el mecanismo de propagación del malware o el nivel de riesgo de infección por dispositivos; esto permitiría encontrar los diferentes puntos de equilibrio en cada caso y realizar comparaciones entre ellos.

En conclusión, se ha obtenido como resultado que tanto los fenómenos de interés de los diferentes escenarios, como las topologías de red, influyen en el proceso de propagación del malware. Asimismo, las características computacionales de los sensores pueden contribuir a evitar una rápida propagación, debido a que es posible que los sensores con altas características computacionales tengan algún tipo de mecanismo de seguridad implementado. Finalmente, la acción humana interviene de manera significativa en el mantenimiento de la red y la prevención de una infección y posterior propagación del malware.

La simulación en el entorno de trabajo Mesa ha presentado limitaciones en la memoria RAM. La memoria ha tenido que ser incrementada cuando el número de nodos ha aumentado. Además, es necesario eliminar con frecuencia la caché del programa, puesto que puede cargar el disco duro con información no útil.

Como trabajo futuro, se propone el uso de algoritmos de aprendizaje automático como herramienta para la simulación y análisis del modelo propuesto, esto permitiría el análisis de topologías complejas con un uso eficiente de memoria. Igualmente, se recomienda la adquisición de un conjunto de datos en escenarios reales. Por último, las herramientas de aprendizaje automático y de *data science*, que se encarga de extraer información a partir de un gran volumen de datos, conocido como *big data*, pueden proporcionar otros resultados con una tasa de error baja.

# Bibliografía


1. Hou, Y. y Wang, J. *Investigation of Wireless Sensor Network of the Internet of Things* en *Advances in Intelligent, Interactive Systems and Applications* (eds. Xhafa, F., Patnaik, S. y Tavana, M.) (Springer, London, UK, 2019), 21-29.
2. Mostafaei, H. y Shojafar, M. A New Meta-heuristic Algorithm for Maximizing Lifetime of Wireless Sensor Networks. *Wireless Personal Communications* **82,** 723-742 (2015).
3. Yick, J., Mukherjee, B. y Ghosal, D. Wireless sensor network survey. *Computer networks* **52,** 2292-2330 (2008).
4. Fahmy, H. M. A. en *Wireless Sensor Networks: Energy Harvesting and Management for Research and Industry* 3-39 (Springer, Cham, Switzerland, 2020).
5. Yang, K. *Wireless sensor networks: Principles, Design and Applications* 1.ª ed. (Springer, London, UK, 2014).
6. Taheri, R. *y col.* Similarity-based Android malware detection using Hamming distance of static binary features. *Future Generation Computer Systems* **105,** 230-247 (2020).
7. Brauer, F. Mathematical epidemiology: Past, present, and future. *Infectious Disease Modelling* **2,** 113-127 (2017).
8. Kermack, W. O. y Mckendrick, A. G. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. A* **115,** 700-721 (1927).
9. Liu, B. *y col.* Malware Propagations in Wireless Ad Hoc Networks. *IEEE Transactions on Dependable and Secure Computing* **15,** 1016-1026 (2018).
10. Wu, X., Cao, Q., Jin, J., Li, Y. y Zhang, H. Nodes Availability Analysis of NB-IoT Based Heterogeneous Wireless Sensor Networks under Malware Infection. *Wireless Communications and Mobile Computing* **2019** (2019).
11. Nwokoye, C. y Umeh, I. Analytic-agent cyber dynamical systems analysis and design method for modeling spatio-temporal factors of malware propagation in wireless sensor networks. *MethodsX* **5,** 1373-1398 (2018).
12. Wang, Y., Li, D. y Dong, N. Cellular automata malware propagation model for WSN based on multi-player evolutionary game. *IET Networks* **7,** 129-135 (2018).
13. Arifin, S. M. N., Madey, G. R. y Collins, F. H. *Spatial agent-based simulation modeling in public health: design, implementation, and applications for malaria epidemiology* 1.ª ed. (John Wiley & Sons, Ltd, Hoboken, NJ, USA, 2016).
14. Helbing, D. *Social Self-Organization: Agent-Based Simulations and Experiments to Study Emergent Social Behavior* (Springer, Berlin, Heidelberg, 2012).

15. Wurzer, G., Kowarik, K. y Reschreiter, H. *Agent-based Modeling and Aimulation in Archaeology* (Springer, Cham, Switzerland, 2015).
16. Chu, Z., Yang, B., Ha, C. Y. y Ahn, K. Modeling GDP fluctuations with agent-based model. *Physica A: Statistical Mechanics and its Applications* **503,** 572-581 (2018).
17. Anderson, T. M. y Dragićević, S. Network-agent based model for simulating the dynamic spatial network structure of complex ecological systems. *Ecological Modelling* **389,** 19-32 (2018).
18. Jindal, A. y Rao, S. *Agent-based modeling and simulation of mosquito-borne disease transmission* en *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems* (International Foundation for Autonomous Agents y Multiagent Systems, Sao Paulo, Brazil, 8–12 May, 2017), 426-435.
19. Kaplan, M., Manore, C. A. y Bagamian, K. H. Agent-based hantavirus transmission model incorporating host behavior and viral shedding heterogeneities derived from field transmission experiments. *Letters in Biomathematics* **3,** 209-228 (2016).
20. Martín del Rey, A., Guillén, J. H. y Sánchez, G. R. *Modeling Malware Propagation in Wireless Sensor Networks with Individual-Based Models* en *Conference of the Spanish Association for Artificial Intelligence* (eds. Luaces, O. *y col.*) (Springer, Cham, Switzerland, 2016), 194-203.
21. Mainetti, L., Patrono, L. y Vilei, A. *Evolution of wireless sensor networks towards the internet of things: A survey* en *SoftCOM 2011, 19th international conference on software, telecommunications and computer networks* (2011), 1-6.
22. Selmic, R. R., Phoha, V. V. y Serwadda, A. *Wireless Sensor Networks: security, coverage, and localization* (Springer, Cham, Switzerland, 2016).
23. Benhaddou, D. y Al-Fuqaha, A. *Wireless Sensor and Mobile Ad-Hoc Networks: Vehicular and Space Applications* 1.ª ed. (Springer, 2015).
24. Kumar, T. y Mane, P. *ZigBee topology: A survey* en *2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)* (2016), 164-166.
25. Fahmy, H. M. A. *Wireless Sensor Networks: Concepts, Applications, Experimentation and Analysis* 1.ª ed. (Springer, 2016).
26. Siddiqui, S. y Fatima, S. *SPIN Protocol for transmission of data of mobile sink in Wireless Sensor Network* en *IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS)* **7** (2017).
27. Priyadarshi, R., Soni, S. K. y Sharma, P. en *Nanoelectronics, Circuits and Communication Systems* 289-297 (Springer, 2019).
28. Jamal, T. y Butt, S. A. Low-energy adaptive clustering hierarchy (LEACH) enhancement for military security operations. *Journal of Basic and Applied Scientific Research,* 2090-4304 (2017).
29. Ali, S., Al Balushi, T., Nadir, Z. y Hussain, O. K. en *Cyber Security for Cyber Physical Systems* 65-87 (Springer International Publishing, Cham, Switzerland, 2018).
30. Siegfried, R. *Modeling and simulation of complex systems: A framework for efficient agent-based modeling and simulation* (Springer, 2014).

