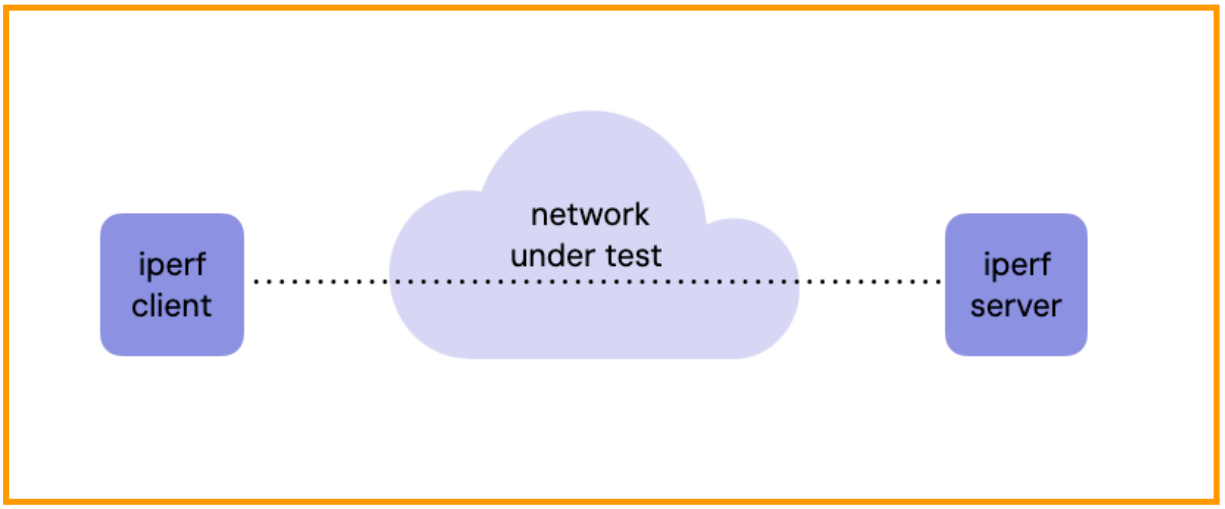
ASSIGNMENT 4

Software Engineering Lab

Member	Enrollment No
Ramesh Chandra Soren	2022CSB086
Moodu Roopa	2022CSB087
Jitendra Kumar Dodwadiya	2022CSB089
Deep Sutariya	2022CSB090



iPerf: A Network Performance Testing Tool

1. Purpose of the Tool

iPerf is a widely used network testing tool designed to measure and analyze the performance of network connections. It helps in determining the bandwidth, throughput, and other performance metrics between two hosts by generating and analyzing TCP, UDP, and SCTP traffic. Network engineers and administrators use iPerf for network tuning, troubleshooting, and performance optimization.

Key Features:

