## ****Estado del arte: IA/ML en enrutamiento de redes mesh y IoT****

## ****¿Qué se está investigando?****

* **Protocolos de enrutamiento adaptativos**: Se desarrollan versiones de protocolos clásicos (AODV, OLSR, DSR, HWMP, Babel, BATMAN, RPL, etc.) que, en vez de usar reglas fijas, emplean modelos de ML para decidir rutas óptimas en tiempo real, adaptándose a cambios de topología, interferencias, congestión y ataques.
* **Predicción de fallos y congestión**: Algoritmos de ML (como redes neuronales, árboles de decisión, Q-learning, LSTM) predicen enlaces que van a fallar o congestionarse, permitiendo al protocolo evitar rutas problemáticas antes de que el problema ocurra.
* **Balanceo de carga inteligente**: Se usan modelos entrenados para distribuir el tráfico de manera óptima, evitando cuellos de botella y mejorando el throughput global.
* **Detección de anomalías y nodos maliciosos**: El ML ayuda a identificar patrones de tráfico anómalos o sospechosos, activando mecanismos de defensa o aislamiento de nodos.
* **Aprendizaje federado**: Cada nodo entrena localmente un modelo y comparte solo los parámetros, no los datos, protegiendo la privacidad y acelerando la convergencia en redes distribuidas.

## ****Ejemplos de investigaciones recientes****

* **Q-Routing y Deep Q-Networks (DQN)**: Adaptaciones de Q-learning para que cada nodo aprenda la mejor acción (siguiente salto) según el estado de la red.
* **Enrutamiento basado en LSTM**: Uso de redes neuronales recurrentes para predecir la calidad de los enlaces en función del historial de métricas.
* **Protocolos híbridos ML-clásicos**: El protocolo clásico toma la decisión por defecto, pero un módulo ML puede sobreescribir la decisión en situaciones críticas (alta congestión, movilidad extrema, ataque detectado).
* **Optimización multiobjetivo**: Algoritmos genéticos o de refuerzo que consideran simultáneamente latencia, consumo energético, fiabilidad y seguridad.

## ****Cómo investigar este tema usando NS-3****

## ****1. Integración de ML en NS-3****

* **Simulación directa**: Puedes implementar el modelo ML directamente en C++ o usar bindings de Python para llamar modelos entrenados externamente (por ejemplo, usando PyTorch o TensorFlow).
* **Interfaz NS-3 ↔ Python**: Usa PyBind o el módulo de Python de NS-3 para que el simulador consulte a un modelo ML (por ejemplo, cargado desde un archivo .pt o .h5) cada vez que un nodo deba tomar una decisión de enrutamiento.
* **Datos de entrenamiento**: Recolecta métricas de la red (delay, pérdida, congestión, calidad de enlace) durante simulaciones previas y entrena el modelo fuera de NS-3.
* **En línea**: Implementa aprendizaje en línea, donde el modelo mejora a medida que avanza la simulación.

## ****2. Ejemplo de flujo experimental****

1. **Define el escenario**: Por ejemplo, una smart city con movilidad, interferencias y nodos maliciosos.
2. **Simula con protocolos clásicos**: Obtén métricas base.
3. **Implementa tu protocolo adaptativo**: Modifica un protocolo existente (AODV, OLSR, etc.) para que consulte un modelo ML antes de tomar decisiones de enrutamiento.
4. **Compara resultados**: Analiza throughput, latencia, jitter, overhead, resiliencia ante fallos/ataques.
5. **Itera**: Ajusta el modelo ML y repite.

## ****3. Ejemplo de integración en NS-3 (pseudocódigo)****

cpp

*// En el método de decisión de enrutamiento de tu protocolo*

**if** (useML) {

*// Llama a un modelo Python que devuelve el siguiente salto óptimo*

**int** nextHop = CallPythonModel(currentState);

**if** (nextHop != -1) {

*// Usa la decisión del modelo ML*

route.SetNextHop(nextHop);

} **else** {

*// Fallback al algoritmo clásico*

route = ClassicRoutingDecision();

}

}

## ****4. Ideas de escenarios para comparar****

* Red mesh urbana con movilidad vehicular y peatonal, interferencia variable y eventos de ataque.
* Red de sensores industriales con patrones de tráfico impredecibles y fallas de enlaces.
* Red de emergencia (resiliencia): nodos deben reorganizarse tras un desastre.

## ****Enfoques innovadores y poco explorados****

* **Aprendizaje transferido**: Entrena el modelo ML en un tipo de topología y prueba su eficacia en otra diferente.
* **Cooperación entre nodos**: Modelos ML que comparten información entre nodos para mejorar la toma de decisiones colectiva.
* **Auto-configuración dinámica**: El protocolo ajusta sus propios parámetros (intervalos de hello, timers, etc.) en tiempo real usando ML.
* **Comparativa ML vs heurísticas clásicas**: Analiza en qué condiciones el ML supera (o no) a los algoritmos tradicionales.

## ****Referencias y recursos para profundizar****

* [Survey**Conclusión**](https://arxiv.org/abs/2107.11314)

La integración de ML e IA en el enrutamiento de redes mesh e IoT es una tendencia en auge, con investigaciones enfocadas en adaptabilidad, resiliencia y optimización multiobjetivo. NS-3 es una plataforma adecuada para experimentar, permitiendo comparar protocolos clásicos y versiones inteligentes bajo escenarios realistas y controlados.

* [: Machine Learning for Routing in Wireless Mesh Networks](https://arxiv.org/abs/2107.11314)
* [Q-Routing: A Reinforcement Learning Approach to Dynamic Routing](https://www.cs.cornell.edu/home/kleinber/networks-book/networks-book-ch07.pdf)
* Papers recientes en IEEE Xplore y arXiv sobre "ML-based routing in IoT/mesh networks"
* Ejemplos de integración NS-3 y Python: [NS-3 Python Bindings](https://www.nsnam.org/docs/release/3.44/tutorial/html/python.html)

## ****Conclusión****

La integración de ML e IA en el enrutamiento de redes mesh e IoT es una tendencia en auge, con investigaciones enfocadas en adaptabilidad, resiliencia y optimización multiobjetivo. NS-3 es una plataforma adecuada para experimentar, permitiendo comparar protocolos clásicos y versiones inteligentes bajo escenarios realistas y controlados.