AN1434: SiWx917 NCP Throughput Application Note

This document provides information about the measurement of SiWx917 throughput performance, and the throughput observed in different protocols like TCP, UDP, and TLS transmit (Tx) and receive (Rx) throughput of SiWx917 using SAPIs on the host MCU. Here, the SiWx917 is configured in Wi-Fi station mode and connects to an access point. The SiWx917 would then connect to an iPerf server or client running on a PC connected to the same access point.

KEY POINTS

- Setup Requirements and Diagram
- Socket Configuration
- Supported Protocols
- Wireshark I/O Graphs

Table of Contents

1. Introduction

In the Wireless communication, the term “Throughput” is defined as the number of data units transferred within a specified amount of time over a communication channel and reflects how the network is performing. In general, throughput is measured in bit/s or bps i.e., the number of bits transferred in one second.

Many applications need to transfer a burst of data quickly over their Wi-Fi link before returning to sleep or wait state. The user measures the throughput that can be sustained by their device. This document provides information about the measurement of SiWx917 throughput performance, and the throughput observed in different protocols like TCP, UDP, and TLS transmit (Tx) and receive (Rx) throughput of SiWx917 using SAPIs on the host MCU. Here, the SiWx917 is configured in Wi-Fi station mode and connects to an access point. The SiWx917 would then connect to an iPerf server or client running on a PC connected to the same access point.

Note: For NCP, SiW**x**917 would be referred to as SiW**N**917 in the following sections of the document.

2. Setup Requirements

2.1 Hardware Requirements

S. No	Item	Quantity	Requirement
1	Windows PC	1	To run the throughput application
2	Access Point	1	To connect the Wireless devices Example: ASUS RT-AC53. FW Version: 3.0.0.4.380_10446 or above
3	Windows PC / Laptop	1	To run the iPerf or Python server/client
4	Silicon Labs NCP radio board	1	BRD4346A
5	Power and USB cables	1	To connect the NCP module and the host MCU to the PC
For EFR32xG24 as Host MCU			
6	Expansion Adapter board	1	BRD8045A
7	Host MCU	1	BRD4002A WPK + EFR32xG24 (10 dBm Radio Board)
For STM32 as Host MCU			
8	Expansion Adapter board	1	BRD8045C
9	Host MCU	1	STM32F411RE

2.2 Software Requirements

S. No	Item	Test Tool
1	Simplicity Studio IDE (for EFR as host MCU)	Simplicity Studio
2	Keil IDE (for STM32 as host MCU)	Keil
3	IPerf Application, Version: 2.0.9 and above	IPerf
4	Python environment, Version 2 and above	Python

Follow the Getting Started with NCP mode guide to set up the Simplicity Studio IDE and start with the example.