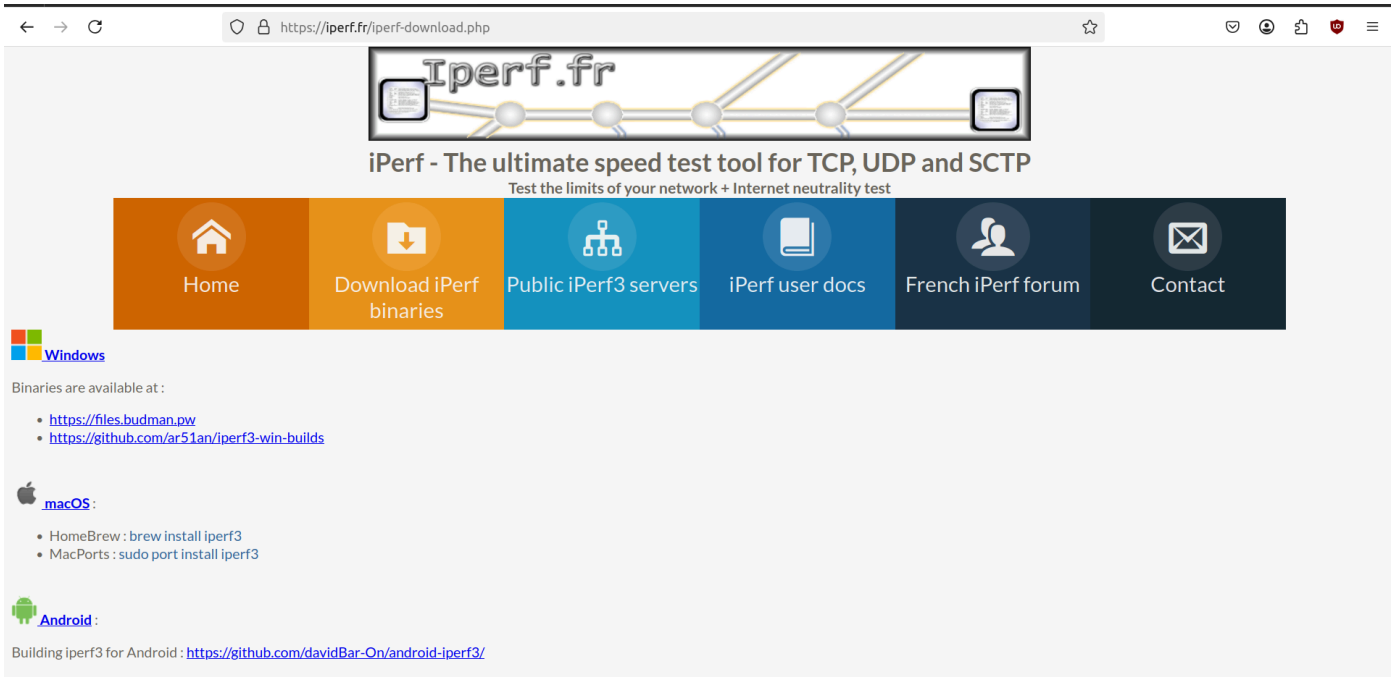
- Measures TCP and UDP bandwidth, jitter, and packet loss.
- Supports multithreaded testing and parallel streams.
- Allows testing in both client-server mode.
- Works across various network protocols like IPv4 and IPv6.
- Supports bidirectional and reverse mode testing.

---

2. Software Download Path

iPerf is open-source and available for free download from multiple sources:

- Official iPerf website: https://iperf.fr/



- GitHub repository: https://github.com/esnet/iperf
- Linux package managers:
    - Debian/Ubuntu: apt install iperf3
    - CentOS/RHEL: yum install iperf3
    - Arch Linux: pacman -S iperf3

---

3. Target Platform and Installation Procedure

iPerf is cross-platform and supports:

- Windows
- Linux
- macOS
- FreeBSD
- Android

Installation Steps

On Windows:

1. Download the iPerf executable from https://iperf.fr/iperf-download.php.
2. Extract the ZIP file.
3. Open Command Prompt and navigate to the extracted folder.
4. Run iperf3.exe to check the installation.

On Linux/macOS:

1. Open a terminal and install using a package manager:
    - Ubuntu/Debian: sudo apt install iperf3
    - Fedora/CentOS: sudo yum install iperf3
    - Arch Linux: sudo pacman -S iperf3
2. Verify installation: iperf3 --version

---

4. Commands to Configure and Run the Tool

iPerf runs in a client-server mode, where one machine acts as a server, and another machine acts as a client.

