

PROYECTO FINAL DEL CURSO DE REDES Y  
TRANSMISIÓN DE DATOS

# GENERADOR DE PAQUETES CAPAZ DE EVALUAR LOS LÍMITES Y SEGURIDAD DE UNA RED UTILIZANDO GNS3 VM Y SCAPY

GRUPO 3:

- Quispe Cabello, Jose Alessandro
- Sanchez Wong, Jatziry Fernanda



# INTRO

El proyecto tiene como objetivo principal diseñar e implementar un generador de paquetes capaz de evaluar la seguridad y el rendimiento de una red. Este sistema permitirá simular distintos escenarios, desde tráfico legítimo de alta carga hasta tráfico malicioso, para identificar vulnerabilidades y analizar el comportamiento de dispositivos de red en condiciones diversas.



# ANTECEDENTES



La seguridad en redes es esencial ante amenazas crecientes y la complejidad actual. Este trabajo se basa en antecedentes como el uso de herramientas avanzadas, pruebas de estrés y simuladores como GNS3 para analizar vulnerabilidades. Rincón (2021) destaca el hacking ético y estándares como NIST para evaluar redes IPv4, mientras que Hussain et al. (2021) presentan IoT-Flock, una herramienta para simular tráfico y detectar ataques, enfoques relevantes para fortalecer la seguridad en redes.



El proyecto se desarrolló en fases que abarcan desde la planificación hasta la documentación de resultados, garantizando un diseño e implementación sistemáticos del generador de paquetes y el análisis del entorno de red. Se incluye el esquema general, los elementos requeridos, las actividades principales y los scripts del sistema.

- Hardware:
  - Una PC o laptop con Oracle VM VirtualBox configurado para alojar la máquina virtual de GNS3.
  - La máquina virtual de GNS3 actuará como la red simulada.
- Software:
  - GNS3 VM en Oracle VM VirtualBox.
  - Wireshark instalado en tu PC anfitriona (Windows) para la captura y análisis de tráfico.
  - Uso de Python en la terminal de Windows (o un entorno de desarrollo como VSCode) para ejecutar los scripts de Scapy.

