Informe de Simulación de Red IoT con NS-3

4 de junio de 2025

1. Descripción de la Simulación

Este informe detalla un estudio de simulación de red realizado con el simulador NS-3 (versión 3.44) para evaluar el rendimiento de una red ad-hoc de Internet de las Cosas (IoT) bajo diversas condiciones, incluyendo la presencia de nodos maliciosos e interferentes. Las simulaciones son gestionadas por un script Bash (`run_simulations.sh`) e implementadas en un script C++ (`simulacioniot.cc`), con procesamiento adicional de métricas para el protocolo DSR manejado por un script Python (`manual_metrics_dsr.py`). Esta sección proporciona una descripción completa de la configuración de la simulación, la topología, los parámetros y los escenarios, de manera clara para que cualquier persona, incluso sin experiencia previa, pueda entender qué se está simulando y proponer nuevas ideas.

1.1. Objetivos de la Simulación

El objetivo principal es evaluar el rendimiento de diferentes protocolos de enrutamiento en una red IoT bajo condiciones normales, con interferencias y con comportamientos maliciosos. Se estudian cuatro protocolos de enrutamiento: AODV, OLSR, DSDV y DSR, analizando su comportamiento con nodos IoT fijos y móviles, así como con nodos maliciosos e interferentes. Las simulaciones generan métricas detalladas sobre el rendimiento de la red, incluyendo throughput, retardo, jitter, pérdida de paquetes y consumo de energía.

1.2. Topología de la Red

La red IoT simulada incluye cuatro tipos de nodos:

- **Nodos IoT Fijos**: Nodos estacionarios dispuestos en una cuadrícula, que representan dispositivos IoT estáticos como sensores o medidores inteligentes.
- **Nodos IoT Móviles**: Nodos que se mueven dentro de un área definida usando un modelo de movilidad de caminata aleatoria, simulando dispositivos como wearables o drones.
- **Nodos Maliciosos**: Nodos que generan tráfico excesivo para interrumpir la red, simulando ataques como denegación de servicio (DoS).
- **Nodos Interferentes**: Nodos estacionarios que generan tráfico adicional, simulando interferencias ambientales o condiciones de red congestionadas.

La red opera como una red inalámbrica ad-hoc utilizando el estándar IEEE 802.11g, con nodos comunicándose a través de un canal Wi-Fi compartido. Las direcciones IP se asignan desde la subred 192.168.1.0/24, asegurando una asignación única para cada nodo.

1.3. Configuraciones y Parámetros de la Simulación

La simulación se configura con los siguientes parámetros, definidos en `run_simulations.sh` y pasados a `simulacioniot.cc` mediante argumentos de línea de comandos:

- **Número de Nodos Fijos**: 20 nodos, dispuestos en una cuadrícula de 5x4 con una separación de 15 metros.
- **Número de Nodos Móviles**: 10 nodos, moviéndose dentro de un área de 100x100 metros usando un modelo de caminata aleatoria con velocidad constante de 1 m/s y cambios de dirección cada 2 segundos.
- **Duración de la Simulación**: 60 segundos por corrida.
- **Número de Corridas**: 10 corridas por configuración, cada una con una semilla aleatoria única (1001 a 1010) para garantizar reproducibilidad y robustez estadística.
- **Protocolos de Enrutamiento**: Se prueban cuatro protocolos:
  - AODV (Ad-hoc On-Demand Distance Vector)
  - OLSR (Optimized Link State Routing)
  - DSDV (Destination-Sequenced Distance-Vector)
  - DSR (Dynamic Source Routing)
- **Configuraciones de Nodos**: Se simulan cuatro escenarios, variando el número de nodos maliciosos e interferentes:
  - `no_mal_no_int`: 0 nodos maliciosos, 0 nodos interferentes (escenario base).
  - `int_no_mal`: 0 nodos maliciosos, 3 nodos interferentes.
  - `mal_no_int`: 2 nodos maliciosos, 0 nodos interferentes.
  - `mal_int`: 2 nodos maliciosos, 3 nodos interferentes.
- **Configuración Inalámbrica**:
  - **Estándar**: IEEE 802.11g.
  - **Canal**: Modelo YansWifiChannel por defecto.
  - **Configuración PHY**: RxSensitivity = -80 dBm, TxPower = 23 dBm, NistErrorRateModel.
  - **MAC**: Modo ad-hoc.
- **Parámetros de Tráfico**:
  - **Tráfico Normal**: Generado por nodos fijos y móviles usando aplicaciones UDP OnOff, enviando paquetes de 512 bytes a una tasa determinada por un intervalo de 2 segundos, dirigidos a un nodo sumidero en el puerto 9.