3. Setup Diagram

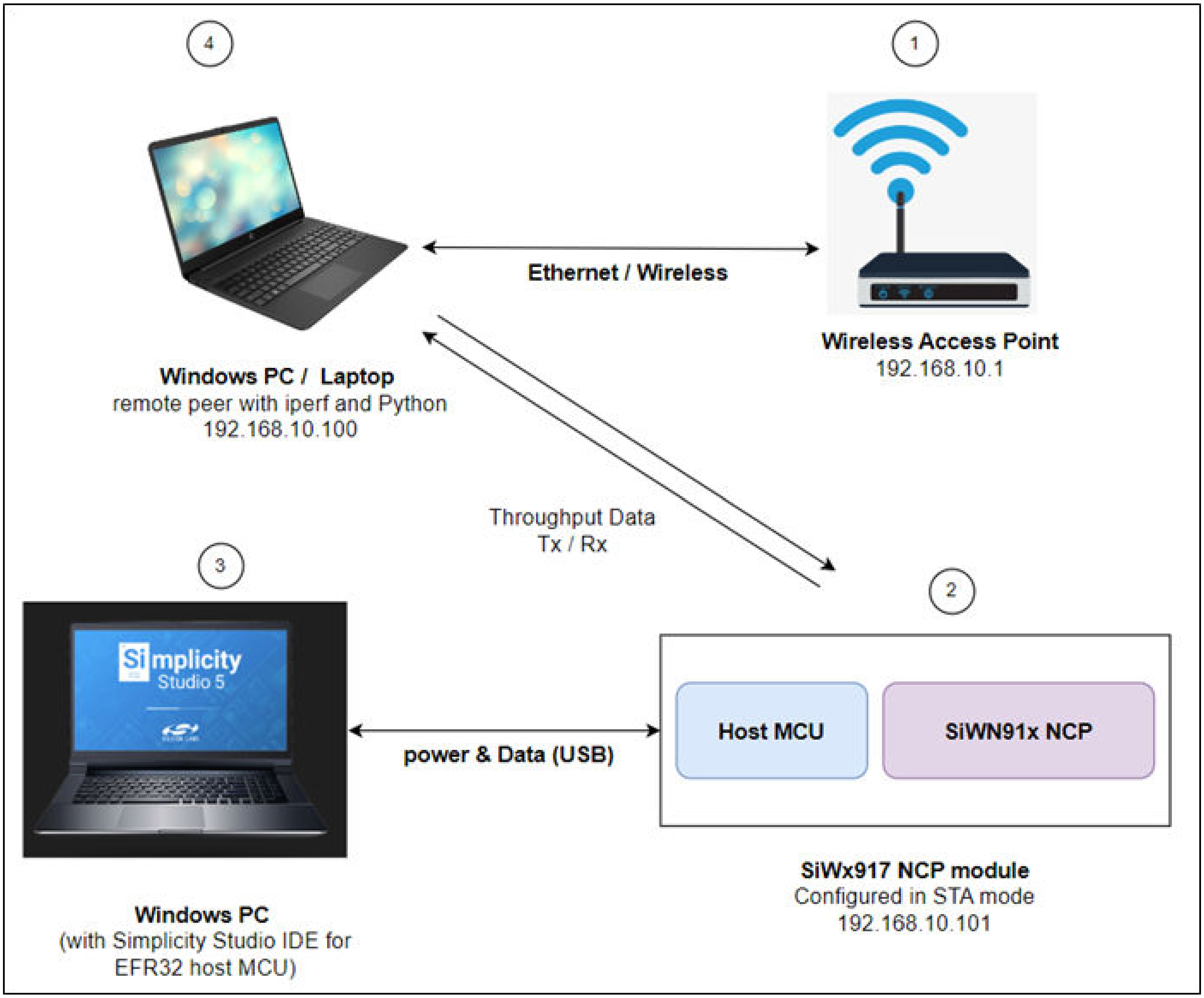


Figure 3.1. Throughput Setup Diagram

4. Network Stack

The SiWN917 Network Stack modes are as shown below.

1. Offloaded Mode
2. Hosted Mode

By default, the TCP/IP Stack runs on the TA (Network Processor) which is the Offloaded mode. If you want to bypass it and use the TCP/IP stack residing on the host processor, which is the Hosted mode, you can refer to the **LWIP TCP client** example in the release.