# METODOLOGÍA PROPUUESTA



# PROTOCOLO TCP E ICMP: DETALLES Y CONFIGURACIÓN



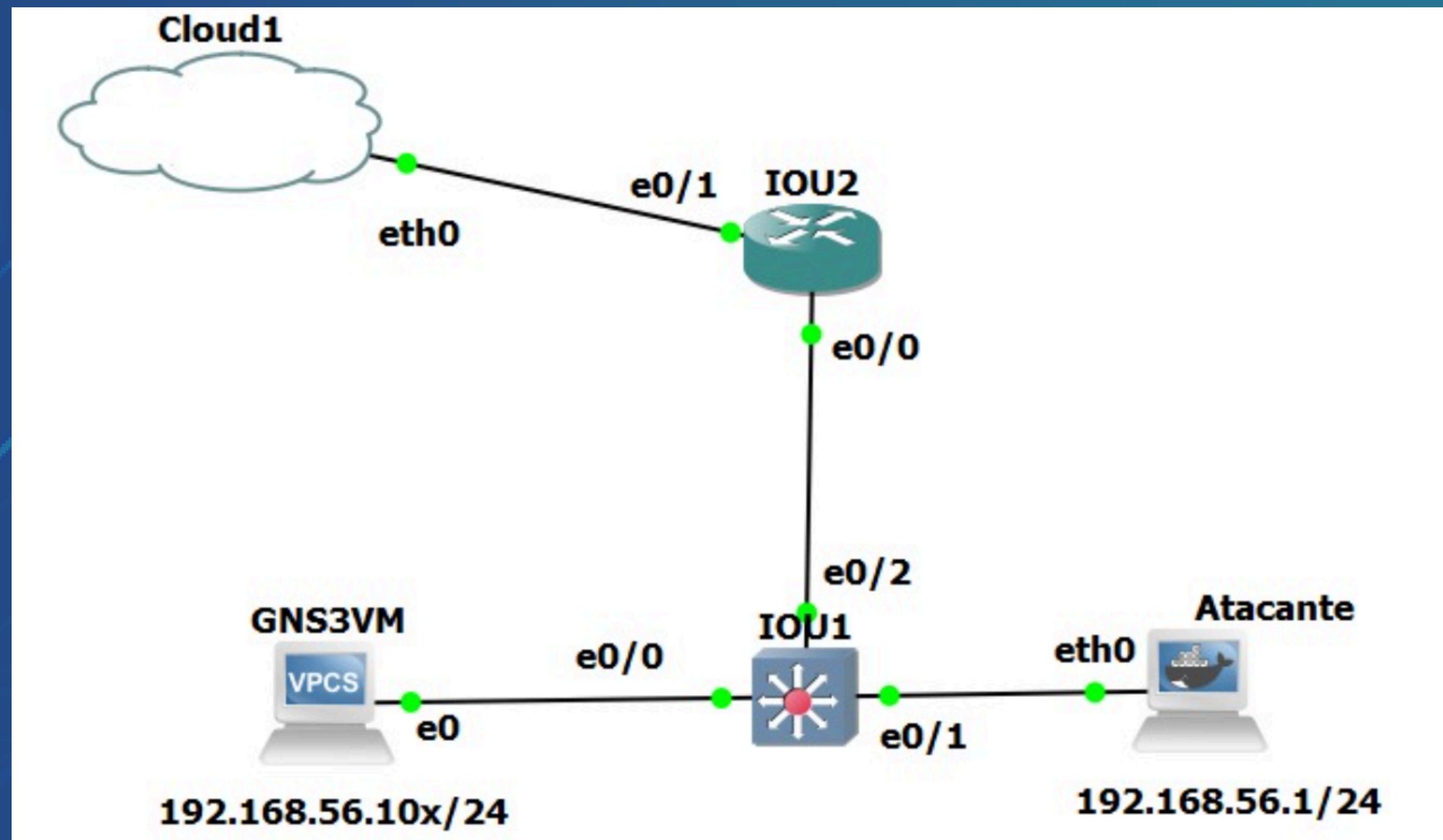
## Protocolo ICMP

- Ping básico
  - Tipo: Echo Request (Tipo 8, Código 0)
  - Tamaño: 64-1500 bytes
  - Intervalo: 1 paquete/0.1 s (configurable)
  - Objetivo: Verificar conectividad básica.
- Ping Flood
  - Tipo: Echo Request (512-1024 bytes)
  - Tasa: Hasta 1000 paquetes/s
  - Objetivo: Evaluar tolerancia a alta carga ICMP.

## Protocolo TCP

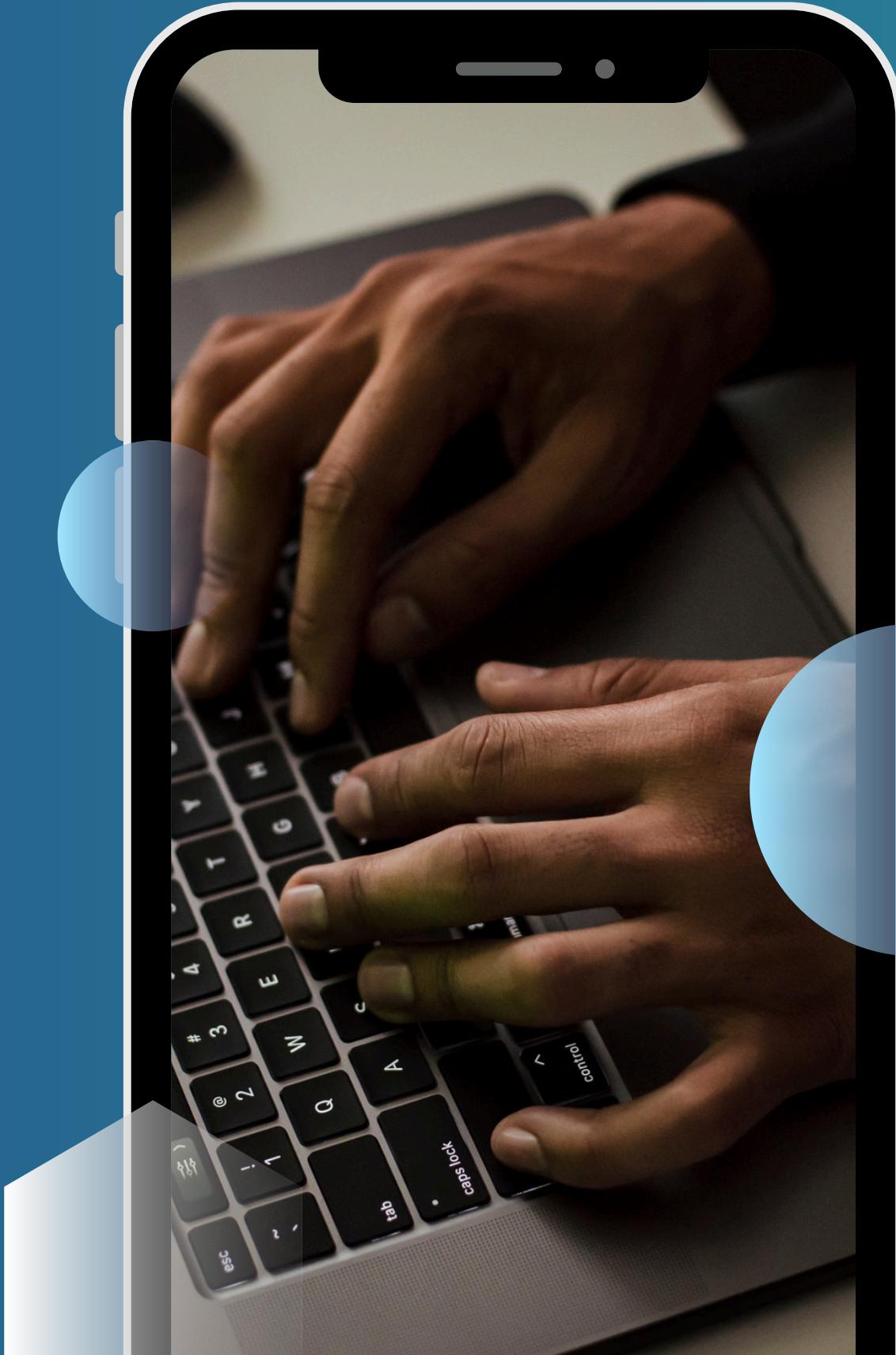
- Tráfico legítimo
  - Tipo: Conexiones estándar (SYN, SYN-ACK, ACK)
  - Puertos: 80 (HTTP), 443 (HTTPS), 22 (SSH)
  - Tasa: 10-100 conexiones/s
  - Objetivo: Medir respuesta de firewalls e IDS/IPS.
- Ataque SYN Flood
  - Tipo: TCP SYN sin completar handshake
  - IP de origen: Aleatoria (simula ataques distribuidos)
  - Tasa: Alta, con intervalos breves
  - Objetivo: Probar mitigación de ataques DoS.