31. Chong, C.-Y. y Kumar, S. P. Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE* **91,** 1247-1256 (2003).
32. Kashyap, R. en *IoT and WSN Applications for Modern Agricultural Advancements: Emerging Research and Opportunities* 8-40 (IGI Global, 2020).
33. Kaiwen, C., Kumar, A., Xavier, N. y Panda, S. K. *An intelligent home appliance control-based on WSN for smart buildings* en *2016 IEEE International Conference on Sustainable Energy Technologies (ICSET)* (2016), 282-287.
34. Ali, A., Ming, Y., Chakraborty, S. e Iram, S. A comprehensive survey on real-time applications of WSN. *Future internet* **9,** 77 (2017).
35. Batista, F. K., Martin del Rey, A. y Queiruga-Dios, A. *Malware propagation software for wireless sensor networks* en *Trends in Cyber-Physical Multi-Agent Systems. The PAAMS Collection - 15th International Conference, PAAMS 2017* (eds. De la Prieta, F. *y col.*) **619** (Springer, Cham, Switzerland, 2017), 238-241.
36. Sohn, I. Small-world and scale-free network models for IoT systems. *Mobile Information Systems* **2017** (2017).
37. Dinh, N.-T. y Kim, Y. Auto-Configuration in Wireless Sensor Networks: A Review. *Sensors* **19,** 4281 (2019).
38. Ketshabetswe, L. K., Zungeru, A. M., Mangwala, M., Chuma, J. M. y Sigweni, B. Communication protocols for wireless sensor networks: A survey and comparison. *Heliyon* **5,** e01591 (2019).
39. Mahara, R., Meena, P. y Agrawal, C. Survey on Energy Efficient Clustering and Routing Protocols of Wireless Sensor Network. *International Journal of Scientific Research & Engineering Trends* **5,** 1177-1184 (2019).
40. Grover, J., Sharma, M. *y col. Optimized GAF in wireless sensor network* en *Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization* (2014), 1-6.
41. Elhoseny, M. y Hassanien, A. E. en *Dynamic Wireless Sensor Networks: New Directions for Smart Technologies* 115-143 (Springer, Cham, Switzerland, 2019).
42. Oreku, G. S. y Pazynyuk, T. *Security in wireless sensor networks* (Springer, 2016).
43. Conti, M. *Secure wireless sensor networks* (Springer, 2015).
44. Alsmadi, I. M., Karabatis, G. y Aleroud, A. *Information fusion for cyber-security analytics* (Springer, 2017).
45. Kumar, M. H., Mohanraj, V., Suresh, Y., Senthilkumar, J. y Nagalalli, G. Trust aware localized routing and class based dynamic block chain encryption scheme for improved security in WSN. *Journal of Ambient Intelligence and Humanized Computing* (2020).
46. Tawalbeh, H., Hashish, S., Tawalbeh, L. y Aldairi, A. Security in Wireless Sensor Networks Using Lightweight Cryptography. *Journal of Information Assurance & Security* **12** (2017).
47. Karlof, C., Sastry, N. y Wagner, D. *TinySec: a link layer security architecture for wireless sensor networks* en *Proceedings of the 2nd international conference on Embedded networked sensor systems* (2004), 162-175.

48. Mbarek, B. y Meddeb, A. *Energy efficient security protocols for wireless sensor networks: SPINS vs TinySec* en *2016 International Symposium on Networks, Computers and Communications (ISNCC)* (2016), 1-4.

49. Park, T. y Shin, K. G. LiSP: A lightweight security protocol for wireless sensor networks. *ACM Transactions on Embedded Computing Systems (TECS)* **3,** 634-660 (2004).

50. Khanji, S., Iqbal, F. y Hung, P. *ZigBee Security Vulnerabilities: Exploration and Evaluating* en *2019 10th International Conference on Information and Communication Systems (ICICS)* (2019), 52-57.

51. Shaikh, R. A., Lee, S., Khan, M. A. y Song, Y. J. *LSec: lightweight security protocol for distributed wireless sensor network* en *IFIP International Conference on Personal Wireless Communications* (2006), 367-377.

52. Tripathy, S. *LISA: lightweight security algorithm for wireless sensor networks* en *International Conference on Distributed Computing and Internet Technology* (2007), 129-134.

53. Luk, M., Mezzour, G., Perrig, A. y Gligor, V. *MiniSec: a secure sensor network communication architecture* en *2007 6th International Symposium on Information Processing in Sensor Networks* (2007), 479-488.

54. Ren, J., Li, T. y Aslam, D. *A power efficient link-layer security protocol (LLSP) for wireless sensor networks* en *MILCOM 2005-2005 IEEE Military Communications Conference* (2005), 1002-1007.

55. Razak, S. A., Furnell, S. y Brooke, P. Attacks against mobile ad hoc networks routing protocols. *Network Research Group, University of Plymouth, Devon* (2004).

56. Dinker, A. G. y Sharma, V. *Attacks and challenges in wireless sensor networks* en *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (2016), 3069-3074.

57. Slade, R. M. en *Information Security Management Handbook* 1736-1780 (Taylor & Francis, 2010).

58. Ab Razak, M. F., Anuar, N. B., Salleh, R. y Firdaus, A. The rise of "malware": Bibliometric analysis of malware study. *Journal of Network and Computer Applications* **75,** 58-76 (2016).

59. Hoefelmeyer, R. y Phillips, T. E. en *Encyclopedia of Information Assurance* 1814-1825 (Taylor & Francis, 2011).

60. Aycock, J. *Computer viruses and malware* (Springer, 2006).

61. Goyal, M. S. *y col.* Preventive Measures from Virus Attack for Energy Optimization in Wireless Sensor Network. *International Journal of Research Studies in Computer Science and Engineering* **4,** 1-8 (2017).

62. Singh, A., Awasthi, A. K., Singh, K. y Srivastava, P. K. Modeling and analysis of worm propagation in wireless sensor networks. *Wireless Personal Communications* **98,** 2535-2551 (2018).

63. Jalalitabar, M., Valero, M. y Bourgeois, A. G. *Demonstrating the Threat of Hardware Trojans in Wireless Sensor Networks* en *2015 24th International Conference on Computer Communication and Networks (ICCCN)* (2015), 1-8.

64. Gupta, K. y Shukla, S. *Internet of Things: Security challenges for next generation networks* en *2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)* (2016), 315-318.

65. Abirami, S. *A Complete Study on the Security Aspects of Wireless Sensor Networks* en *International Conference on Innovative Computing and Communications* (eds. Bhattacharyya, S., Hassanien, A. E., Gupta, D., Khanna, A. y Pan, I.) (Springer, Singapore, 2019), 223-230.

66. Acarali, D., Rajarajan, M., Komninos, N. y Zarpelão, B. B. Modelling the Spread of Botnet Malware in IoT-Based Wireless Sensor Networks. *Security and Communication Networks* **2019** (2019).

67. Fu, D. y Peng, X. TPM-based remote attestation for Wireless Sensor Networks. *Tsinghua Science and Technology* **21,** 312-321 (2016).

68. Pradeep, P., Kj, S. *y col.* en *Information Science and Applications (ICISA) 2016* 715-727 (Springer, 2016).

69. Azmoodeh, A., Dehghantanha, A., Conti, M. y Choo, K.-K. R. Detecting crypto-ransomware in IoT networks based on energy consumption footprint. *Journal of Ambient Intelligence and Humanized Computing* **9,** 1141-1152 (2018).

70. Cohen, F. Computer viruses: theory and experiments. *Computers & security* **6,** 22-35 (1987).

71. Jimeno García, M., Míguez Pérez, C., Matas García, A. y Pérez Agudín, J. *Destripa la red* (Anaya Multimedia, 2007).

72. Xiang, Y., Fan, X., Zhu, W. *y col.* Propagation of active worms: a survey. *International journal of computer systems science & engineering* **24** (2009).

73. Bettany, A. y Halsey, M. *Windows virus and malware troubleshooting* (Springer, 2017).

74. Zhu, Z. *y col. Botnet research survey* en *2008 32nd Annual IEEE International Computer Software and Applications Conference* (2008), 967-972.

75. Riley, R., Jiang, X. y Xu, D. *Multi-aspect profiling of kernel rootkit behavior* en *Proceedings of the 4th ACM European conference on Computer systems* (2009), 47-60.

76. O'Gorman, G. y McDonald, G. *Ransomware: A growing menace* (Symantec Corporation, 2012).

77. Europol. *Internet Organised Crime Threat Assessment (IOCTA 2017)* inf. téc. (European Union Agency for Law Enforcement Cooperation (Europol), 2017).

78. Railsback, S. F. y Grimm, V. *Agent-based and individual-based modeling: a practical introduction* (Princeton university press, 2011).

79. García-Valdecasas, J. I. *Simulación basada en agentes. Introducción a Netlogo* (CIS-Centro de Investigaciones Sociológicas, 2016).

80. Vynnycky, E. y White, R. *An introduction to infectious disease modelling* (OUP oxford, 2010).

81. Delamater, P. L., Street, E. J., Leslie, T. F., Yang, Y. T. y Jacobsen, K. H. Complexity of the basic reproduction number (R0). *Emerging infectious diseases* **25,** 1 (2019).