Basic Usage

1. Start iPerf Server (on one machine): `iperf3 -s`

```
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ sudo apt install iperf
[sudo] password for ramesh:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  libpugixml1v5
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  iperf
0 upgraded, 1 newly installed, 0 to remove and 529 not upgraded.
Need to get 121 kB of archives.
After this operation, 315 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 iperf amd64 2.1.5+dfsg1-1 [121 kB]
Fetched 121 kB in 9s (12.8 kB/s)
Selecting previously unselected package iperf.
(Reading database ... 652217 files and directories currently installed.)
Preparing to unpack .../iperf_2.1.5+dfsg1-1_amd64.deb ...
Unpacking iperf (2.1.5+dfsg1-1) ...
Setting up iperf (2.1.5+dfsg1-1) ...
Processing triggers for ufw (0.36.1-4ubuntu0.1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for doc-base (0.11.1) ...
Processing 1 added doc-base file...
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  1] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 45358
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0008 sec   110 GBytes  94.3 Gbits/sec
```

Run iPerf Client (on another machine, specifying the server's IP address):

`iperf3 -c <server-IP>`

```
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ iperf -c 127.0.0.1
------------------------------------------------------------
Client connecting to 127.0.0.1, TCP port 5001
TCP window size: 2.50 MByte (default)
------------------------------------------------------------
[  1] local 127.0.0.1 port 45358 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0105 sec   110 GBytes  94.3 Gbits/sec
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$
```

From where did i get the server-ip:

Since I am using my own device as server as well as client, I have used a loopback address to find my server's address.

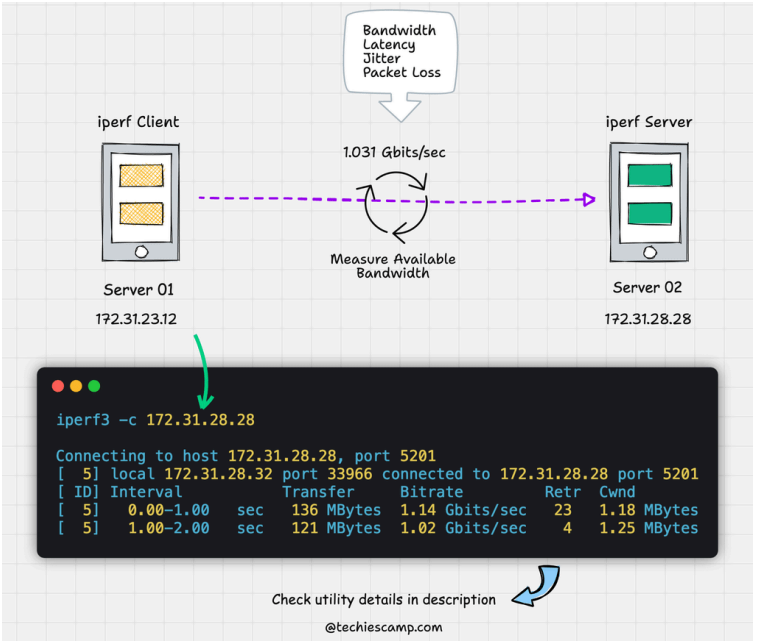
```
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~/Desktop$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:0e:2d:b0:0f  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eno1: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 5c:60:ba:64:6b:29  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 486  bytes 47538 (47.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 486  bytes 47538 (47.5 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp0s20f3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.54  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::1c02:13fb:63ce:19  prefixlen 64  scopeid 0x20<link>
        inet6 2401:4900:3dab:6967:cb23:dc75:a6b3:71b0  prefixlen 64  scopeid 0x0<global>
        inet6 2401:4900:3dab:6967:d14f:22c5:d103:2889  prefixlen 64  scopeid 0x0<global>
        ether 54:6c:eb:b8:0a:c9  txqueuelen 1000  (Ethernet)
        RX packets 2386  bytes 1432158 (1.4 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1971  bytes 666331 (666.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~/Desktop$
```



Common iPerf Commands

- Measure TCP bandwidth (default):

  iperf3 -c <server-IP>

```
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ iperf -c 127.0.0.1
------------------------------------------------------------
Client connecting to 127.0.0.1, TCP port 5001
TCP window size: 2.50 MByte (default)
------------------------------------------------------------
[  1] local 127.0.0.1 port 52478 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0096 sec   112 GBytes  96.5 Gbits/sec
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$
```