# ESQUEMA DE IMPLEMENTACIÓN



# CONFIGURACIÓN DE GRUPOS DE PAQUETES

- **Tráfico ICMP Básico:** Destino: IP del router principal, Tamaño: 64 bytes, Tasa: 1 paquete/segundo.
- **Tráfico ICMP Malicioso:** Destino: Servidor simulado, Tamaño: 1024 bytes, Tasa: 1000 paquetes/segundo.
- **Tráfico TCP Legítimo:** Destino: Servidor simulado, Puertos: 80 y 443, Frecuencia: 10 conexiones/segundo.
- **Ataque SYN Flood:** Destino: Firewall, Puertos: Aleatorios, Frecuencia: 1000 paquetes/segundo.



- Generación de Tráfico: Scripts en Python utilizando Scapy para personalizar los paquetes.
- Análisis de Tráfico: Wireshark para capturar y analizar la comunicación en tiempo real dentro de la red simulada.
- Sistema Operativo: Las máquinas virtuales utilizan sistemas Linux por su flexibilidad y compatibilidad con herramientas como Scapy.



## HERRAMIENTAS Y CONFIGURACIÓN ADICIONAL



# FASES Y ACTIVIDADES DEL PROYECTO

09/15

Fase	Actividades
Planificación	<ul style="list-style-type: none"><li>• Identificación de las limitaciones del entorno y ajustar las herramientas necesarias.</li><li>• }Configuración de la conexión entre la máquina anfitriona (Windows) y las máquinas virtuales (GNS3 VM o Linux VM).</li></ul>
Diseño del sistema	<ul style="list-style-type: none"><li>• Creación de scripts personalizados en Python utilizando Scapy</li><li>• .Diseño de una red dentro de GNS3 o VirtualBox, asegurando que las máquinas puedan comunicarse.</li></ul>
Implementación	<ul style="list-style-type: none"><li>• Configuración de Red: Asignación de IP estáticas y configuración de servicios en máquinas objetivo (e.g., servidor web en Linux).</li><li>• Generación de Tráfico: Ejecución de scripts Scapy para probar ICMP, Ping Flood, TCP legítimo y SYN Flood.</li><li>• Captura de Tráfico: Uso de Wireshark o tcpdump para capturar paquetes en tiempo real.</li></ul>
Documentación	<ul style="list-style-type: none"><li>• Registro de los resultados obtenidos</li><li>• ).Proposición de recomendaciones basadas en los datos obtenidos.</li></ul>