- **Tráfico Malicioso**: Generado por nodos maliciosos usando aplicaciones UDP OnOff, enviando paquetes de 512 bytes a alta tasa (intervalo de 0.01 segundos), dirigidos a un nodo sumidero en el puerto 10, comenzando a los 10 segundos.
- **Tráfico Interferente**: Generado por nodos interferentes usando aplicaciones UDP OnOff, enviando paquetes de 512 bytes a la misma tasa que el tráfico normal, comenzando a los 5 segundos.

- **Modelo de Energía**: Cada nodo está equipado con una fuente de energía básica (energía inicial: 100 Joules) y un modelo de energía WifiRadioEnergyModel para rastrear el consumo.
- **Modelos de Movilidad**:
  - Los nodos fijos usan un ConstantPositionMobilityModel en una disposición de cuadrícula.
  - Los nodos móviles usan un RandomWalk2dMobilityModel dentro de un área de 100x100 metros.
  - Los nodos maliciosos usan un RandomWalk2dMobilityModel dentro de un subárea de 60x90 metros.
  - Los nodos interferentes usan un ConstantPositionMobilityModel dentro de un subárea de 120x150 metros.

Cada corrida de simulación crea una estructura de directorios única bajo `simulacion_YYYYMMDD_HHMM`, organizada por configuración, protocolo y número de corrida (por ejemplo, `mal_int/AODV/run1`). El script `simulacioniot.cc` registra datos detallados, incluyendo capturas de paquetes, posiciones de nodos, consumo de energía, cambios en tablas de enrutamiento y métricas de rendimiento.

1.4. Escenarios de Simulación

Los cuatro escenarios están diseñados para probar la resiliencia de los protocolos de enrutamiento bajo diferentes condiciones de red:

- **Escenario Base (no_mal_no_int)**: Representa una red ideal con solo nodos IoT fijos y móviles, sirviendo como referencia para el rendimiento sin interrupciones.
- **Solo Interferencia (int_no_mal)**: Introduce 3 nodos interferentes para simular congestión de red, probando la capacidad de los protocolos para manejar tráfico adicional.
- **Solo Malicioso (mal_no_int)**: Incluye 2 nodos maliciosos generando tráfico de alta tasa, simulando un ataque DoS para evaluar la robustez de los protocolos.
- **Combinado (mal_int)**: Combina 2 nodos maliciosos y 3 nodos interferentes, representando un escenario de peor caso con ataques maliciosos e interferencias ambientales.

Cada escenario se ejecuta 10 veces para cada protocolo de enrutamiento, resultando en 160 corridas totales (4 escenarios Œ 4 protocolos Œ 10 corridas). Esto asegura significancia estadística y considera la variabilidad debido a la colocación aleatoria de nodos y su movilidad.

1.5. Proceso de Ejecución

El script `run_simulations.sh` automatiza la ejecución de las simulaciones:

1. Crea un directorio de simulación con marca de tiempo (`simulacion_YYYYMMDD_HHMM`).
2. Copia los scripts necesarios (`run_simulations.sh`, `simulacioniot.cc`, `manual_metrics_dsr.py` a un subdirectorio `scripts`.
3. Itera sobre cada configuración, protocolo y corrida, ejecutando `simulacioniot.cc` con NS-3 usando parámetros como conteo de nodos, tiempo de simulación, protocolo y semilla.
4. Organiza los archivos de salida en una jerarquía de directorios estructurada.
5. Ejecuta `manual_metrics_dsr.py` para generar métricas adicionales para el protocolo DSR si corresponde.