- Run in reverse mode (server to client):

  iperf3 -c <server-IP> -R

```
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ iperf -c 127.0.0.1 -R
------------------------------------------------------------
Client connecting to 127.0.0.1, TCP port 5001
TCP window size: 2.50 MByte (default)
------------------------------------------------------------
[  1] local 127.0.0.1 port 43260 connected with 127.0.0.1 port 5001 (reverse)
[ ID] Interval       Transfer     Bandwidth
[ *1] 0.0000-10.0003 sec   112 GBytes  96.4 Gbits/sec
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$
```

- Test UDP bandwidth (default 1 Mbps):

  iperf3 -u -c <server-IP>

  # Start the UDP Server: Open a new terminal and run

```
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ iperf -s -u
------------------------------------------------------------
Server listening on UDP port 5001
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  1] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 41664
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  1] 0.0000-10.0154 sec  1.25 MBytes  1.05 Mbits/sec   0.006 ms 0/895 (0%)
```

  This command starts the iPerf server in UDP mode on the default port (5001 for iPerf version 2).

Run the UDP Client: In another terminal, run:

```
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ iperf -u -c 127.0.0.1
------------------------------------------------------------
Client connecting to 127.0.0.1, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  1] local 127.0.0.1 port 41664 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0155 sec  1.25 MBytes  1.05 Mbits/sec
[  1] Sent 896 datagrams
[  1] Server Report:
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  1] 0.0000-10.0154 sec  1.25 MBytes  1.05 Mbits/sec   0.005 ms 0/895 (0%)
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$
```

This client command will now connect to the server at 127.0.0.1 on UDP port 5001.

Port Specification:

- If you need to specify a custom port, you can add the `-p` option. For example, for iPerf (v2):

bash Copy Edit

```
iperf -s -u -p 5002
iperf -u -c 127.0.0.1 -p 5002
```

- Set custom bandwidth for UDP (e.g., 10 Mbps):

iperf3 -u -c <server-IP> -b 10M

Start the iPerf3 Server (on your test machine):

```
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ iperf -s -u
------------------------------------------------------------
Server listening on UDP port 5001
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  1] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 40309
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  1] 0.0000-10.0017 sec  12.5 MBytes  10.5 Mbits/sec   0.001 ms 0/8920 (0%)
```

Ensure the server is running before starting the client.

Run the Client Command: Open another terminal and execute:

```
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ iperf -u -c 127.0.0.1 -b 10M
------------------------------------------------------------
Client connecting to 127.0.0.1, UDP port 5001
Sending 1470 byte datagrams, IPG target: 1121.52 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  1] local 127.0.0.1 port 40309 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0018 sec  12.5 MBytes  10.5 Mbits/sec
[  1] Sent 8921 datagrams
[  1] Server Report:
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  1] 0.0000-10.0017 sec  12.5 MBytes  10.5 Mbits/sec   0.000 ms 0/8920 (0%)
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$
```

This command will run a UDP test from the client to the server at a specified bandwidth of 10 Mbps.

- Run test for a specific duration (e.g., 60 seconds):

  iperf3 -c <server-IP> -t 60

  Start the server but 1st kill it

```
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ iperf -s
listener bind failed: Address already in use
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ pkill iperf
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
```

```
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ iperf -c 127.0.0.1 -t 60
------------------------------------------------------------
Client connecting to 127.0.0.1, TCP port 5001
TCP window size: 2.50 MByte (default)
------------------------------------------------------------
[  1] local 127.0.0.1 port 42188 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-60.0079 sec   668 GBytes  95.6 Gbits/sec
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$
```

`-t 60`: Sets the duration of the test to 60 seconds.

After 60 sec on the server side.

```
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ iperf -s
listener bind failed: Address already in use
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ pkill iperf
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  1] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 42188
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-60.0008 sec   668 GBytes  95.6 Gbits/sec
```

- Enable multiple parallel streams (e.g., 4 streams):

  iperf3 -c <server-IP> -P 4

  When and Why to Use Multiple Streams:

  - **Simulate Congestion:** Multiple streams simulate a more realistic, congested network scenario.
  - **Increase Throughput:** If a single stream cannot saturate your network link due to protocol limitations or TCP window scaling issues, multiple streams might help achieve higher throughput.
  - **Performance Analysis:** It gives you an aggregated view of performance across multiple simultaneous connections, which is useful in network testing and troubleshooting.