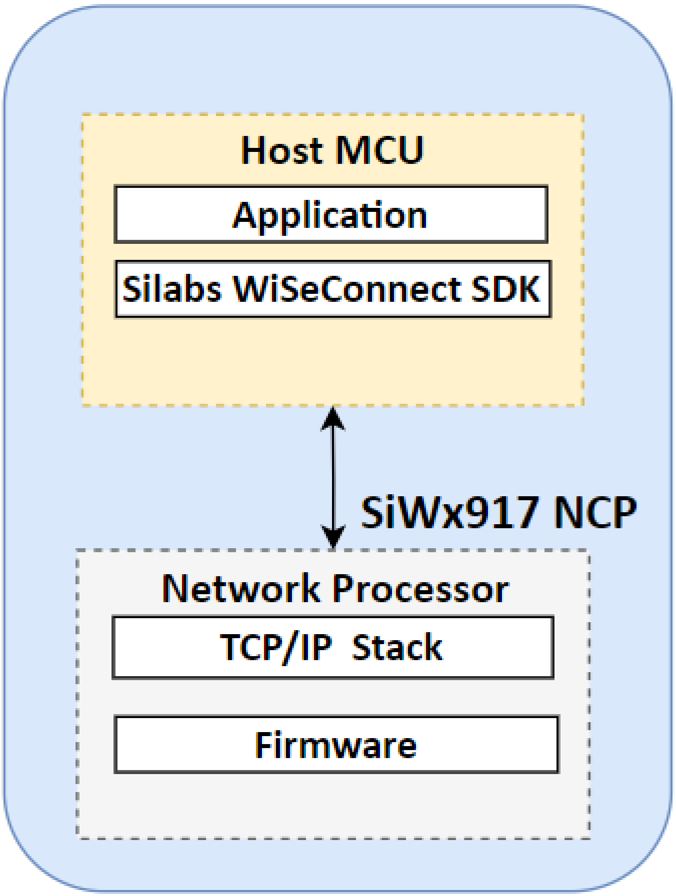


Figure 4.1. Offloaded Mode

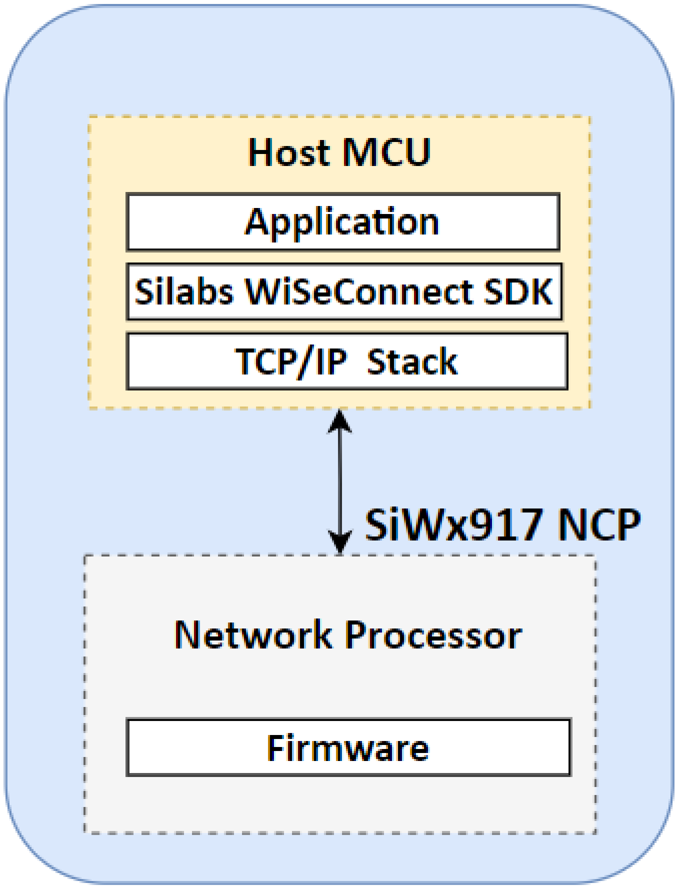


Figure 4.2. Hosted Mode

5. Socket Configuration

The SiWN917 supports two modes of socket configurations:

1. BSD Socket Configuration
2. Asynchronous Socket Configuration

5.1 BSD Socket

The BSD sockets application programming interface (API) is a set of standard function calls that can be used in an application. They allow programmers to add Internet Protocol communication to their products.

BSD Socket configuration is synchronized (the data is sent synchronously). Synchronous is a blocking architecture and is best for programming reactive systems. As a single-thread model, it follows a strict set of sequences, which means that operations are performed one at a time, in perfect order. While one operation is being performed, other operations instructions are blocked. The completion of the first task triggers the next, and so on.

The BSD socket implementation in SiWN917 tries to emulate the standards library to the extent possible for embedded offerings from Silicon Labs. There are substantial limitation/exceptions however, and those are mentioned in the API Reference Guide documentation.

5.2 Asynchronous Socket

The asynchronous socket is a non-blocking architecture, which means it doesn't block further execution while one or more operations are in progress. With async programming, multiple related operations can run concurrently without waiting for other tasks to complete.

Any scenario where you need to handle any significant number of concurrent connections, the asynchronous APIs are the only ones that provide adequate performance. In any interactive scenario (i.e., where you must deal with user input and output), the asynchronous approach is more easily integrated with the user interface. Async programming translates to a faster, more seamless flow in the real world.

In the SiWN917 asynchronous socket configuration, the data is sent or received asynchronously with a callback registered.

6. Supported Protocols

The SiWN917 supports the following protocols:

1. User Datagram Protocol (UDP)
2. Transmission Control Protocol (TCP)
3. Transport Layer Security (TLS)

6.1 User Datagram Protocol (UDP)

UDP is a simple, connectionless internet protocol where error-checking and recovery services are not required. It is a lightweight protocol with no support for handshaking, ordering, error recovery etc., and is usually used in protocols that stream music or video. When data transfer is over the internet, packets may be routed through dozens of intermediate nodes as they are sent across the world. Some routers may drop some packets if there is congestion. That is the reason why TCP provides an ACK and retry mechanism. If data is to be sent to a nearby device (local network), then UDP is perfectly reliable. Many implementations of sensor data transfer to a local server could use UDP for simplicity and power savings.

UDP is a preferred protocol where reliability can be traded-off for low end-to-end delay experienced by the users. With UDP, there is no overhead for opening a connection, maintaining a connection, or terminating a connection. Data is continuously sent to the recipient, whether they receive it or not.

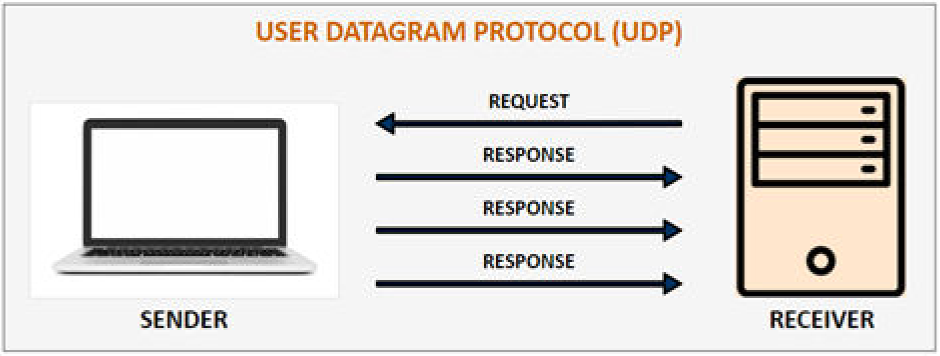


Figure 6.1. User Datagram Protocol (UDP)