# RESULTADOS

## PRUEBA 1: GENERADOR ICMP CON MEDICIÓN DE RENDIMIENTO PRUEBA 2: GENERADOR ICMP CON MEDICIÓN DE RENDIMIENTO

```

import time
from scapy.all import *
import pandas as pd

def generate_and_measure_icmp(target_ip, count=10, interval=1):
    latencies = []
    lost_packets = 0

    print(f"Enviando {count} paquetes ICMP a {target_ip}...")

    for i in range(count):
        send_time = time.time()
        packet = IP(dst=target_ip) / ICMP()
        reply = sr1(packet, timeout=2, verbose=False)
        receive_time = time.time()

        if reply:
            latency = (receive_time - send_time) * 1000 # Convertir a ms
            latencies.append(latency)
            print(f"Paquete {i+1}: Latencia = {latency:.2f} ms")
        else:
            lost_packets += 1
            print(f"Paquete {i+1}: Perdido")

        time.sleep(interval)

    # Calcular métricas
    avg_latency = sum(latencies) / len(latencies) if latencies else 0
    jitter = max(latencies) - min(latencies) if len(latencies) > 1 else 0
    loss_rate = (lost_packets / count) * 100

    print("\nResultados:")
    print(f"Latencia promedio: {avg_latency:.2f} ms")
    print(f"Jitter: {jitter:.2f} ms")
    print(f"Tasa de pérdida de paquetes: {loss_rate:.2f}%")

    # Guardar datos en CSV
    data = {"Packet": list(range(1, count+1)), "Latency (ms)": latencies + [None]*lost_packets}
    df = pd.DataFrame(data)
    df.to_csv("icmp_results.csv", index=False)
    print("Resultados guardados en 'icmp_results.csv'")

# Uso
target_ip = "192.168.56.102"
generate_and_measure_icmp(target_ip, count=20, interval=0.5)

```



# RESULTADOS

## PRUEBA 1: GENERADOR ICMP CON MEDICIÓN DE RENDIMIENTO PRUEBA 2: GENERADOR ICMP CON MEDICIÓN DE RENDIMIENTO

C:\Users\hotdo\Downloads>py ataque.py  
Enviando 20 paquetes ICMP a 192.168.56.102...  
Paquete 1: Latencia = 109.75 ms  
Paquete 2: Latencia = 8.95 ms  
Paquete 3: Latencia = 3.86 ms  
Paquete 4: Latencia = 7.49 ms  
Paquete 5: Latencia = 105.43 ms  
Paquete 6: Latencia = 10.37 ms  
Paquete 7: Latencia = 107.05 ms  
Paquete 8: Latencia = 114.41 ms  
Paquete 9: Latencia = 114.05 ms  
Paquete 10: Latencia = 6.98 ms  
Paquete 11: Latencia = 109.45 ms  
Paquete 12: Latencia = 8.85 ms  
Paquete 13: Latencia = 113.18 ms  
Paquete 14: Latencia = 10.17 ms  
Paquete 15: Latencia = 7.71 ms  
Paquete 16: Latencia = 13.47 ms  
Paquete 17: Latencia = 12.05 ms  
Paquete 18: Latencia = 106.32 ms  
Paquete 19: Latencia = 120.62 ms  
Paquete 20: Latencia = 21.80 ms  
  
Resultados:  
Latencia promedio: 55.60 ms  
Jitter: 116.76 ms  
Tasa de pérdida de paquetes: 0.00%  
Resultados guardados en 'icmp\_results.csv'  
  
C:\Users\hotdo\Downloads>

192.168.56.102 (gns3)  
Terminal Sessions View X server Tools Games Settings Macros Help  
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help  
X server Exit  
Quick connect...  
/home/gns3/  
Name  
.. .cache .config .local .ssh .venv GNS3 .bash\_history .bash\_logout .bash\_profile .bashrc .profile .python\_history .sudo\_as\_admin\_successful .wget-hsts .Xauthority  
07:41:07.893737 IP 192.168.56.1 > 192.168.56.102: ICMP echo request, id 0, seq 0, length 8  
07:41:07.893777 IP 192.168.56.102 > 192.168.56.1: ICMP echo reply, id 0, seq 0, length 8  
07:41:08.507305 IP 192.168.56.1 > 192.168.56.102: ICMP echo request, id 0, seq 0, length 8  
07:41:08.507355 IP 192.168.56.102 > 192.168.56.1: ICMP echo reply, id 0, seq 0, length 8  
07:41:09.028477 IP 192.168.56.1 > 192.168.56.102: ICMP echo request, id 0, seq 0, length 8  
07:41:09.028539 IP 192.168.56.102 > 192.168.56.1: ICMP echo reply, id 0, seq 0, length 8  
07:41:09.642662 IP 192.168.56.1 > 192.168.56.102: ICMP echo request, id 0, seq 0, length 8  
07:41:09.642705 IP 192.168.56.102 > 192.168.56.1: ICMP echo reply, id 0, seq 0, length 8  
07:41:10.151826 IP 192.168.56.1 > 192.168.56.102: ICMP echo request, id 0, seq 0, length 8  
07:41:10.151868 IP 192.168.56.102 > 192.168.56.1: ICMP echo reply, id 0, seq 0, length 8  
07:41:10.665839 IP 192.168.56.1 > 192.168.56.102: ICMP echo request, id 0, seq 0, length 8  
07:41:10.665873 IP 192.168.56.102 > 192.168.56.1: ICMP echo reply, id 0, seq 0, length 8  
  