  Start the server:

```
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
[  1] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 52660
[  2] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 52690
[  3] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 52674
[  4] local 127.0.0.1 port 5001 connected with 127.0.0.1 port 52694
[ ID] Interval       Transfer     Bandwidth
[  3] 0.0000-10.0004 sec  74.1 GBytes  63.7 Gbits/sec
[  1] 0.0000-10.0029 sec  70.7 GBytes  60.7 Gbits/sec
[  2] 0.0000-9.9999 sec  71.8 GBytes  61.7 Gbits/sec
[  4] 0.0000-9.9987 sec  70.9 GBytes  60.9 Gbits/sec
[SUM] 0.0000-10.0018 sec   287 GBytes   247 Gbits/sec
```

```
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ iperf -c 127.0.0.1 -P 4
------------------------------------------------------------
Client connecting to 127.0.0.1, TCP port 5001
TCP window size: 2.50 MByte (default)
------------------------------------------------------------
[  4] local 127.0.0.1 port 52674 connected with 127.0.0.1 port 5001
[  1] local 127.0.0.1 port 52660 connected with 127.0.0.1 port 5001
[  3] local 127.0.0.1 port 52694 connected with 127.0.0.1 port 5001
[  2] local 127.0.0.1 port 52690 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  4] 0.0000-10.0097 sec  74.1 GBytes  63.6 Gbits/sec
[  1] 0.0000-10.0097 sec  70.7 GBytes  60.7 Gbits/sec
[  2] 0.0000-10.0097 sec  71.8 GBytes  61.6 Gbits/sec
[  3] 0.0000-10.0098 sec  70.9 GBytes  60.8 Gbits/sec
[SUM] 0.0000-10.0000 sec   287 GBytes   247 Gbits/sec
[ CT] final connect times (min/avg/max/stdev) = 0.058/0.072/0.090/0.015 ms (tot/err) = 4/0
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$
```

This will initiate 4 parallel streams to the server at **`127.0.0.1`**.

- Run as a daemon (background process):

  iperf3 -s -D

```
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ iperf -s -D
Running Iperf Server as a daemon
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$
```

We Tested the server that is running in background

```
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$ iperf -c 127.0.0.1
------------------------------------------------------------
Client connecting to 127.0.0.1, TCP port 5001
TCP window size: 2.50 MByte (default)
------------------------------------------------------------
[  1] local 127.0.0.1 port 57396 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-10.0130 sec   112 GBytes  95.9 Gbits/sec
ramesh@ramesh-Victus-by-HP-Gaming-Laptop-15-fa0xxx:~$
```

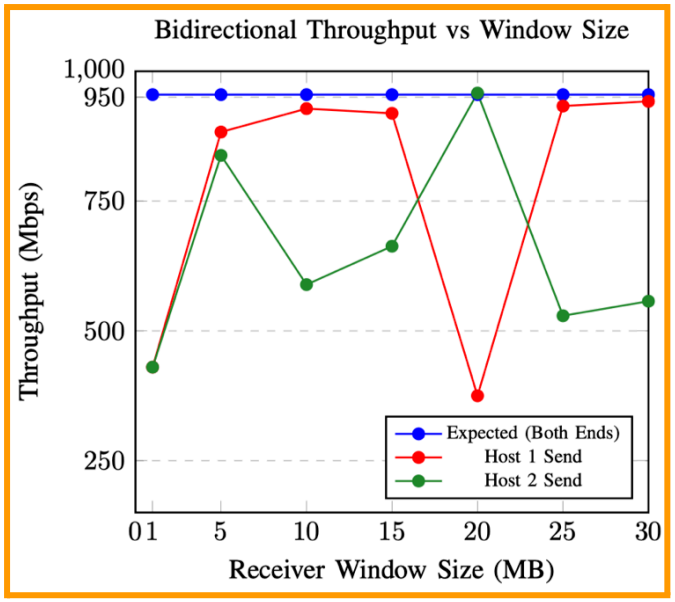
5. Case Studies – Experiments Using iPerf

iPerf is widely used in networking and system administration.