82. Krone, S. Spatial models: stochastic and deterministic. *Mathematical and Computer Modelling* **40,** 393-409 (2004).

83. Fresnadillo Martıénez, M. J., Garcıéa-Sánchez, E., Garcıéa-Merino, E., Martıén del Rey, Á. y Garcıéa-Sánchez, J. E. Modelización matemática de la propagación de enfermedades infecciosas: de dónde venimos y hacia dónde vamos. *Revista Española de Quimioterapia* **26** (2013).

84. Kiss, I. Z., Miller, J. C., Simon, P. L. *y col. Mathematics of epidemics on networks* (Springer, 2017).

85. Zhang, Z. y Si, F. Dynamics of a delayed SEIRS-V model on the transmission of worms in a wireless sensor network. *Advances in Difference Equations,* 295 (2014).

86. Chizari, H. y Zulkurnain, A. U. Modelling malware response in wireless sensor networks using stochastic cellular automata. *Journal of Mobile, Embedded and Distributed Systems* **6,** 159-166 (2014).

87. Li, Q., Zhang, B., Cui, L., Fan, Z. y Athanasios, V. V. Epidemics on small worlds of tree-based wireless sensor networks. *Journal of Systems Science and Complexity* **27,** 1095-1120 (2014).

88. Keshri, N. y Mishra, B. K. Optimal Control Model for Attack of Worms in Wireless Sensor Network. *International Journal of Grid and Distributed Computing* **7,** 251-272 (2014).

89. Keshri, N. y Mishra, B. K. Two time-delay dynamic model on the transmission of malicious signals in wireless sensor network. *Chaos, Solitons & Fractals* **68,** 151-158 (2014).

90. Feng, L., Song, L., Zhao, Q. y Wang, H. Modeling and stability analysis of worm propagation in wireless sensor network. *Mathematical Problems in Engineering* **2015,** 1-8 (2015).

91. Hu, J. y Song, Y. *The model of malware propagation in wireless sensor networks with regional detection mechanism* en *China Conference on Wireless Sensor Networks* (2014), 651-662.

92. Zhu, L. y Zhao, H. Dynamical analysis and optimal control for a malware propagation model in an information network. *Neurocomputing* **149,** 1370-1386 (2015).

93. Zhu, L., Zhao, H. y Wang, X. Stability and bifurcation analysis in a delayed reaction-diffusion malware propagation model. *Computers & Mathematics with Applications* **69,** 852-875 (2015).

94. Shen, S. *y col.* Reliability evaluation for clustered WSNs under malware propagation. *Sensors* **16** (2016).

95. Shen, S. *y col.* A non-cooperative non-zero-sum game-based dependability assessment of heterogeneous WSNs with malware diffusion. *Journal of Network and Computer Applications* **91,** 26-35 (2017).

96. Wang, T. *y col.* Propagation modeling and defending of a mobile sensor worm in wireless sensor and actuator networks. *Sensors* **17** (2017).

97. Batista, F. K., Martin del Rey, A., Quintero-Bonilla, S. y Queiruga-Dios, A. *A SEIR Model for Computer Virus Spreading Based on Cellular Automata* en *International Joint Conference SOCO'17-CISIS'17-ICEUTE'17* (eds. Pérez García, H., Alfonso-Cendón, J., Sánchez González, L., Quintián, H. y Corchado, E.) **649** (Springer, Cham, Switzerland, 2018), 641-650.
98. Shen, S., Zhou, H., Feng, S., Liu, J. y Cao, Q. SNIRD: Disclosing Rules of Malware Spread in Heterogeneous Wireless Sensor Networks. *IEEE Access* **7,** 92881-92892 (2019).
99. Queiruga-Dios, A., Encinas, A. H., Martıén-Vaquero, J. y Encinas, L. H. *Malware propagation models in wireless sensor networks: a review* en *International Joint Conference SOCO'16-CISIS'16-ICEUTE'16* (eds. Graña, M. *y col.*) **527** (Springer, Cham, Switzerland, 2017), 648-657.
100. Amouroux, E., Desvaux, S. y Drogoul, A. *Towards virtual epidemiology: an agent-based approach to the modeling of H5N1 propagation and persistence in North-Vietnam* en *Pacific Rim International Conference on Multi-Agents* (Berlin, Heidelberg, 2008), 26-33.
101. Cliff, O. M. *y col.* Investigating spatio-temporal dynamics and synchrony of influenza epidemics in Australia: An agent-based modelling approach. *Simulation Modelling Practice and Theory* **87,** 412-431 (2018).
102. Gharakhanlou, N. M., Mesgari, M. S. y Hooshangi, N. Developing an agent-based model for simulating the dynamic spread of Plasmodium vivax malaria: A case study of Sarbaz, Iran. *Ecological Informatics* **54,** 101006 (2019).
103. Bose, A. y Shin, K. G. Agent-based modeling of malware dynamics in heterogeneous environments. *Security and Communication Networks* **6,** 1576-1589 (2013).
104. Hosseini, S., Abdollahi Azgomi, M. y Rahmani Torkaman, A. Agent-based simulation of the dynamics of malware propagation in scale-free networks. *Simulation* **92,** 709-722 (2016).
105. Von Neumann, J. First Draft of a Report on the EDVAC, 1945. *Reprinted in The Origins of Digital Computers Selected Papers,* 355-364 (1975).
106. Conway, J. The game of life. *Scientific American* **223,** 4 (1970).
107. Schelling, T. C. Dynamic models of segregation. *Journal of mathematical sociology* **1,** 143-186 (1971).
108. Epstein, J. M. y Axtell, R. *Growing artificial societies: social science from the bottom up* (Brookings Institution Press, 1996).
109. Barnes, D. J. y Chu, D. en *Guide to Simulation and Modeling for Biosciences* 15-78 (Springer, London, 2015).
110. Van Dam, K., Nikolic, I. y Lukszo, Z. *Agent-Based Modelling of Socio-Technical Systems* (Springer, 2013).
111. Truszkowski, W. F. en *Agent Technology from a Formal Perspective* (eds. Rouff, C. A., Hinchey, M., Rash, J., Truszkowski, W. y Gordon-Spears, D.) 3-24 (Springer, London, 2006).

112. Shvecov, A., Dianov, S. y Zaripova, D. *Agent architecture implementation in models of socio-ecological-economic systems* en *Proceedings of the III International Scientific and Practical Conference* (2020), 1-4.

113. Cardoso, C., Bert, F. y Podestá, G. Modelos Basados en Agentes (MBA): definición, alcances y limitaciones. *Instituto Interamericano para la investigación del cambio global* (2011).

114. Heckbert, S., Baynes, T. y Reeson, A. Agent-based modeling in ecological economics. *Annals of the New York Academy of Sciences* **1185,** 39-53 (2010).

115. Ndii, M. Z., Djahi, B. S., Rumlaklak, N. D. y Supriatna, A. K. *Determining the Important Parameters of Mathematical Models of the Propagation of Malware* en *International Symposium of Information and Internet Technology* (2018), 1-9.

116. Abar, S., Theodoropoulos, G. K., Lemarinier, P. y O'Hare, G. M. Agent Based Modelling and Simulation tools: A review of the state-of-art software. *Computer Science Review* **24,** 13-33 (2017).

117. García Vázquez, J. C. y Sancho Caparrini, F. *NetLogo: Una herramienta de Modelado* (Editorial Universidad de Sevilla, 2016).

118. Kaveh, A. y Bakhshpoori, T. *Metaheuristics: Outlines, MATLAB Codes and Examples* (Springer, 2019).

119. Project Mesa Team. *Mesa: Agent-Based Modeling in Python 3+* 2018. https://github.com/projectmesa/mesa/.

120. Quezada, A. y Canessa, E. Modelado basado en agentes: una herramienta para complementar el análisis de fenómenos sociales. *Avances en Psicologıéa Latinoamericana* **28,** 226-238 (2011).

121. Thompson, J. R. *y col.* Interdependent Critical Infrastructure Model (ICIM): An agent-based model of power and water infrastructure. *International Journal of Critical Infrastructure Protection* **24,** 144-165 (2019).