6.1.1 UDP Tx

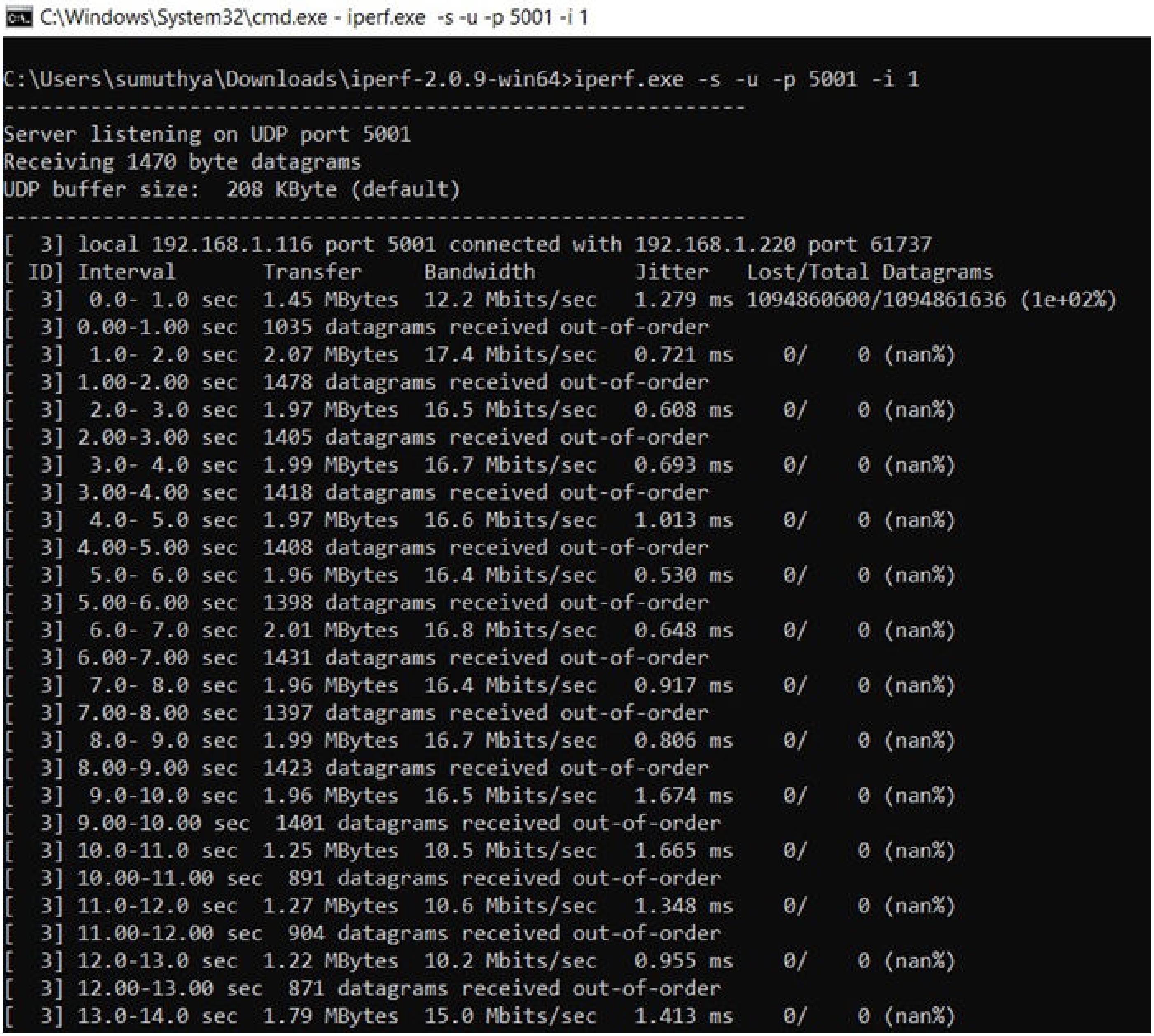
To measure UDP Tx throughput, configure the SiWN917 as a UDP client (2) and open the UDP server at the remote peer (4) using the following command:

```
C:\> iperf.exe -s -u -p <SERVER_PORT> -i 1

For example: C:\> iperf.exe -s -u -p 5001 -i 1
```

Note: Refer to wlan_throughput example readme for setup details and other information.