2. Análisis de los Resultados de la Simulación

Esta sección describe los resultados generados por las simulaciones, sus ubicaciones de almacenamiento, formatos y las métricas que contienen. Cada métrica se define para proporcionar claridad sobre su importancia y cómo refleja el rendimiento de la red.

2.1. Estructura de Directorios de Salida

Los resultados de la simulación se almacenan en un directorio con marca de tiempo (`simulacion_YYYYMMDD_HHMM`), con la siguiente estructura:

- `scripts/`: Contiene copias de `run_simulations.sh`, `simulacioniot.cc` y `manual_metrics_dsr.`
- `error_log.txt`: Registra errores, como archivos de salida faltantes (por ejemplo, `metrics.csv`, `nodes.csv` o directorios `pcap` vacíos).
- `[config_name]/[protocol]/run[1-10]/`: Directorios de salida por corrida para cada configuración (por ejemplo, `mal_int/AODV/run1`), que contienen:
  - `metadata.txt`: Metadatos de la simulación (por ejemplo, conteo de nodos, protocolo, marca de tiempo).
  - `node_metadata/nodes.csv`: Información específica de nodos (ID de nodo, dirección IP, tipo).
  - `metrics/`: Archivos de métricas de rendimiento:
    - `metrics.csv`: Métricas de rendimiento agregadas de la red.
    - `node_metrics.csv`: Métricas de rendimiento por nodo.
  - `packet_logs/`: Registros a nivel de paquetes:
    - `packets_normal.csv`: Registros de paquetes de tráfico normal.
    - `packets_malicious.csv`: Registros de paquetes de tráfico malicioso.
  - `routing_logs/control_messages.csv`: Registra mensajes de control de los protocolos de enrutamiento.

- `routing_logs/routing_table_changes.csv`: Registra cambios en las tablas de enrutamiento.
- `mobile_positions.csv`: Registra posiciones de nodos móviles a lo largo del tiempo.
- `energy_consumption.csv`: Registra el consumo de energía de los nodos a lo largo del tiempo.
- `pcap/`: Contiene archivos PCAP que capturan el tráfico de red, nombrados con prefijos como `iot_simulation_[protocol]_[nodes]_[config]`.

2.2. Métricas y sus Definiciones

Las simulaciones generan un conjunto completo de métricas para evaluar el rendimiento de la red, almacenadas principalmente en `metrics.csv` y `node_metrics.csv`. Para el protocolo DSR, métricas adicionales son calculadas por `manual_metrics_dsr.py` y almacenadas en `metrics.csv`. A continuación, se describe cada métrica, su definición y su importancia:

- **Marca de Tiempo** (`timestamp`): La fecha y hora en que se completó la corrida de simulación, con formato `YYYY-MM-DD HH:MM:SS`. Ayuda a rastrear cuándo se generaron los resultados.
- **Protocolo de Enrutamiento** (`protocolo`): El protocolo de enrutamiento usado (AODV, OLSR, DSDV o DSR), permite comparar el rendimiento entre protocolos.
- **Número de Nodos Fijos** (`nodos_fijos`): El conteo de nodos IoT fijos (20), refleja el componente estático de la red.
- **Número de Nodos Móviles** (`nodos_moviles`): El conteo de nodos IoT móviles (10), refleja el componente dinámico.
- **Número de Nodos Maliciosos** (`nodos_maliciosos`): El conteo de nodos maliciosos (0 o 2), indica la presencia de nodos de ataque.
- **Número de Nodos Interferentes** (`nodos_interferentes`): El conteo de nodos interferentes (0 o 3), indica fuentes de congestión.
- **Throughput Promedio** (`throughput_promedio`, kbps): La tasa de datos promedio recibida en todos los flujos, calculada como:

$$\text{Throughput} = \frac{\text{Bytes Recibidos Totales} \times 8}{\text{Tiempo de Simulación} \times 1000}$$