122. Fitwi, A. H., Nagothu, D., Chen, Y. y Blasch, E. *A distributed agent-based framework for a constellation of drones in a military operation* en *2019 Winter Simulation Conference (WSC)* (2019), 2548-2559.

123. Echavarria, L. D., Valencia, A. y Bermúdez, J. Agent-based model for the analysis of technological acceptance of mobile learning. *IEEE Latin America Transactions* **15,** 1121-1127 (2017).

124. Gorochowski, T. E. Agent-based modelling in synthetic biology. *Essays in biochemistry* **60,** 325-336 (2016).

125. Barbuto, A., Lopolito, A. y Santeramo, F. G. Improving diffusion in agriculture: an agent-based model to find the predictors for efficient early adopters. *Agricultural and food economics* **7,** 1 (2019).

126. Vallier, K. Three concepts of political stability: an agent-based model. *Social Philosophy & Policy* **34,** 232 (2017).

127. Klein, D., Marx, J. y Fischbach, K. Agent-Based Modeling in Social Science, History, and Philosophy: An Introduction. *Historical Social Research / Historische Sozialforschung* **43,** 7-27 (abr. de 2018).

128. Cuevas, E. An agent-based model to evaluate the COVID-19 transmission risks in facilities. *Computers in Biology and Medicine,* 103827 (2020).

129. Ringler, P., Keles, D. y Fichtner, W. Agent-based modelling and simulation of smart electricity grids and markets–a literature review. *Renewable and Sustainable Energy Reviews* **57,** 205-215 (2016).

130. Garbey, M., Casarin, S. y Berceli, S. A. Vascular adaptation: pattern formation and cross validation between an agent based model and a dynamical system. *Journal of theoretical biology* **429,** 149-163 (2017).

131. Grant, T. J., Parry, H. R., Zalucki, M. P. y Bradbury, S. P. Predicting monarch butterfly (Danaus plexippus) movement and egg-laying with a spatially-explicit agent-based model: the role of monarch perceptual range and spatial memory. *Ecological Modelling* **374,** 37-50 (2018).

132. Zhuge, C. y Shao, C. Agent-based modelling of locating public transport facilities for conventional and electric vehicles. *Networks and Spatial Economics* **18,** 875-908 (2018).

133. Fuentes, R. y Sengupta, A. Using insurance to manage reliability in the distributed electricity sector: Insights from an agent-based model. *Energy Policy* **139,** 111251 (2020).

134. Münzberg, T., Müller, T. y Raskob, W. en *Urban Disaster Resilience and Security* 261-284 (Springer, 2018).

135. Masad, D. y Kazil, J. *MESA: an agent-based modeling framework* en *14th PYTHON in Science Conference* (2015), 53-60.

136. More, A. y Raisinghani, V. A survey on energy efficient coverage protocols in wireless sensor networks. *Journal of King Saud University-Computer and Information Sciences* **29,** 428-448 (2017).

137. Yang, Z., Yong, Y. y He, J. *Determining the Number of Nodes for Wireless Sensor Networks* en *Proceedings of the IEEE 6th Circuits and Systems Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication* **2** (2004), 501-504.

138. Ismaili, I. A. *y col. Comparative study of ZigBee and 6LoWPAN protocols* en *Third International Conference on Computing and Wireless Communication Systems ( ICCWCS)* (2019).

139. Lara-Cueva, R. A., Gordillo, R. y Poaquiza, V. *Determining the number of nodes in a volcano monitoring system by using WSN* en *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)* (2017), 1-6.

140. P., Đ. M., Z., T., G., D. y V., M. *A survey of military applications of wireless sensor networks* en *Mediterranean Conference on Embedded Computing (MECO)* (2012), 196-199.

141. Bollobás, B., Borgs, C., Chayes, J. y Riordan, O. *Directed scale-free graphs* en *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms* (2003), 132-139.

142. Barabási, A.-L. y Albert, R. Emergence of scaling in random networks. *science* **286,** 509-512 (1999).

143. Lee, K. D. y Hubbard, S. en *Data Structures and Algorithms with Python* 41-65 (Springer, Cham, Switzerland, 2015).

# Anexo A

# Publicaciones relacionadas con la temática de la Tesis

**2020**

. Batista, F. K., Martin del Rey, A. y Queiruga-Dios, A. A New Individual-Based Model to Simulate Malware Propagation in Wireless Sensor Networks. *Mathematics* **8,** 410 (2020).

Indicadores de calidad: factor de impacto 1.747, cuartil JCR Q1, 28/324 en la categoría Mathematics, año 2019.

Resumen: Las redes de sensores inalámbricas (WSN) son un conjunto de dispositivos de sensores desplegados en una zona determinada que forman una red sin una arquitectura preestablecida. Recientemente, los ataques de malware se han incrementado considerablemente y constituyen una vulnerabilidad potencial para el Internet de las Cosas, y por consiguiente para estas redes. La propagación de software malicioso en las redes de sensores inalámbricas se ha estudiado desde diferentes perspectivas, en la mayoría de los modelos propuestos excluyendo las características individuales. El objetivo principal de este estudio ha sido introducir un modelo basado en agentes para analizar la propagación de malware en las redes de sensores inalámbricas, detallando los agentes que intervienen, los coeficientes de la propagación y reglas de transición de estados. Por último, se incluyeron algunas simulaciones del modelo propuesto.

**2019**

. Batista, F. K., Martin del Rey, A. y Queiruga-Dios, A. *A Review of SEIR-D Agent-Based Model* en *Distributed Computing and Artificial Intelligence, 16th International*

*Conference, Special Sessions* (eds. Herrera-Viedma, E., Vale, Z., Nielsen, P., Martin Del Rey, A. y Casado Vara, R.) **1004** (Springer, Cham, Switzerland, 2019), 133-140.

Indicadores de calidad: factor de impacto 0.184, cuartil SJR Q3, en la categoría Computer Science (miscellaneous), año 2019.

Resumen: El malware es una vulnerabilidad potencial para el Internet de las Cosas; por esta razón, la propagación de malware en las redes de sensores inalámbricas ha sido estudiada desde diferentes perspectivas. Sin embargo, en la mayoría de los modelos propuestos no se han considerado las características individuales, motivo por el cual se ha planteado el utilizar modelos basados en agentes como herramientas matemática para analizar la propagación de malware. En este trabajo, se plantea un modelo basado en agentes a partir de tres elementos principales: agentes, entorno y reglas. Este artículo presenta además una revisión de modelos ya existentes, basados en agentes, para simular la propagación de malware en redes de sensores inalámbricas.

**2018**

. Batista, F. K., Martin del Rey, A., Quintero-Bonilla, S. y Queiruga-Dios, A. *A SEIR Model for Computer Virus Spreading Based on Cellular Automata* en *International Joint Conference SOCO'17-CISIS'17-ICEUTE'17* (eds. Pérez García, H., Alfonso-Cendón, J., Sánchez González, L., Quintián, H. y Corchado, E.) **649** (Springer, Cham, Switzerland, 2018), 641-650.

Indicadores de calidad: factor de impacto 0.174, cuartil SJR Q3, en la categoría Computer Science (miscellaneous), año 2018.

Resumen: Hay una gran variedad de tipos de malware: virus informáticos, gusanos, troyanos, etc. Actualmente, el malware es uno de los problemas de seguridad informática más importantes y la fuente de grandes pérdidas financieras. En consecuencia, es necesario diseñar herramientas que permitan simular el comportamiento de la propagación del malware. Estas herramientas se basan en modelos matemáticos y la mayoría de ellos abordan el estudio de un tipo particular de malware llamado gusanos informáticos. Sin embargo, hasta donde sabemos, hay pocos modelos dedicados al estudio de la propagación de los virus informáticos. En este sentido, el objetivo principal de este trabajo es introducir un nuevo modelo matemático, basado en autómatas celulares, para analizar el comportamiento epidemiológico de los virus informáticos. Concretamente, se trata de un modelo SEIR (Susceptible-Expuesto-Infectado-Recuperado) donde los nodos de la red se dividen en cuatro compartimentos: susceptibles, expuestos, infectados y recuperados.

**2017**

. Batista, F. K., Martin del Rey, A. y Queiruga-Dios, A. Malware propagation in Wireless Sensor Networks: global models vs Individual-based models. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal* **6,** 5-15 (2017).