Below is the iPerf image for reference:



```
C:\Windows\System32\cmd.exe - iperf.exe -s -u -p 5001 -i 1

C:\Users\sumuthya\Downloads\iperf-2.0.9-win64>iperf.exe -s -u -p 5001 -i 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.116 port 5001 connected with 192.168.1.220 port 61737
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0- 1.0 sec  1.45 MBytes  12.2 Mbits/sec   1.279 ms 1094860600/1094861636 (1e+02%)
[  3] 0.00-1.00 sec  1035 datagrams received out-of-order
[  3]  1.0- 2.0 sec  2.07 MBytes  17.4 Mbits/sec   0.721 ms    0/    0 (nan%)
[  3] 1.00-2.00 sec  1478 datagrams received out-of-order
[  3]  2.0- 3.0 sec  1.97 MBytes  16.5 Mbits/sec   0.608 ms    0/    0 (nan%)
[  3] 2.00-3.00 sec  1405 datagrams received out-of-order
[  3]  3.0- 4.0 sec  1.99 MBytes  16.7 Mbits/sec   0.693 ms    0/    0 (nan%)
[  3] 3.00-4.00 sec  1418 datagrams received out-of-order
[  3]  4.0- 5.0 sec  1.97 MBytes  16.6 Mbits/sec   1.013 ms    0/    0 (nan%)
[  3] 4.00-5.00 sec  1408 datagrams received out-of-order
[  3]  5.0- 6.0 sec  1.96 MBytes  16.4 Mbits/sec   0.530 ms    0/    0 (nan%)
[  3] 5.00-6.00 sec  1398 datagrams received out-of-order
[  3]  6.0- 7.0 sec  2.01 MBytes  16.8 Mbits/sec   0.648 ms    0/    0 (nan%)
[  3] 6.00-7.00 sec  1431 datagrams received out-of-order
[  3]  7.0- 8.0 sec  1.96 MBytes  16.4 Mbits/sec   0.917 ms    0/    0 (nan%)
[  3] 7.00-8.00 sec  1397 datagrams received out-of-order
[  3]  8.0- 9.0 sec  1.99 MBytes  16.7 Mbits/sec   0.806 ms    0/    0 (nan%)
[  3] 8.00-9.00 sec  1423 datagrams received out-of-order
[  3]  9.0-10.0 sec  1.96 MBytes  16.5 Mbits/sec   1.674 ms    0/    0 (nan%)
[  3] 9.00-10.00 sec  1401 datagrams received out-of-order
[  3] 10.0-11.0 sec  1.25 MBytes  10.5 Mbits/sec   1.665 ms    0/    0 (nan%)
[  3] 10.00-11.00 sec  891 datagrams received out-of-order
[  3] 11.0-12.0 sec  1.27 MBytes  10.6 Mbits/sec   1.348 ms    0/    0 (nan%)
[  3] 11.00-12.00 sec  904 datagrams received out-of-order
[  3] 12.0-13.0 sec  1.22 MBytes  10.2 Mbits/sec   0.955 ms    0/    0 (nan%)
[  3] 12.00-13.00 sec  871 datagrams received out-of-order
[  3] 13.0-14.0 sec  1.79 MBytes  15.0 Mbits/sec   1.413 ms    0/    0 (nan%)
```