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

# RESULTADOS

# PRUEBA 1: GENERADOR ICMP CON MEDICIÓN DE RENDIMIENTO1: GENERADOR ICMP CON MEDICIÓN DE RENDIMIENTOTURE

## Resultados

Latencia promedio: 58.61 m

Jitter: 129.83 ms

Tasa de pérdida de paquetes: 0.00%

Resultados guardados en 'icmp\_results.csv'

Capturando desde Ethernet 2

Archivo Edición Visualización Ir Captura Analizar Estadísticas Teléfono Wireless Herramientas Ayuda

Aplique un filtro de visualización ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0a:00:27:00:00:0b	Broadcast	ARP	42	Who has 192.168.56.102? Tell 192.168.56.1
2	0.000700	PCSSystemtec_da:72:...	0a:00:27:00:00:0b	ARP	60	192.168.56.102 is at 08:00:27:da:72:4a
3	0.001978	192.168.56.1	192.168.56.102	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 4)
4	0.002383	192.168.56.102	192.168.56.1	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 3)
5	0.507358	192.168.56.1	192.168.56.102	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 6)
6	0.507761	192.168.56.102	192.168.56.1	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 5)
7	1.016534	192.168.56.1	192.168.56.102	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 8)
8	1.016886	192.168.56.102	192.168.56.1	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 7)
9	1.623247	192.168.56.1	192.168.56.102	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 10)
10	1.623716	192.168.56.102	192.168.56.1	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 9)
11	2.228872	192.168.56.1	192.168.56.102	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 12)
12	2.229587	192.168.56.102	192.168.56.1	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 11)
13	2.740002	192.168.56.1	192.168.56.102	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 14)
14	2.740234	192.168.56.102	192.168.56.1	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 13)
15	3.243180	192.168.56.1	192.168.56.102	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 16)
16	3.243561	192.168.56.102	192.168.56.1	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 15)
17	3.748091	192.168.56.1	192.168.56.102	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 18)
18	3.748588	192.168.56.102	192.168.56.1	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 17)
19	4.256394	192.168.56.1	192.168.56.102	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 20)
20	4.257665	192.168.56.102	192.168.56.1	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 19)
21	4.765804	192.168.56.1	192.168.56.102	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 22)
22	4.766712	192.168.56.102	192.168.56.1	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 21)
23	5.173226	PCSSystemtec_da:72:...	0a:00:27:00:00:0b	ARP	60	Who has 192.168.56.1? Tell 192.168.56.102

Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF\_{87D4D3C8-37 0000 ff ff ff ff ff 0a 00 27 00 00 0b 08 06 00 01 ..... 8f  
Ethernet II, Src: 0a:00:27:00:00:0b (0a:00:27:00:00:0b), Dst: Broadcast (ff:ff:ff:ff:ff:ff)  
0010 08 00 06 04 00 01 0a 00 27 00 00 0b c0 a8 38 01 ..... 8f  
0020 00 00 00 00 00 00 c0 a8 38 66 ..... 8f  
Address Resolution Protocol (request)

# RESULTADOS

## PRUEBA 2: GENERADOR TCP SYN

```
import time
from scapy.all import *
import pandas as pd

def generate_and_measure_syn(target_ip, target_port, count=10, interval=0.5):
    success_count = 0
    failed_count = 0
    start_time = time.time()

    print(f"Enviando {count} paquetes SYN a {target_ip}:{target_port}...")

    for i in range(count):
        packet = IP(dst=target_ip) / TCP(dport=target_port, flags="S")
        response = sr1(packet, timeout=2, verbose=False)

        if response and response.haslayer(TCP) and response[TCP].flags == "SA":
            success_count += 1
            print(f"Paquete {i+1}: Respuesta recibida (ACK)")
        else:
            failed_count += 1
            print(f"Paquete {i+1}: Sin respuesta")

        time.sleep(interval)

    end_time = time.time()
    total_time = end_time - start_time
    throughput = success_count / total_time if total_time > 0 else 0

    print("\nResultados:")
    print(f"Paquetes enviados: {count}")
    print(f"Respuestas recibidas: {success_count}")
    print(f"Respuestas fallidas: {failed_count}")
    print(f"Throughput: {throughput:.2f} paquetes/segundo")

    # Guardar datos en CSV
    data = {"Packet": list(range(1, count+1)), "Status": ["ACK" if i < success_count else "No Response" for i in range(count)]}
    df = pd.DataFrame(data)
    df.to_csv("tcp_syn_results.csv", index=False)
    print("Resultados guardados en 'tcp_syn_results.csv'")

# Uso
target_ip = "192.168.56.102"
target_port = 80
generate_and_measure_syn(target_ip, target_port, count=15, interval=0.3)
```



