**Proyecto: Detección de Anomalías en Tráfico de Red con Integrales**

**Parte I: Fundamento Matemático**

**Integrales Discretas en Detección de Anomalías**

La detección de anomalías en tráfico de red utiliza el concepto de **suma acumulativa (CUSUM)**, que es una aplicación práctica de la integral definida en un contexto discreto. Esta técnica permite identificar desviaciones significativas en el comportamiento de una serie temporal.

**Definición matemática:**

La suma acumulativa se define mediante las ecuaciones:

S₀ = 0

Sₜ = max(0, Sₜ₋₁ + (xₜ - μ))

Donde:

* Sₜ es la suma acumulada hasta el tiempo t
* xₜ es el valor del tráfico en el instante t
* μ es la media del tráfico normal (línea base)
* max(0,·) impide valores negativos, lo que evita que la suma se reduzca demasiado tras periodos de bajo tráfico

Esta formulación equivale a calcular una integral discreta que mide la desviación acumulada respecto a la media normal, permitiendo que pequeñas desviaciones sostenidas o grandes desviaciones puntuales sean detectadas eficazmente.

**Relación con Ataques de Red**

Los ataques de denegación de servicio (DoS) y denegación de servicio distribuida (DDoS) se caracterizan por generar volúmenes anormalmente altos de tráfico que sobrepasan la capacidad de los sistemas objetivo. Estos ataques pueden manifestarse como:

1. **Picos abruptos**: Aumento súbito y masivo del tráfico (flood)
2. **Incrementos sostenidos**: Aumento gradual pero persistente del tráfico

La suma acumulativa es especialmente eficaz porque:

* Detecta ambos tipos de anomalías, ya que acumula desviaciones a lo largo del tiempo
* No depende únicamente de valores instantáneos, reduciendo falsos positivos
* La función max(0,·) permite que el algoritmo "recupere la memoria" tras periodos normales

**Parte II: Implementación Técnica**

**Descripción del Sistema**

El sistema desarrollado implementa un detector de anomalías en tiempo real basado en el algoritmo CUSUM, con las siguientes características:

1. **Simulación de tráfico**: Genera datos simulados de tráfico de red durante 60 segundos (60 muestras/segundo)
2. **Inserción de ataque**: Introduce un pico de tráfico (ataque simulado) en un momento aleatorio
3. **Detección**: Aplica el algoritmo CUSUM para detectar la anomalía
4. **Visualización**: Muestra en tiempo real:

* El tráfico original vs tiempo
* La suma acumulada vs tiempo
* Los umbrales de detección (fijo y estadístico)

1. **Alerta**: Notifica cuando se detecta una anomalía, indicando el momento exacto

**Arquitectura del Sistema**

El sistema se compone de tres clases principales:

1. **DetectorAnomalias**: Núcleo algorítmico que implementa CUSUM
2. **PantallaCarga**: Proporciona una animación inicial para mejorar la experiencia de usuario
3. **AplicacionTiempoReal**: Maneja la interfaz gráfica y la visualización en tiempo real

**Método de Detección**

Se implementan dos tipos de umbrales para la detección:

1. **Umbral fijo**: Valor constante predefinido (2000 en esta implementación)
2. **Umbral estadístico**: Media + 3 desviaciones estándar del comportamiento reciente

umbral\_estadistico[i] = media + 3 \* desv\_est

**Análisis del Código**

**Librerías principales:**

* **NumPy**: Para cálculos vectoriales eficientes (generación de datos y procesamiento de señales)
* **Matplotlib**: Para visualización y animación de gráficos en tiempo real
* **Tkinter**: Para crear la interfaz gráfica de usuario

**Funciones clave:**

def generar\_datos(self):

# Tráfico normal

self.trafico = np.random.normal(

self.media\_trafico\_normal,

self.desviacion\_trafico\_normal,

self.total\_muestras

)

# Insertar ataque

self.trafico[self.inicio\_ataque:self.fin\_ataque] += (self.media\_trafico\_normal

\* self.factor\_ataque

\* np.ones(self.duracion\_ataque))

Esta función genera datos de tráfico normal siguiendo una distribución gaussiana e inserta un ataque simulado multiplicando la media normal por un factor de ataque.

def procesar\_paso(self):

# ...

# Calcular suma acumulada para este punto

if self.indice\_actual > 0:

# Línea base

ventana\_base = min(5 \* self.muestras\_por\_segundo, self.indice\_actual)

linea\_base = np.mean(self.trafico[:ventana\_base])

# Actualizar suma acumulada

self.suma\_acumulada[self.indice\_actual] = max(

0,

self.suma\_acumulada[self.indice\_actual-1] +

(self.trafico[self.indice\_actual] - linea\_base)

)

# Calcular umbral estadístico

if self.indice\_actual >= 5 \* self.muestras\_por\_segundo:

ventana = 5 \* self.muestras\_por\_segundo

valores\_ventana = self.suma\_acumulada[max(0, self.indice\_actual-ventana):self.indice\_actual]

media = np.mean(valores\_ventana)

desv\_est = np.std(valores\_ventana)

self.umbral\_estadistico[self.indice\_actual] = media + 2.5 \* desv\_est

Esta función implementa el cálculo en tiempo real de:

1. La suma acumulada mediante la fórmula CUSUM
2. El umbral estadístico basado en una ventana deslizante de 5 segundos

**Parte III: Resultados y Análisis**

**Comportamiento del Sistema**

Durante la ejecución, el sistema muestra:

1. **Fase inicial**: Tráfico normal con pequeñas fluctuaciones; la suma acumulada se mantiene en valores bajos.
2. **Ataque**: Al ocurrir el pico de tráfico, la suma acumulada comienza a crecer rápidamente.
3. **Detección**: Cuando la suma supera alguno de los umbrales, se genera una alerta que indica:

* El momento exacto de la detección
* El momento real del ataque
* El tiempo de detección (diferencia entre ambos)

1. **Post-ataque**: Tras el ataque, el tráfico vuelve a la normalidad, pero la suma acumulada permanece elevada hasta que se reinicia el sistema.

**Ventajas del Método CUSUM**

* **Robustez**: Resistente a fluctuaciones normales del tráfico
* **Sensibilidad ajustable**: Mediante la configuración de umbrales
* **Detección rápida**: El tiempo de detección típico es inferior a 1 segundo
* **Baja tasa de falsos positivos**: Gracias al uso de umbrales estadísticos adaptativos

**Conclusiones**

El método de suma acumulada (CUSUM) basado en integrales discretas proporciona una herramienta matemática potente para la detección de anomalías en tráfico de red. La implementación desarrollada demuestra:

1. La eficacia del enfoque matemático para detectar ataques DDoS simulados
2. La viabilidad de implementación en tiempo real con baja latencia de detección
3. La flexibilidad del sistema para adaptarse a diferentes perfiles de tráfico

Esta técnica podría extenderse para:

* Detectar otros tipos de ataques (escaneos, exfiltración de datos)
* Implementarse en sistemas de seguridad perimetral
* Integrarse con sistemas de mitigación automática

**Apéndice: Instrucciones de Ejecución**

Para ejecutar el sistema:

1. Asegúrese de tener instaladas las bibliotecas necesarias:

   pip install numpy matplotlib

1. Ejecute el archivo principal:

   python detector\_anomalias\_v2.py

1. La aplicación mostrará:

* Una pantalla de carga inicial
* La interfaz principal de monitoreo
* Gráficos en tiempo real del tráfico y suma acumulada
* Alertas cuando se detecten anomalías

1. Utilice el botón "ⓘ Explicación" para obtener detalles sobre el fundamento matemático del algoritmo.