Figure 6.2. iPerf as UDP Server

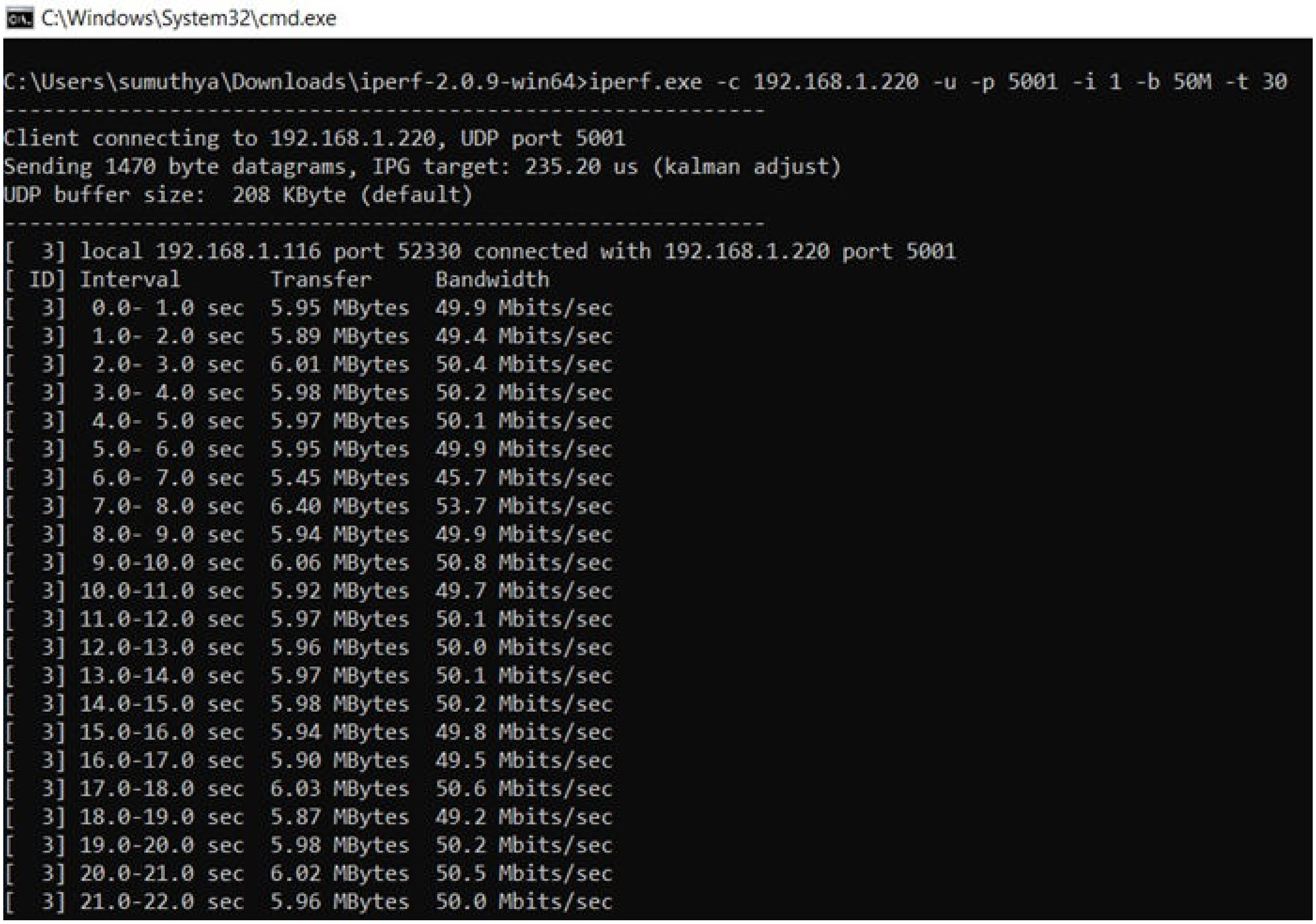
6.1.2 UDP Rx

To measure UDP Rx throughput, configure the SiWN917 as a UDP server (2) and open the UDP client at the remote peer (4) using the following command:

```
C:\> iperf.exe -c <Module_IP> -u -p <Module_Port> -i 1 -b <Bandwidth> -t <time interval in seconds>

For example: C:\> iperf.exe -c 192.168.1.220 -u -p 5001 -i 1 -b 50M -t 30
```

Below is the iPerf image for reference:



```
C:\Windows\System32\cmd.exe

C:\Users\sumuthya\Downloads\iperf-2.0.9-win64>iperf.exe -c 192.168.1.220 -u -p 5001 -i 1 -b 50M -t 30
------------------------------------------------------------
Client connecting to 192.168.1.220, UDP port 5001
Sending 1470 byte datagrams, IPG target: 235.20 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.116 port 52330 connected with 192.168.1.220 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  5.95 MBytes  49.9 Mbits/sec
[  3]  1.0- 2.0 sec  5.89 MBytes  49.4 Mbits/sec
[  3]  2.0- 3.0 sec  6.01 MBytes  50.4 Mbits/sec
[  3]  3.0- 4.0 sec  5.98 MBytes  50.2 Mbits/sec
[  3]  4.0- 5.0 sec  5.97 MBytes  50.1 Mbits/sec
[  3]  5.0- 6.0 sec  5.95 MBytes  49.9 Mbits/sec
[  3]  6.0- 7.0 sec  5.45 MBytes  45.7 Mbits/sec
[  3]  7.0- 8.0 sec  6.40 MBytes  53.7 Mbits/sec
[  3]  8.0- 9.0 sec  5.94 MBytes  49.9 Mbits/sec
[  3]  9.0-10.0 sec  6.06 MBytes  50.8 Mbits/sec
[  3] 10.0-11.0 sec  5.92 MBytes  49.7 Mbits/sec
[  3] 11.0-12.0 sec  5.97 MBytes  50.1 Mbits/sec
[  3] 12.0-13.0 sec  5.96 MBytes  50.0 Mbits/sec
[  3] 13.0-14.0 sec  5.97 MBytes  50.1 Mbits/sec
[  3] 14.0-15.0 sec  5.98 MBytes  50.2 Mbits/sec
[  3] 15.0-16.0 sec  5.94 MBytes  49.8 Mbits/sec
[  3] 16.0-17.0 sec  5.90 MBytes  49.5 Mbits/sec
[  3] 17.0-18.0 sec  6.03 MBytes  50.6 Mbits/sec
[  3] 18.0-19.0 sec  5.87 MBytes  49.2 Mbits/sec
[  3] 19.0-20.0 sec  5.98 MBytes  50.2 Mbits/sec
[  3] 20.0-21.0 sec  6.02 MBytes  50.5 Mbits/sec
[  3] 21.0-22.0 sec  5.96 MBytes  50.0 Mbits/sec
```

Figure 6.3. iPerf as UDP Server

6.2 Transmission Control Protocol (TCP)

TCP is connection-oriented, meaning once a connection has been established, data can be transmitted in two directions. TCP has built-in systems to check for errors and to guarantee data will be delivered in the order it was sent, making it the perfect protocol for transferring information like still images, data files, and web pages.