# RESULTADOS

## PRUEBA 2: GENERADOR TCP SYN

The screenshot shows a Windows desktop environment with several open windows:

- A terminal window titled "C:\Windows\System32\cmd.e" displays the output of a Python script named "ataque2.py". The output indicates that 15 SYN packets were sent to the target IP 192.168.56.102. It shows responses for 10 packets (ACK) and 5 packets (No Response). The results are saved to "icmp\_results.csv".
- An "X server" window titled "192.168.56.102 (gns3)" shows a terminal session with a log of network traffic. The log lists numerous connections between the host (192.168.56.102) and the target (192.168.56.1), primarily involving HTTP and FTP protocols.
- A file explorer window titled "C:\Users\hotdo\Downloads>" shows various files and folders, including "DAX Studio", "Anki", "Liberar Espacio", "code", "Solicitud Simple...", and "VLC media player".

The desktop background features a blue circuit board pattern. A status bar at the bottom of the terminal window reads "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net".

# RESULTADOS

**Resultados:**  
**Paquetes enviados: 15**  
**Respuestas recibidas: 10**  
**Respuestas fallidas: 5**  
**Throughput: 0.58 paquetes/segundo**  
**Resultados guardados en 'tcp\_syn\_results.csv'**

## PRUEBA 2: GENERADOR TCP SYN

No.	Time	Source	Destination	Protocol	Length Info
59	48.928735	192.168.56.1	192.168.56.102	TCP	54 20 → 80 [SYN] Seq=0 Win=8192 Len=0
60	48.929025	192.168.56.102	192.168.56.1	TCP	60 80 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
61	49.233417	192.168.56.1	192.168.56.102	TCP	54 [TCP Retransmission] 20 → 80 [SYN] Seq=0 Win=8192 Len=0
62	49.234118	192.168.56.102	192.168.56.1	TCP	60 [TCP Retransmission] 80 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
63	49.541250	192.168.56.1	192.168.56.102	TCP	54 [TCP Retransmission] 20 → 80 [SYN] Seq=0 Win=8192 Len=0
64	50.261160	192.168.56.102	192.168.56.1	TCP	60 [TCP Retransmission] 80 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
65	50.564604	192.168.56.1	192.168.56.102	TCP	54 [TCP Retransmission] 20 → 80 [SYN] Seq=0 Win=8192 Len=0
66	50.565737	192.168.56.102	192.168.56.1	TCP	60 [TCP Retransmission] 80 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
67	50.878692	192.168.56.1	192.168.56.102	TCP	54 [TCP Retransmission] 20 → 80 [SYN] Seq=0 Win=8192 Len=0
68	52.597220	192.168.56.102	192.168.56.1	TCP	60 [TCP Retransmission] 80 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
69	52.901162	192.168.56.1	192.168.56.102	TCP	54 [TCP Retransmission] 20 → 80 [SYN] Seq=0 Win=8192 Len=0
70	52.902001	192.168.56.102	192.168.56.1	TCP	60 [TCP Retransmission] 80 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
71	53.206815	192.168.56.1	192.168.56.102	TCP	54 [TCP Retransmission] 20 → 80 [SYN] Seq=0 Win=8192 Len=0
74	55.518890	192.168.56.1	192.168.56.102	TCP	54 [TCP Retransmission] 20 → 80 [SYN] Seq=0 Win=8192 Len=0
75	55.519806	192.168.56.102	192.168.56.1	TCP	60 [TCP Retransmission] 80 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
76	55.825107	192.168.56.1	192.168.56.102	TCP	54 [TCP Retransmission] 20 → 80 [SYN] Seq=0 Win=8192 Len=0
77	58.131418	192.168.56.1	192.168.56.102	TCP	54 [TCP Retransmission] 20 → 80 [SYN] Seq=0 Win=8192 Len=0
78	58.135172	192.168.56.102	192.168.56.1	TCP	60 [TCP Retransmission] 80 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
79	58.441464	192.168.56.1	192.168.56.102	TCP	54 [TCP Retransmission] 20 → 80 [SYN] Seq=0 Win=8192 Len=0
80	60.760365	192.168.56.1	192.168.56.102	TCP	54 [TCP Retransmission] 20 → 80 [SYN] Seq=0 Win=8192 Len=0
81	60.761400	192.168.56.102	192.168.56.1	TCP	60 [TCP Retransmission] 80 → 20 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
82	61.066061	192.168.56.1	192.168.56.102	TCP	54 [TCP Retransmission] 20 → 80 [SYN] Seq=0 Win=8192 Len=0
83	63.388643	192.168.56.1	192.168.56.102	TCP	54 [TCP Retransmission] 20 → 80 [SYN] Seq=0 Win=8192 Len=0