Indicadores de calidad: indexada en el ESCI (Emerging Sources Citation Index)

Resumen: El objetivo principal de este estudio fue la propuesta de un nuevo entorno de trabajo para diseñar una novedosa familia de modelos matemáticos que permitan simular la propagación del malware en las redes de sensores inalámbricas. El análisis de los modelos existentes en la literatura científica revela que la mayoría de modelos de simulación de propagación de malware son modelos globales basados en sistemas de ecuaciones diferenciales ordinarias, de manera que no tienen en cuenta las características individuales de los sensores y sus interacciones locales. Este es un gran inconveniente cuando se consideran las WSN. Teniendo en cuenta las principales características de estas redes (elementos, topologías de la red, ciclo de vida de los nodos, etc.), se demuestra que los modelos basados en individuos son más adecuados para este fin que los globales. Se exponen las principales características de este nuevo tipo de modelos de propagación de malware para las WSN.

. Batista, F. K., Martin del Rey, A. y Queiruga-Dios, A. *Malware propagation software for wireless sensor networks* en *Trends in Cyber-Physical Multi-Agent Systems. The PAAMS Collection - 15th International Conference, PAAMS 2017* (eds. De la Prieta, F. *y col.*) **619** (Springer, Cham, Switzerland, 2017), 238-241.

Resumen: La infección por malware en una red de sensores inalámbricas (WSN) puede representar una vulnerabilidad potencial, debido al bajo nivel de seguridad que presentan estas redes. En consecuencia, es muy importante estudiar el comportamiento de la propagación del malware en una WSN. Este trabajo tiene como objetivo diseñar un novedoso modelo basado en agentes para simular la propagación del malware. Se proporcionará un software eficiente de gran ayuda para los administradores de seguridad.

# Anexo B

# Código fuente del proyecto

Este proyecto de software está compuesto por 4 ficheros principales. Para su desarrollo se ha tomado como base el proyecto *virus on network*, incluido en los ejemplos del Framework Mesa. Este proyecto se ha desarrollado en Python. Para la ejecución del programa desarrollado en este proyecto es necesario la instalación del paquete networkx.

## Fichero run.py

El fichero `run.py` se utiliza para iniciar el servidor y crear el proyecto, a través de la consola de Linux (ver Código B.1). Para esto, se ejecuta el comando `python3 run.py`.

```
1  from abm_model.server import server
2  server.launch()
```

Código B.1 Fichero run.py

## Fichero server.py

El fichero `server.py` se encarga de llamar y cargar las librerías del Framework Mesa (ver Código B.2). Además, crea el fichero html para la visualización del entorno en el navegador.

```
1  import math
2
3  from mesa.visualization.ModularVisualization import ModularServer
4  from mesa.visualization.UserParam import UserSettableParameter
5  from mesa.visualization.modules import ChartModule
6  from mesa.visualization.modules import NetworkModule
7  from mesa.visualization.modules import TextElement
8  from .model import wsnModel, State, sensors, malware, topology, phenomenon,
9      humans, devices, susceptible_counter, infected_counter, exposed_counter,
10     recovered_counter, dead_counter, sensor_counter, router_counter, sink_counter,
```

```
        sensor_susceptible_counter, sensor_infected_counter, sensor_exposed_counter,
        sensor_recovered_counter, sensor_dead_counter, router_susceptible_counter,
        router_infected_counter, router_exposed_counter, router_recovered_counter,
        router_dead_counter, sink_susceptible_counter, sink_infected_counter,
        sink_exposed_counter, sink_recovered_counter, sink_dead_counter

    def network_portrayal(G):

        def node_color(agent):
            return {
                State.INFECTED: '#FF0000',
                State.EXPOSED: '#0404B4',
                State.RECOVERED: '#610B5E',
                State.DEAD: '#000000',
                State.SUSCEPTIBLE: '#04B404'
            }.get(agent.state, '#808080')

        portrayal = dict()

        portrayal['nodes'] = [{'size': 6,
                               'color': node_color(agents[0]),
                               'tooltip': "id: {}<br>state: {}<br>type: {}".format(
                               agents[0].unique_id, agents[0].state.name, agents[0].
                               type),
                              }
                              for (_, agents) in G.nodes.data('agent')]

        portrayal['edges'] = [{'source': source,
                               'target': target,
                               'color': '#333333',
                               'width': 1.0,
                              }
                              for (source, target) in G.edges]

        return portrayal

    network = NetworkModule(network_portrayal, 600, 800, library='d3')
    chart = ChartModule([{'Label': 'Susceptible', 'Color': '#04B404'},
                {'Label': 'Infected', 'Color': '#FF0000'},
                        {'Label': 'Exposed', 'Color': '#0404B4'},
                        {'Label': 'Recovered', 'Color': '#610B5E'},
                {'Label': 'Dead', 'Color': '#000000'},])

    class MyTextElement(TextElement):
        def render(self, model):
            susceptible_text = str(susceptible_counter(model))
            infected_text = str(infected_counter(model))
            exposed_text = str(exposed_counter(model))
            recovered_text = str(recovered_counter(model))
            dead_text = str(dead_counter(model))
            sensor_text = str(sensor_counter(model))
            router_text = str(router_counter(model))
            sink_text = str(sink_counter(model))
            sensor_susceptible_text = str(sensor_susceptible_counter(model))
            sensor_infected_text = str(sensor_infected_counter(model))
```

```
66              sensor_exposed_text = str(sensor_exposed_counter(model))
67              sensor_recovered_text = str(sensor_recovered_counter(model))
68              sensor_dead_text = str(sensor_dead_counter(model))
69              router_susceptible_text = str(router_susceptible_counter(model))
70              router_infected_text = str(router_infected_counter(model))
71              router_exposed_text = str(router_exposed_counter(model))
72              router_recovered_text = str(router_recovered_counter(model))
73              router_dead_text = str(router_dead_counter(model))
74              sink_susceptible_text = str(sink_susceptible_counter(model))
75              sink_infected_text = str(sink_infected_counter(model))
76              sink_exposed_text = str(sink_exposed_counter(model))
77              sink_recovered_text = str(sink_recovered_counter(model))
78              sink_dead_text = str(sink_dead_counter(model))
79              total = str(sensor_counter(model)+router_counter(model)+sink_counter(model))
80
81              return "<table style='width:100%'><tr><th>State</th><th>Sensor</th>
82              <th>Router</th><th>Sink</th><th>Total</th></tr><tr><td>Susceptible</td>
83              <td>{}</td><td>{}</td><td>{}</td><td>{}</td></tr><tr><td>Infected</td>
84              <td>{}</td><td>{}</td><td>{}</td><td>{}</td></tr><tr><td>Exposed</td>
85              <td>{}</td><td>{}</td><td>{}</td><td>{}</td></tr><tr><td>Recovered</td>
86              <td>{}</td><td>{}</td><td>{}</td><td>{}</td></tr><tr><td>Dead</td>
87              <td>{}</td><td>{}</td><td>{}</td><td>{}</td></tr><tr><th>Total</th>
88              <td>{}</td><td>{}</td><td>{}</td><td>{}</td></tr></table>".format(
89              sensor_susceptible_text,router_susceptible_text,sink_susceptible_text,
90              susceptible_text,sensor_infected_text,router_infected_text,sink_infected_text,
91              infected_text,sensor_exposed_text,router_exposed_text,sink_exposed_text,
92              exposed_text,sensor_recovered_text,router_recovered_text,sink_recovered_text,
93              recovered_text,sensor_dead_text,router_dead_text,sink_dead_text,dead_text,
94              sensor_text,router_text,sink_text,total)
95
96      model_params = {
97          'num_nodes': UserSettableParameter('slider', 'Number of nodes', 50, 50, 1000, 100,
98                                             description='Choose how many sensors nodes to
99                                             include in the model'),
100         'topology_types': UserSettableParameter('choice', 'Topology of WSN',
101                                                 value='Hybrid', choices=list(wsnModel.
102                                                 topology_types.keys()), description='Choose
103                                                 the type of the topology'),
104         'phenomenon_types': UserSettableParameter('choice', 'Phenomenon of interest',
105                                                  value='Military/Industrial',
106                                                  choices=list(wsnModel.phenomenon_types.keys()),
107                                                  description='Choose the phenomenon of
108                                                  interest'),'any_infected': 0,
109     }
110
111     server = ModularServer(wsnModel, [network, MyTextElement(), chart], 'SEIR-D agent-based
112                            model', model_params)
113     server.port = 8824
```