TCP throughput is the rate at which the data is successfully delivered over a TCP connection. It is an important metric to measure the quality of a network connection. TCP protocol integrates a mechanism that checks that all packets are correctly delivered. This mechanism is called acknowledgment, it consists of having the receiver transmit a specific packet or flag to the sender to confirm the proper reception of a packet.

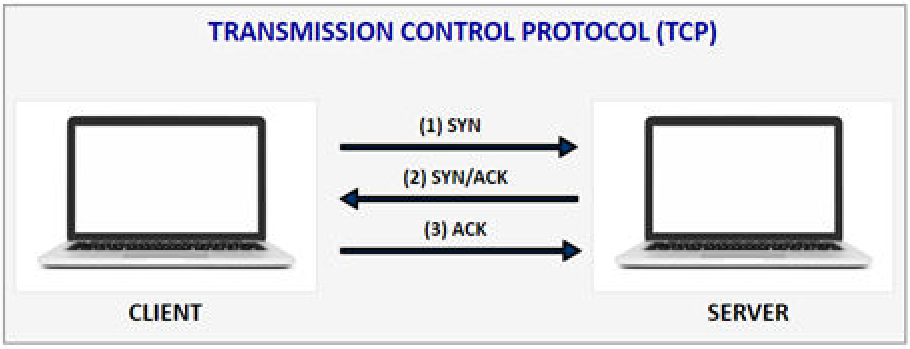


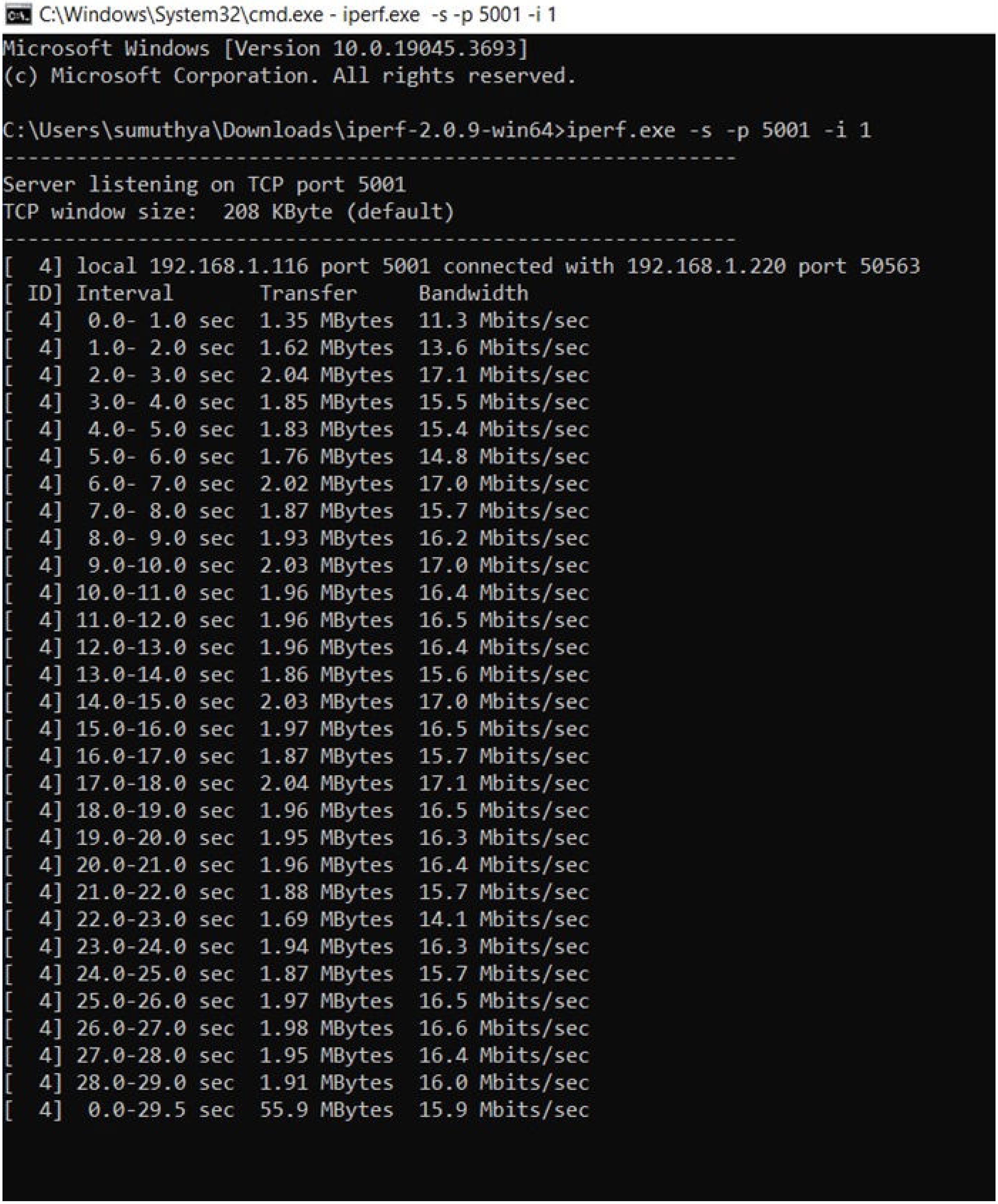
Figure 6.4. Transmission Control Protocol (TCP)