Frame 59: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF\_{87D4D3C8-3... 0000 08 00 27 da 72 4a 0a 00 27 00 00 0b 08 00 45 00 ...'rJ... 'E...  
 Ethernet II, Src: 0a:00:27:00:00:0b (0a:00:27:00:00:0b), Dst: PCSSystemtec\_da:72:4a (08:00:27:da:72:4a)  
 Internet Protocol Version 4, Src: 192.168.56.1, Dst: 192.168.56.102  
 Transmission Control Protocol, Src Port: 20, Dst Port: 80, Seq: 0, Len: 0



# RESULTADOS

## PRUEBA 3: GENERADOR DE CARGA CON EVALUACIÓN AVANZADA

```
from scapy.all import *
import random
import time

def load_test(target_ip, duration=30):
    start_time = time.time()
    packet_count = 0

    print(f"Iniciando prueba de carga contra {target_ip} durante {duration} segundos...")

    while time.time() - start_time < duration:
        payload_size = random.randint(64, 1500) # Tamaño del paquete
        packet = IP(dst=target_ip) / UDP() / ("X" * payload_size)
        send(packet, verbose=False)
        packet_count += 1
        time.sleep(random.uniform(0.01, 0.1)) # Intervalo aleatorio

    print(f"Prueba completada. Paquetes enviados: {packet_count}")

# Uso
target_ip = "192.168.56.102"
load_test(target_ip, duration=60)
```



# RESULTADOS

## PRUEBA 3: GENERADOR DE CARGA CON EVALUACIÓN AVANZADA

The screenshot shows a Windows desktop environment with several open windows:

- Command Prompt Window:** Shows the output of a packet capture or test script. It lists 15 packages sent and 10 received ACKs, with 5 failing. Throughput is 0.58 packets/second. Results are saved to 'tcp\_syn\_results.csv'.
- File Explorer Window:** Shows the directory structure at '/home/gns3/'. It includes files like .cache, .config, .local, .ssh, .venv, GNS3, .bash\_history, .bash\_logout, .bash\_profile, .bashrc, .profile, .python\_history, .sudo\_as\_admin\_successful, .wget-hsts, and .Xauthority.
- Terminal Window:** Shows the results of running 'tcpdump -i eth0 tcp port 80' on a host named 'gns3'. It captures 28 packets, receives 28 by filter, and drops 0 by kernel. It also shows the command to start a TCP dump on port 80.

At the bottom of the terminal window, there is a watermark: "UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>".