Código B.2 Fichero server.py

# Fichero schedule.py

En el fichero `schedule.py` se definen los tipos de agentes, se asigna un id único para cada agente; además, controla las características de los agentes en cada instante de tiempo (ver Código B.3).

```
1   import random
2   from collections import defaultdict
3
4   from mesa.time import RandomActivation
5
6   class RandomActivationByBreed(RandomActivation):
7
8       def __init__(self, model):
9           super().__init__(model)
10          self.agents_by_breed = defaultdict(dict)
11
12      def add(self, agent):
13          self._agents[agent.unique_id] = agent
14          agent_class = type(agent)
15          self.agents_by_breed[agent_class][agent.unique_id] = agent
16
17      def remove(self, agent):
18          del self._agents[agent.unique_id]
19
20          agent_class = type(agent)
21          del self.agents_by_breed[agent_class][agent.unique_id]
22
23      def step(self, by_breed=True):
24          if by_breed:
25              for agent_class in self.agents_by_breed:
26                  self.step_breed(agent_class)
27              self.steps += 1
28              self.time += 1
29          else:
30              super().step()
31
32      def step_breed(self, breed):
33          agent_keys = list(self.agents_by_breed[breed].keys())
34          random.shuffle(agent_keys)
35          for agent_key in agent_keys:
36              self.agents_by_breed[breed][agent_key].step()
37
38      def get_breed_count(self, breed_class):
39          return len(self.agents_by_breed[breed_class].values())
40
41      def get_agent_breed(self, breed_class):
42          return self.agents_by_breed[breed_class].values()
```