Some common experiments include:

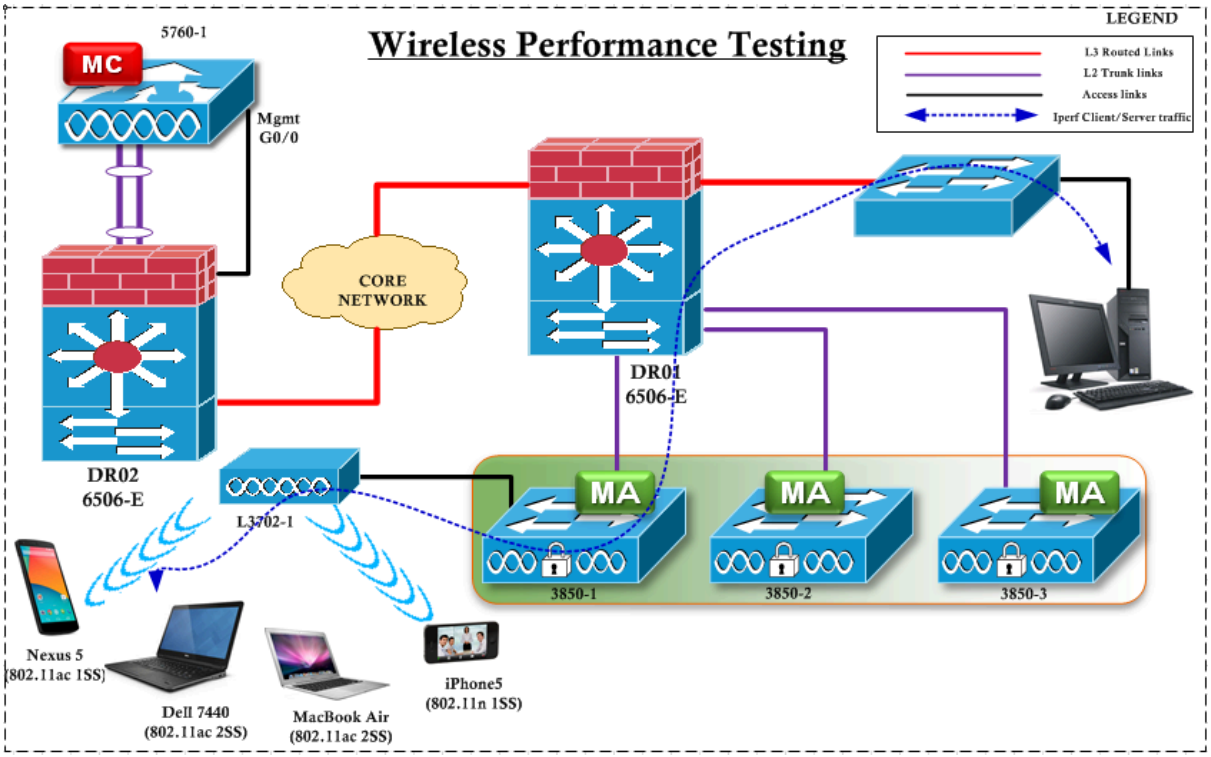
A. Network Throughput Analysis

- Used by data center engineers to evaluate **LAN and WAN performance**.
- Helps identify **network congestion** and optimize routing.



B. Wireless Network Performance Testing

- Researchers use iPerf to test **Wi-Fi performance** in different environments.
- Helps in evaluating **signal strength, interference, and latency** in 2.4 GHz vs. 5 GHz bands.