# RESULTADOS

## PRUEBA 3: GENERADOR DE CARGA CON EVALUACIÓN AVANZADA

```

Capturando desde interfaz
Archivo Edición Visualización Ir Captura Analizar Estadísticas Teléfono Wireless Herramientas Ayuda
udp
No. udp udp[...]
Source Destination Protocol Length Info
192.168.56.1 224.0.0.251 MDNS 388 Standard query response 0x0000 PTR MSI._dosvc._tcp.local SRV 0 0 7680 MSI.local TXT
192.168.56.1 224.0.0.251 MDNS 328 Standard query response 0x0000 PTR MSI._dosvc._tcp.local SRV 0 0 7680 MSI.local TXT
192.168.56.1 224.0.0.251 MDNS 81 Standard query 0x0000 ANY MSI._dosvc._tcp.local, "QM" question
192.168.56.1 224.0.0.251 MDNS 101 Standard query 0x0000 ANY MSI._dosvc._tcp.local, "QM" question
192.168.56.1 224.0.0.251 MDNS 81 Standard query 0x0000 ANY MSI._dosvc._tcp.local, "QM" question
192.168.56.1 224.0.0.251 MDNS 101 Standard query 0x0000 ANY MSI._dosvc._tcp.local, "QM" question
192.168.56.1 224.0.0.251 MDNS 81 Standard query 0x0000 ANY MSI._dosvc._tcp.local, "QM" question
192.168.56.1 224.0.0.251 MDNS 101 Standard query 0x0000 ANY MSI._dosvc._tcp.local, "QM" question
192.168.56.1 224.0.0.251 MDNS 361 Standard query response 0x0000 PTR, cache flush MSI._dosvc._tcp.local SRV, cache flush 0 0 7680 MSI.local TXT, cache flush A, cache
192.168.56.1 224.0.0.251 MDNS 381 Standard query response 0x0000 PTR, cache flush MSI._dosvc._tcp.local SRV, cache flush 0 0 7680 MSI.local TXT, cache flush A, cache
192.168.56.1 224.0.0.251 MDNS 309 Standard query response 0x0000 SRV, cache flush 0 0 7680 MSI.local TXT, cache flush A, cache flush 192.168.56.1 AAAA, cache flush F,
192.168.56.1 224.0.0.251 MDNS 329 Standard query response 0x0000 SRV, cache flush 0 0 7680 MSI.local TXT, cache flush A, cache flush 192.168.56.1 AAAA, cache flush F,
192.168.56.1 192.168.56.102 DNS 1270 Unknown operation (11) 0x5858[Malformed Packet]
192.168.56.1 192.168.56.1 ICMP 590 Destination unreachable (Port unreachable)
192.168.56.1 192.168.56.102 DNS 1388 Unknown operation (11) 0x5858[Malformed Packet]
192.168.56.1 192.168.56.102 ICMP 590 Destination unreachable (Port unreachable)
192.168.56.1 192.168.56.102 DNS 229 Unknown operation (11) 0x5858[Malformed Packet]
192.168.56.1 192.168.56.102 ICMP 257 Destination unreachable (Port unreachable)
192.168.56.1 192.168.56.102 DNS 1435 Unknown operation (11) 0x5858[Malformed Packet]
192.168.56.1 192.168.56.102 ICMP 590 Destination unreachable (Port unreachable)
192.168.56.1 192.168.56.102 DNS 554 Unknown operation (11) 0x5858[Malformed Packet]
192.168.56.1 192.168.56.102 ICMP 582 Destination unreachable (Port unreachable)
192.168.56.1 192.168.56.102 DNS 484 Unknown operation (11) 0x5858[Malformed Packet]

Frame 56: 329 bytes on wire (2632 bits), 329 bytes captured (2632 bits) on interface \Device\NPF_{87D4D0}
Ethernet II, Src: 0a:00:27:00:00:0b (0a:00:27:00:00:0b), Dst: IPv6cast_fb (33:33:00:00:00:fb)
Internet Protocol Version 6, Src: fe80::20af:bf29:beb0:5269, Dst: ff02::fb
User Datagram Protocol, Src Port: 5353, Dst Port: 5353
Multicast Domain Name System (response)

0000 33 33 00 00 00 fb 0a 00 27 00 00 0b 06 dd 60 02 33 ..... *
0010 95 7d 01 13 11 01 fe 00 00 00 00 00 00 20 xf .. .
0020 bf 24 8c 68 52 69 ff 02 00 00 00 00 00 00 00 Ri .. .
0030 00 00 00 00 00 fb 14 v9 14 v9 01 13 60 09 00 00 .....
0040 84 00 00 00 01 00 00 00 03 03 4d 53 49 06 5f ..... MSI_ ...
0050 64 6f 73 76 63 04 5f 74 63 70 05 6c 6f 63 61 6c dosvc_t cp local
0060 00 00 21 80 01 00 00 00 78 00 11 00 00 00 00 1e .. ! ... x ..
0070 00 03 4d 53 49 05 6c 6f 63 61 60 00 c0 00 10 .. MSI_lo cal ...
0080 00 01 00 00 11 94 00 00 8c 07 50 3d 36 35 32 38 30 .. Pm65280
0090 15 53 48 30 3d 55 2f 64 45 2f 73 4e 46 77 64 .. SH80-U/ nE/sNFwd
00a0 4f 57 6a 55 70 54 15 53 48 30 31 3d 57 66 52 67 .. OWjUpT-S H01-WFRg

```

# CONCLUSIONES

## Resultados clave:

- Tráfico ICMP: Latencia promedio de 58.61 ms (ligeramente elevada), jitter alto de 129.83 ms y tasa de pérdida de paquetes del 0.00%. Picos de latencia (>100 ms) sugieren congestión transitoria.
- Conexiones TCP al puerto 80: Tasa de respuesta del 66.67%, con 33.33% de fallos atribuidos a firewall, congestión o limitaciones del servicio. Throughput moderado de 0.58 paquetes/segundo.
- Cargas intensivas: Identificaron límites y puntos débiles en la capacidad de manejo de tráfico.

## Recomendaciones:

1. Monitoreo continuo de latencia y jitter para detectar patrones de congestión.
2. Uso de traceroute para mitigar cuellos de botella.
3. Optimización del firewall y asignación de más recursos al servicio en el puerto objetivo.
4. Repetir pruebas periódicamente para evaluar la estabilidad bajo alta demanda.



# Bibliografía

Hussain, F., Abbas, S. G., Shah, G. A., Pires, I. M., Fayyaz, U. U., Shahzad, F., García, N. M., & Zdravevski, E. (2021). A framework for malicious traffic detection in IoT healthcare environment. *Sensors*, 21(9), 3025.

<https://doi.org/10.3390/s21093025>

Rincón, L. (2021). Test de penetración para el estudio de vulnerabilidades a los ciberataques mediante técnicas de hacking ético en redes IPv4. *Revista Electrónica de Estudios Telemáticos*, 20(2).



# GRACIAS!