Código B.3 Fichero schedule.py

# Fichero model.py

En el fichero `model.py` se definen los tipos de agentes con sus características, tal como se detalló en la Tabla 3.1, los coeficientes y las reglas de transición, que han sido definidos en el modelo SEIRS-D propuesto en este trabajo (ver Código B.4).

```
    import random
import math
from enum import Enum
import networkx as nx

from mesa import Agent, Model
from mesa.time import RandomActivation
from mesa.datacollection import DataCollector
from mesa.space import NetworkGrid
from .schedule import RandomActivationByBreed

##### States of each agent #####

class State(Enum):
    SUSCEPTIBLE = 0
    INFECTED = 1
    EXPOSED = 2
    RECOVERED = 3
    DEAD = 4

def become_state(model, state):
    return sum([1 for a in model.grid.get_all_cell_contents() if a.state is state])

def susceptible_counter(model):
    return become_state(model, State.SUSCEPTIBLE)

def infected_counter(model):
    return become_state(model, State.INFECTED)

def exposed_counter(model):
    return become_state(model, State.EXPOSED)

def recovered_counter(model):
    return become_state(model, State.RECOVERED)

def dead_counter(model):
    return become_state(model, State.DEAD)

##### Type of agents #####

def become_type(model, nodetype):
    return sum([1 for a in model.grid.get_all_cell_contents() if a.type is nodetype])

def sensor_counter(model):
    return become_type(model, 0) # Sensor

def router_counter(model):
    return become_type(model, 1) # Router
```

```
49
50  def sink_counter(model):
51      return become_type(model, 2) # Sink
52
53  ##### State per type of agents #####
54
55  def become_state_type(model, nodetype, state):
56      return sum([1 for a in model.grid.get_all_cell_contents() if a.type is nodetype and
57                  a.state is state])
58
59  ### SENSOR
60
61  def sensor_susceptible_counter(model):
62      return become_state_type(model, 0, State.SUSCEPTIBLE) # Sensors Susceptible
63
64  def sensor_infected_counter(model):
65      return become_state_type(model, 0, State.INFECTED) # Sensors Infected
66
67  def sensor_exposed_counter(model):
68      return become_state_type(model, 0, State.EXPOSED) # Sensors Exposed
69
70  def sensor_recovered_counter(model):
71      return become_state_type(model, 0, State.RECOVERED) # Sensors Recovered
72
73  def sensor_dead_counter(model):
74      return become_state_type(model, 0, State.DEAD) # Sensors Dead
75
76  ### ROUTER
77
78  def router_susceptible_counter(model):
79      return become_state_type(model, 1, State.SUSCEPTIBLE) # Router Susceptible
80
81  def router_infected_counter(model):
82      return become_state_type(model, 1, State.INFECTED) # Router Infected
83
84  def router_exposed_counter(model):
85      return become_state_type(model, 1, State.EXPOSED) # Router Exposed
86
87  def router_recovered_counter(model):
88      return become_state_type(model, 1, State.RECOVERED) # Router Recovered
89
90  def router_dead_counter(model):
91      return become_state_type(model, 1, State.DEAD) # Router Dead
92
93  ### SINK
94
95  def sink_susceptible_counter(model):
96      return become_state_type(model, 2, State.SUSCEPTIBLE) # Sink Susceptible
97
98  def sink_infected_counter(model):
99      return become_state_type(model, 2, State.INFECTED) # Sink Infected
100
101 def sink_exposed_counter(model):
102     return become_state_type(model, 2, State.EXPOSED) # Sink Exposed
103
```

```
104 def sink_recovered_counter(model):
105     return become_state_type(model, 2, State.RECOVERED) # Sink Recovered
106 
107 def sink_dead_counter(model):
108     return become_state_type(model, 2, State.DEAD) # Sink Dead
109 
110 ##### Definition of Model #####
111 
112 class wsnModel(Model):
113 
114     # Type of topology selected
115     topology_types = {"Hybrid": '1',
116                       "Star": '2',
117                       "Mesh": '3',
118                       "Scale free": '4',
119                       "Small world": '5'}
120 
121     # Type of phenomenon selected
122     phenomenon_types = {"Military/Industrial": 'Military/Industrial',
123                         "Health/Environmental": 'Health/Environmental',
124                         "Daily activities/Multimedia": 'Daily activities/Multimedia'}
125 
126     any_infected = 0
127 
128     def __init__(self, num_nodes, topology_types, phenomenon_types, any_infected):
129 
130         self.num_nodes = num_nodes
131         self.topology_types = topology_types
132         self.any_infected = any_infected
133 
134         #Topology
135         if self.topology_types == 'Mesh':
136             self.G = nx.erdos_renyi_graph(n=self.num_nodes, p=0.1)
137         elif self.topology_types == 'Star':
138             self.G = nx.star_graph(self.num_nodes)
139         elif self.topology_types == 'Scale free':
140             D = nx.scale_free_graph(n=self.num_nodes)
141             self.G = nx.Graph(D)
142         elif self.topology_types == 'Small world':
143             self.G = nx.connected_watts_strogatz_graph(self.num_nodes, 4, 0.5, tries=20,
144                     seed=None)
145         else:
146             self.G = nx.random_tree(self.num_nodes)
147         self.grid = NetworkGrid(self.G)
148         self.schedule = RandomActivationByBreed(self)
149 
150         # Phenomenon of interest to be monitored
151         self.phenomenon_types = phenomenon_types
152 
153         ##### Create agents #####
154 
155         # Malware
156         for i in range (1):
157             m = malware(i, self)
158             self.schedule.add(m)
```

```
159
160         # Topology
161         for i in range (1):
162             t = topology(i, self, self.topology_types)
163             self.schedule.add(t)
164
165         # Phenomenon
166         for i in range (1):
167             p = phenomenon(i, self, self.phenomenon_types)
168             self.schedule.add(p)
169
170         # Humans
171         for i in range (1):
172             h = humans(i, self, self.phenomenon_types)
173             self.schedule.add(h)
174
175         # Devices
176         for i in range (1):
177             d = devices(i, self)
178             self.schedule.add(d)
179
180         # Sensor nodes
181         for i, node in enumerate(self.G.nodes()):
182             s = sensors(i, self, State.SUSCEPTIBLE, self.topology_types)
183             self.schedule.add(s)
184             self.grid.place_agent(s, node)
185
186         ##### Infect some node #####
187         # FROM SUSCEPTIBLE TO INFECTED #
188         infected_nodes = random.sample(self.G.nodes(), self.num_nodes)
189         for a in self.grid.get_cell_list_contents(infected_nodes):
190             if self.any_infected == 0 and coefficients.malware_coefficient() == 1 and
191                 coefficients.infection_coefficient(1, a.comp, a.sec, a.data, a.cycle) == 1
192                 and coefficients.energy_coefficient(a.energy) == 1
193                 and coefficients.transmission_coefficient(a.comp, a.econs, a.antenna,
194                 a.sec, a.cycle):
195                 a.state = State.INFECTED
196                 self.any_infected = 1
197
198         self.datacollector = DataCollector({"Susceptible": susceptible_counter,
199                         "Infected": infected_counter,
200                         "Exposed": exposed_counter,
201                         "Recovered": recovered_counter,
202                         "Dead": dead_counter,
203                         "Sensors": lambda m: m.schedule.get_breed_count(sensors),
204                         "Malware": lambda m: m.schedule.get_breed_count(malware),
205                         "Topology": lambda m: m.schedule.get_breed_count(topology),
206                         "Phenomenon": lambda m: m.schedule.get_breed_count(phenomenon),
207                         "Humans": lambda m: m.schedule.get_breed_count(humans),
208                         "Devices": lambda m: m.schedule.get_breed_count(devices)})
209
210         self.running = True
211         self.datacollector.collect(self)
212
213     def step(self):
```

```
214         self.schedule.step()
215         self.datacollector.collect(self)
216 
217     def run_model(self, n):
218         for i in range(168):
219             self.step()
220 
221 ##### Actors and their characteristics #####
222 
223 # Sensor nodes
224 class sensors(Agent):
225     def __init__(self, unique_id, model, initial_state, topology_types):
226         super().__init__(unique_id, model)
227 
228         self.state = initial_state              # Status: 0 Susceptible, 1 Infected,
229                                                 # 2 Exposed, 3 Recovered, 4 Dead
230         self.energy = random.randint(15,100)    # Energy of nodes (0-100)
231         self.comp = 1 #random.randint(0,1)      # Computational capacity: 0 Low, 1 High
232         self.econs = random.randint(0,2)        # Energy consumption: 0 Very-Low/Low,
233                                                 # 1 Medium, 2 High/Very High
234         self.antenna = 1 #random.randint(0,1)   # Transmitter and receiver range: 0 Low,
235                                     1 High
236         self.sec = random.randint(1,2)          # Security: 0 Low, 1 Medium, 2 High
237         self.data = random.randint(0,2)         # Data collect method: 0 Periodical,
238                                                 # 1 External, 2 Request
239         self.cycle = random.randint(0,1)        # Duty cycle: 0 Inactive, 1 Active
240 
241         if topology_types == 'Star':
242             if self.unique_id is 0:
243                 self.type = 2 # Sink
244             else:
245                 self.type = 0 # Sensors
246         elif topology_types == 'Mesh':
247             if self.unique_id is 0:
248                 self.type = 2 # Sink
249             else:
250                 self.type = 1 # Router
251         else:
252             if self.unique_id is 0:
253                 self.type = 2 # Sink
254             else:
255                 neighbors = self.model.grid.get_neighbors(self.unique_id,
256                             include_center=False)
257                 if len(neighbors) == 1:
258                     self.type = 0
259                 else:
260                     self.type = 1   # Type of sensor node: 0 Sensors, 1 Router, 2 Sink
261 
262     ##### Rules of transitions #####
263 
264     def neighbors_to_infected_or_exposed(self):
265         neighbors_nodes = self.model.grid.get_neighbors(self.pos, include_center=False)
266         susceptible_neighbors = [agent for agent in self.model.grid.get_cell_list_contents
267                                  (neighbors_nodes) if agent.state is State.SUSCEPTIBLE]
268         for a in susceptible_neighbors:
```

```
269             if coefficients.malware_coefficient() == 1 and
270                 coefficients.infection_coefficient(1, a.comp, a.sec, a.data, a.cycle) == 1:
271                     a.state = State.INFECTED
272                     return 1
273                 elif coefficients.malware_coefficient() == 1 and
274                 coefficients.energy_coefficient(a.energy) == 1 and
275                 coefficients.transmission_coefficient(a.comp, a.econs, a.antenna,
276                 a.sec, a.cycle) == 0:
277                     a.state = State.EXPOSED
278                     return 1
279                 else:
280                     return 0
281
282     # FROM INFECTED TO RECOVERED #
283     def infected_to_recovered(self):
284         #print("RECOVERED")
285         if coefficients.detection_coefficient(self.comp, self.econs, self.sec) == 1 and
286         coefficients.recovery_coefficient(self.comp) == 1:
287             self.state = State.RECOVERED
288         else:
289             return 0
290
291     # FROM EXPOSED TO RECOVERED #
292     def exposed_to_recovered(self):
293         if coefficients.recovery_coefficient(self.comp) == 1:
294             neighbors_nodes = self.model.grid.get_neighbors(self.pos, include_center=False)
295             infected_neighbors = [agent for agent in self.model.grid.get_cell_list_contents
296                                   (neighbors_nodes) if agent.state is State.INFECTED]
297             if not infected_neighbors:
298                 self.state = State.RECOVERED