C. Cloud and VPN Performance Measurement

- Companies use iPerf to compare **VPN performance before and after optimizations.**
- Cloud providers analyze **network speeds across data centers.**

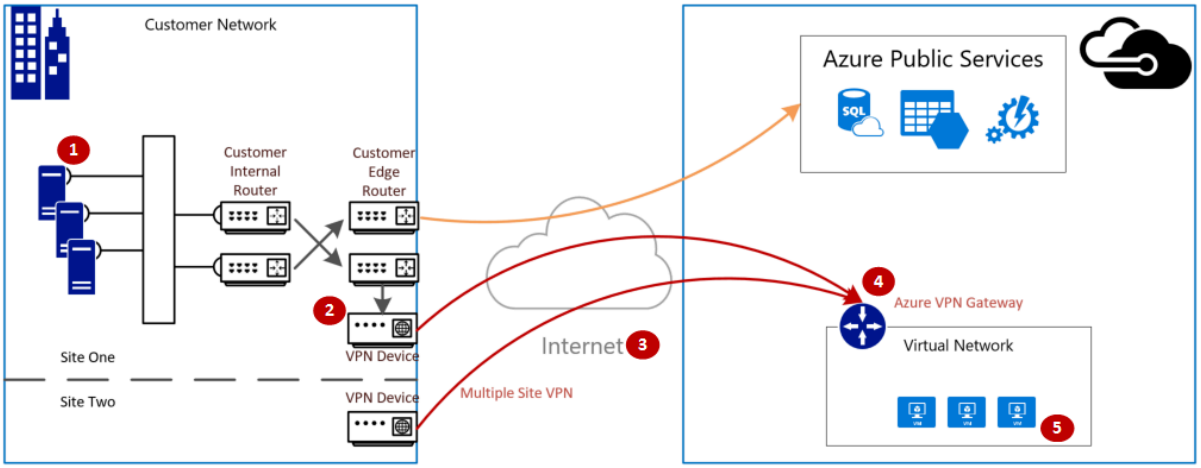


Diagram Purpose: Shows a multi-site VPN connection to Azure.

How iPerf Could Fit Into This Diagram

1. **On-Premises Client:** A server or workstation on Site One or Site Two running the iPerf client.
2. **Azure VM iPerf Server:** A virtual machine in the Azure Virtual Network running the iPerf server.
3. **Test Traffic Flow:** The iPerf traffic would traverse the VPN devices, the internet, and the Azure VPN gateway, allowing you to measure **real-world throughput** and **latency**.

Such a setup can help verify if your **ISP bandwidth** or **VPN configuration** is delivering the expected performance.

D. ISP Bandwidth Verification

- End users test if their ISP provides the advertised internet speed.
- Measures **upload and download bandwidth** using remote iPerf servers.

6. Drawbacks of iPerf

While iPerf is a powerful tool, it has some limitations:

- Lacks real-world traffic simulation: Unlike other tools like Wireshark or NetFlow, iPerf doesn't generate real-world application traffic.
- Limited GUI support: iPerf is mostly command-line based, which may not be user-friendly for beginners.
- No built-in latency measurement: While it measures jitter, it doesn't provide precise latency calculations like ICMP-based tools.
- High CPU utilization: Running multiple parallel streams can consume significant CPU resources.

7. Additional Aspects

- Alternatives to iPerf:
    - Netcat (nc) – Simple TCP/UDP testing.
    - Flent – Provides GUI-based network tests.
    - Speedtest-cli – Measures internet speed using Ookla servers.
- Best Practices:
    - Always test with multiple streams for better accuracy.
    - Run tests at different times to detect congestion patterns.
    - Use iPerf on a wired connection to avoid Wi-Fi interference.

8. Conclusion

iPerf is an essential tool for network performance testing and troubleshooting.

It provides detailed bandwidth analysis, helps diagnose bottlenecks, and is widely used by IT professionals and researchers.

Despite its limitations, iPerf remains a go-to tool for network benchmarking due to its simplicity and effectiveness.