  Mide la eficiencia de transferencia de datos de la red. Valores más altos indican mejor rendimiento, mientras que nodos maliciosos o interferentes pueden reducirlo.
- **Throughput Máximo** (`throughput_maximo`, kbps): La suma del throughput de todos los flujos, usado como proxy del rendimiento máximo en `manual_metrics_dsr.py`. En `simulacioniot.cc`, refleja la suma de los throughputs de los flujos.
- **Retardo Promedio** (`delay_promedio`, segundos): El tiempo promedio que tarda un paquete en viajar de origen a destino, calculado como la media de las sumas de retardos por flujo dividida por el número de paquetes. Retardos más bajos indican

una entrega más rápida, mientras que nodos maliciosos pueden aumentarlos debido a interrupciones.

- **Retardo Máximo** (`delay_maximo`, segundos): El retardo máximo observado en todos los flujos, indica la latencia en el peor caso.
- **Retardo Mínimo** (`delay_minimo`, segundos): El retardo promedio mínimo por paquete en los flujos, refleja la latencia en el mejor caso.
- **Jitter Promedio** (`jitter_promedio`, segundos): La variación promedio en los tiempos de llegada de paquetes, calculada como la media de las diferencias absolutas entre retardos consecutivos. Un jitter más bajo indica una entrega más consistente, crucial para aplicaciones IoT en tiempo real.
- **Tasa de Pérdida de Paquetes** (`perdida_paquetes`, %): El porcentaje de paquetes perdidos, calculado como:

$$\text{Tasa de Pérdida} = \frac{\text{Paquetes Perdidos}}{\text{Paquetes Transmitidos Totales}} \times 100$$

  Tasas más altas indican una red menos confiable, a menudo incrementadas por nodos maliciosos o interferentes.
- **Tasa de Entrega de Paquetes (PDR)** (`pdr`, %): El porcentaje de paquetes entregados exitosamente, calculado como:

$$\text{PDR} = \frac{\text{Paquetes Recibidos}}{\text{Paquetes Transmitidos Totales}} \times 100$$

  Un PDR más alto indica mayor confiabilidad de la red.
- **Paquetes Totales** (`paquetes_totales`): El número total de paquetes transmitidos, proporciona contexto para las tasas de pérdida y entrega.
- **Paquetes Perdidos** (`paquetes_perdidos`): El número de paquetes perdidos, usado para calcular la tasa de pérdida.
- **Número de Flujos** (`numero_flujos`): El número de flujos de datos distintos monitoreados, típicamente un flujo agregado en `manual_metrics_dsr.py` o múltiples flujos en `simulacioniot.cc`.
- **Tiempo de Simulación** (`tiempo_simulacion`, segundos): La duración de la simulación (60 segundos), usada para normalizar métricas como el throughput.
- **Métricas Específicas por Nodo** (`node_metrics.csv`):
  - **ID de Nodo** (`node_id`): El identificador único de cada nodo.
  - **Throughput Promedio** (`throughput_avg`, kbps): Throughput promedio por nodo.
  - **Retardo Promedio** (`delay_avg`, segundos): Retardo promedio por nodo.
  - **Jitter Promedio** (`jitter_avg`, segundos): Jitter promedio por nodo.
  - **Energía Consumida** (`energy_consumed`, Joules): La energía restante de cada nodo, reflejando la eficiencia energética.

2.3. Archivos de Salida Adicionales

Además de las métricas de rendimiento, las simulaciones generan registros para un análisis detallado:

- **Registros de Paquetes** (`packets_normal.csv`, `packets_malicious.csv`):
  - **Columnas**: timestamp, source_ip, port, traffic_type, packet_size, sim_time.
  - **Propósito**: Registrar detalles de los paquetes recibidos, distinguiendo tráfico normal (puerto 9) y malicioso (puerto 10). Usado por `manual_metrics_dsr.py` para calcular métricas para DSR.
- **Registros de Enrutamiento** (`control_messages.csv`, `routing_table_changes.csv`):
  - **Columnas de Mensajes de Control**: timestamp, protocolo, nodo_id, tipo_mensaje, tamaño.
  - **Columnas de Cambios en Tablas de Enrutamiento**: timestamp, node_id, protocol, destination, next_hop, metric.
  - **Propósito**: Rastrear la sobrecarga de los protocolos de enrutamiento y las actualizaciones de tablas, útil para analizar la eficiencia y estabilidad del protocolo.
- **Posiciones de Nodos Móviles** (`mobile_positions.csv`):
  - **Columnas**: time, node_id, x, y, z.
  - **Propósito**: Registrar las posiciones de nodos móviles y maliciosos cada segundo, permite analizar el impacto de la movilidad.
- **Consumo de Energía** (`energy_consumption.csv`):
  - **Columnas**: time, node_id, energy_remaining.
  - **Propósito**: Rastrear el uso de energía a lo largo del tiempo, crucial para evaluar la eficiencia energética de dispositivos IoT.
- **Archivos PCAP** (`pcap/` directorio):
  - **Formato**: Archivos PCAP estándar que capturan todo el tráfico de red.
  - **Propósito**: Permiten un análisis detallado a nivel de paquetes usando herramientas como Wireshark, útil para depuración y verificación de patrones de tráfico.

2.4. Proceso de Generación

El script `simulacioniot.cc` utiliza `FlowMonitor` de NS-3 para recolectar métricas agregadas y funciones de registro personalizadas para guardar datos de paquetes, enrutamiento, movilidad y energía. Las funciones clave incluyen:

- `PacketLogger::LogNormalPacket` y `LogMaliciousPacket`: Registran detalles de paquetes en `packets_normal.csv` y `packets_malicious.csv`.
- `RoutingLogger::LogControlMessage`: Registra mensajes de control de los protocolos de enrutamiento.

- `LogRoutingTableChanges`: Registra actualizaciones de tablas de enrutamiento cada segundo.
- `LogMobilePositions`: Registra posiciones de nodos móviles cada segundo.
- `LogEnergyConsumption`: Registra niveles de energía cada segundo.
- `CalculateMetrics`: Calcula métricas agregadas al final de la simulación, escribiendo en `metrics.csv` y `node_metrics.csv`.

Para el protocolo DSR, `manual_metrics_dsr.py` procesa `packets_normal.csv` para calcular métricas, sobrescribiendo `metrics.csv` con valores derivados de los registros de paquetes. Este script asume un flujo agregado único y usa el tiempo de simulación como proxy para los retardos de paquetes, lo que puede diferir de las métricas basadas en `FlowMonitor`.

2.5. Perspectivas y Posibles Extensiones

Las métricas generadas permiten un análisis exhaustivo del rendimiento de los protocolos de enrutamiento bajo estrés. Por ejemplo:

- Comparar `throughput_promedio` y `pdr` entre escenarios puede revelar qué protocolos son más resilientes a ataques maliciosos o interferencias.
- `energy_consumption.csv` puede identificar protocolos eficientes en energía, crucial para dispositivos IoT alimentados por batería.
- `mobile_positions.csv` puede correlacionar la movilidad de nodos con la degradación del rendimiento.

Posibles extensiones incluyen:

- Agregar más protocolos de enrutamiento o tipos de ataques (por ejemplo, ataques de agujero negro).
- Variar la densidad de nodos o patrones de movilidad.
- Incorporar modelos de energía avanzados o condiciones de canal realistas.
- Analizar archivos PCAP para estudiar el comportamiento a nivel de paquetes en detalle.

3. Conclusión

Este estudio de simulación proporciona un marco robusto para evaluar el rendimiento de redes IoT usando NS-3. La topología detallada, las configuraciones y los archivos de salida permiten un análisis profundo del comportamiento de los protocolos de enrutamiento bajo diversas condiciones. Las métricas y registros generados ofrecen perspectivas sobre la eficiencia, confiabilidad y uso de energía de la red, sirviendo como base para futuras investigaciones y optimizaciones de redes IoT.