299                 self.energy = 100
300                 return 1
301             else:
302                 return 0
303         else:
304             return 0
305
306     # FROM RECOVERED TO SUSCEPTIBLE #
307     def recovered_to_susceptible(self):
308         self.energy = 100
309         self.state = State.SUSCEPTIBLE
310
311     def step(self):
312          # VERIFICA SI NO HAY INFECTADOS NI EXPUESTOS EN LA RED Y QUE TODOS LOS RECUPERADOS
313          # PASEN A SUSCEPTIBLE
314          if router_infected_counter(self.model) == 0 and
315          router_exposed_counter(self.model) == 0:
316              if self.state is State.RECOVERED:
317                  self.recovered_to_susceptible()
318          # Change state of duty cycle
319          else:
320              self.cycle = random.randint(0,1)
321              if self.cycle == 1:
322                  # ENERGY
323                  if 0 < self.energy <= 100:
```

```
324                     if self.econs == 2:
325                         self.energy = self.energy - 0.3
326                     elif self.econs == 1:
327                         self.energy = self.energy - 0.2
328                     else:
329                         self.energy = self.energy - 0.1
330                 elif self.energy <= 0:
331                     self.state = State.DEAD
332 
333                 # CHANGES OF STATES
334                 #if self.state is State.SUSCEPTIBLE:
335                 if self.state is State.INFECTED:
336                     self.energy = self.energy - 0.5
337                     if self.neighbors_to_infected_or_exposed() == 0:
338                         self.infected_to_recovered()
339                 elif self.state is State.EXPOSED:
340                     self.exposed_to_recovered()
341 
342 # Malware
343 class malware(Agent):
344     def __init__(self, unique_id, model):
345         super().__init__(unique_id, model)
346 
347         global mtype
348         global spread
349         global targ
350         self.mtype = 0              # Malware type
351         self.spread = 0             #random.randint(0,2)
352                                     # Spreading mechanisms: 0 Self-replication, 1 Exploit,
353                                     # 2 User interaction
354         self.targ = 1               #random.randint(0,3)
355                                     # Target: 0 Malicious code distribution, 1 Information
356                                     # exfiltration, 2 DoS, 3 Fraud
357 
358         if self.spread == 0:
359             self.spread_name = "Self-replication"
360         elif self.spread == 1:
361             self.spread_name = "Exploit"
362         else:
363             self.spread_name = "User interaction"
364 
365         if self.targ == 0:
366             self.targ_name = "Malicious code distribution"
367         elif self.targ == 1:
368             self.targ_name = "Information exfiltration"
369         elif self.targ == 2:
370             self.targ_name = "Denegation of services"
371         else:
372             self.targ_name = "Fraud"
373 
374         mtype = self.mtype
375         spread = self.spread
376         targ = self.targ
377 
378     def get_malware_parameters(option):
```

```
379         if option == 1:
380             return spread
381         elif option == 2:
382             return targ
383         else:
384             return mtype
385 
386     def step(self):
387         print("Malware")
388         print("Spreading mechanisms: ", self.spread_name)
389         print("Target: ", self.targ_name)
390 
391 # Topology
392 class topology(Agent):
393     def __init__(self, unique_id, model, topology_types):
394         super().__init__(unique_id, model)
395 
396         global topo
397         global proto
398 
399         self.topology_types = topology_types
400         if self.topology_types == 'Star':
401             self.topo = 0               # Star topology
402         elif self.topology_types == 'Mesh':
403             self.topo = 1               # Mesh topology
404         else:
405             self.topo = 2               # Hybrid topology
406 
407         self.proto = 0 #random.randint(0,1)     # Protocol: 0 SOP, 1 EECR
408 
409         if self.proto == 1:
410             self.proto_name = "EECR"
411         else:
412             self.proto_name = "SOP"
413 
414         topo = self.topo
415         proto = self.proto
416 
417     def get_topology_parameters(option):
418         if option == 1:
419             return proto
420         else:
421             return topo
422 
423     def step(self):
424         print("Topology: ", self.topology_types)    # Topology
425         print("Protocol: ", self.proto_name)        # Protocol
426 
427 # Phenomenon
428 class phenomenon(Agent):
429     def __init__(self, unique_id, model, phenomenon_types):
430         super().__init__(unique_id, model)
431 
432         global risk
433 
```

```
434         self.phenomenon_types = phenomenon_types
435         if self.phenomenon_types == 'Military/Industrial':
436             self.risk = 2 # High
437             self.risk_name = "High"
438         elif self.phenomenon_types == 'Health/Environmental':
439             self.risk = 1 # Medium
440             self.risk_name = "Medium"
441         else:
442             self.risk = 0 # Low
443             self.risk_name = "Low"
444 
445         risk = self.risk
446 
447     def get_phenomenon_parameters(option):
448         if option == 0:
449             return risk
450 
451     def step(self):
452         print("Risk of malware attacks: ", self.risk_name)  # Risk of malware attacks
453 
454 # Human action
455 class humans(Agent):
456     def __init__(self, unique_id, model, phenomenon_types):
457         super().__init__(unique_id, model)
458 
459         global level
460 
461         self.phenomenon_types = phenomenon_types
462         if self.phenomenon_types == 'Military/Industrial':
463             self.level = 0 # Low
464             self.level_name = "Low"
465         elif self.phenomenon_types == 'Health/Environmental':
466             self.level = 1 # Medium
467             self.level_name = "Medium"
468         else:
469             self.level = 2 # High
470             self.level_name = "High"
471 
472         level = self.level
473 
474     def get_humans_parameters(option):
475         if option == 0:
476             return level
477 
478     def step(self):
479         print("Human action level: ", self.level_name)  # Human action level on the network
480 
481 # External Devices
482 class devices(Agent):
483     def __init__(self, unique_id, model):
484         super().__init__(unique_id, model)
485 
486         global dev
487 
488         self.dev = 1 #random.randint(0,2)       # Risk of devices infected with malware:
```

```
489                                                 # 0 Low, 1 Medium, 2 High
490
491         if self.dev == 0:
492             self.dev_name = "Low"
493         elif self.dev == 1:
494             self.dev_name = "Medium"
495         else:
496             self.dev_name = "High"
497
498         dev = self.dev
499
500     def get_devices_parameters(option):
501         if option == 0:
502             return dev
503
504     def step(self):
505         print("Risk of devices infected with malware: ", self.dev_name)
506
507 ##### Coefficients #####
508
509 class coefficients():
510 # Infection coefficient
511     def infection_coefficient(first, capacity, security, data, cycle):
512         malware_type = malware.get_malware_parameters(0)
513         spread = malware.get_malware_parameters(1)
514         target = malware.get_malware_parameters(2)
515         risk = phenomenon.get_phenomenon_parameters(0)
516         human_action = humans.get_humans_parameters(0)
517         dev = devices.get_devices_parameters(0)
518         #prob = random.randint(0,100)
519         if first == 1:
520             if capacity == 1 and security == 0 and 0 <= data <= 2 and 0 <= cycle <= 1 and
521             malware_type == 0 and 0 <= spread <= 2 and 0 <= target <= 3 and 0 <= risk <= 2
522             and 0 <= human_action <= 2 and 1 <= dev <= 2:
523                 infection_coefficient = 1
524             elif 0 <= capacity <= 1 and 0 <= security <= 1 and 0 <= data <= 2 and 0 <=
525             cycle <= 1 and malware_type == 0 and 0 <= spread <= 2 and 0 <= target <= 3 and
526             0 <= risk <= 2 and 0 <= human_action <= 2 and 0 <= dev <= 1:
527                 infection_coefficient = 1
528             else:
529                 infection_coefficient = 0
530         else:
531             if 0 <= capacity <= 1 and 0 <= security <= 2 and 0 <= data <= 2 and 0 <=
532             cycle <= 1 and malware_type == 0 and 0 <= spread <= 2 and 0 <= target <= 3 and
533             0 <= risk <= 2 and 0 <= human_action <= 2 and 0 <= dev <= 2:
534                 infection_coefficient = 1
535             else:
536                 infection_coefficient = 0
537
538         return infection_coefficient
539
540 # Transmission coefficient
541     def transmission_coefficient(capacity, consumption, antenna, security, cycle):
542         malware_type = malware.get_malware_parameters(0)
543         spread = malware.get_malware_parameters(1)
```

```
544         target = malware.get_malware_parameters(2)
545         topo = topology.get_topology_parameters(0)
546         proto = topology.get_topology_parameters(1)
547         if capacity == 1 and consumption == 2 and antenna == 1 and 1 <= cycle <= 2 and
548         malware_type == 0 and 0 <= spread <= 2 and 0 <= target <= 3 and 0 <= topo <= 2 and
549         0 <= proto <= 1:
550             transmission_coefficient = 1
551         elif 0 <= capacity <= 1 and consumption == 1 and 0 <= antenna <= 1 and security ==
552         1 and 0 <= cycle <= 1 and malware_type == 0 and 0 <= spread <= 2 and 0 <= target
553         <= 3 and 0 <= topo <= 2 and 0 <= proto <= 1:
554             transmission_coefficient = 1
555         elif capacity == 0 and consumption == 0 and antenna == 0 and 0 <= cycle <= 1 and
556         malware_type == 0 and 0 <= spread <= 2 and 0 <= target <= 3 and 0 <= topo <= 2 and
557         0 <= proto <= 1:
558             transmission_coefficient = 1
559         else:
560             transmission_coefficient = 0
561 
562         return transmission_coefficient
563 
564 # Detection coefficient
565     def detection_coefficient(capacity, consumption, security):
566         human_action = humans.get_humans_parameters(0)
567         malware_type = malware.get_malware_parameters(0)
568         risk = phenomenon.get_phenomenon_parameters(0)
569         if 0 <= capacity <= 1 and 0 <= consumption <= 2 and 0 <= security <= 2 and
570         malware_type == 0 and 0 <= risk <= 2 and 0 <= human_action <= 2:
571             detection_coefficient = 1
572         return detection_coefficient
573 
574 # Recovery coefficient
575     def recovery_coefficient(capacity):
576         human_action = humans.get_humans_parameters(0)
577         proto = topology.get_topology_parameters(1)
578         if 0 <= capacity <= 1 and 0 <= human_action <= 2 and 0 <= proto <= 1:
579             recovery_coefficient = 1
580         return recovery_coefficient
581 
582 # Maintenance coefficient
583     def maintenance_coefficient(consumption):
584         human_action = humans.get_humans_parameters(0)
585         topo = topology.get_topology_parameters(0)
586         if 0 <= consumption <= 2 and 0 <= human_action <= 2 and 0 <= topo <= 2:
587             maintenance_coefficient = 1
588         else:
589             maintenance_coefficient = 0
590         return maintenance_coefficient
591 
592 # Energy coefficient
593     def energy_coefficient(energy_sensor):
594         if energy_sensor >= 15:
595             energy_coefficient = 1
596         elif energy_sensor < 15:
597             energy_coefficient = 0
598         return energy_coefficient
```

```
599
600 # Malware coefficient
601     def malware_coefficient():
602         malware_type = malware.get_malware_parameters(0)
603         if malware_type == 0:
604             malware_coefficient = 1
605         else:
606             malware_coefficient = 0
607         return malware_coefficient
```

Código B.4 Fichero model.py