6.2.1 TCP Tx

To measure TCP Tx throughput, configure the SiWN917 as a TCP client (2) and open the TCP server at the remote peer (4) using the following command:

```
C:\> iperf.exe -s -p <SERVER_PORT> -i 1
For example: C:\> iperf.exe -s -p 5001 -i 1
```

Below is the iPerf image for reference:



```
C:\Windows\System32\cmd.exe - iperf.exe  -s -p 5001 -i 1
Microsoft Windows [Version 10.0.19045.3693]
(c) Microsoft Corporation. All rights reserved.

C:\Users\sumuthya\Downloads\iperf-2.0.9-win64>iperf.exe -s -p 5001 -i 1
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  208 KByte (default)
------------------------------------------------------------
[  4] local 192.168.1.116 port 5001 connected with 192.168.1.220 port 50563
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 1.0 sec  1.35 MBytes  11.3 Mbits/sec
[  4]  1.0- 2.0 sec  1.62 MBytes  13.6 Mbits/sec
[  4]  2.0- 3.0 sec  2.04 MBytes  17.1 Mbits/sec
[  4]  3.0- 4.0 sec  1.85 MBytes  15.5 Mbits/sec
[  4]  4.0- 5.0 sec  1.83 MBytes  15.4 Mbits/sec
[  4]  5.0- 6.0 sec  1.76 MBytes  14.8 Mbits/sec
[  4]  6.0- 7.0 sec  2.02 MBytes  17.0 Mbits/sec
[  4]  7.0- 8.0 sec  1.87 MBytes  15.7 Mbits/sec
[  4]  8.0- 9.0 sec  1.93 MBytes  16.2 Mbits/sec
[  4]  9.0-10.0 sec  2.03 MBytes  17.0 Mbits/sec
[  4] 10.0-11.0 sec  1.96 MBytes  16.4 Mbits/sec
[  4] 11.0-12.0 sec  1.96 MBytes  16.5 Mbits/sec
[  4] 12.0-13.0 sec  1.96 MBytes  16.4 Mbits/sec
[  4] 13.0-14.0 sec  1.86 MBytes  15.6 Mbits/sec
[  4] 14.0-15.0 sec  2.03 MBytes  17.0 Mbits/sec
[  4] 15.0-16.0 sec  1.97 MBytes  16.5 Mbits/sec
[  4] 16.0-17.0 sec  1.87 MBytes  15.7 Mbits/sec
[  4] 17.0-18.0 sec  2.04 MBytes  17.1 Mbits/sec
[  4] 18.0-19.0 sec  1.96 MBytes  16.5 Mbits/sec
[  4] 19.0-20.0 sec  1.95 MBytes  16.3 Mbits/sec
[  4] 20.0-21.0 sec  1.96 MBytes  16.4 Mbits/sec
[  4] 21.0-22.0 sec  1.88 MBytes  15.7 Mbits/sec
[  4] 22.0-23.0 sec  1.69 MBytes  14.1 Mbits/sec
[  4] 23.0-24.0 sec  1.94 MBytes  16.3 Mbits/sec
[  4] 24.0-25.0 sec  1.87 MBytes  15.7 Mbits/sec
[  4] 25.0-26.0 sec  1.97 MBytes  16.5 Mbits/sec
[  4] 26.0-27.0 sec  1.98 MBytes  16.6 Mbits/sec
[  4] 27.0-28.0 sec  1.95 MBytes  16.4 Mbits/sec
[  4] 28.0-29.0 sec  1.91 MBytes  16.0 Mbits/sec
[  4]  0.0-29.5 sec  55.9 MBytes  15.9 Mbits/sec
```

Figure 6.5. iPerf as TCP Server

6.2.2 TCP Rx

To measure TCP Rx throughput, configure the SiWN917 as a TCP server (2) and open the TCP client at the remote peer (4) using the following command:

```
C:\> iperf.exe -c <Module_IP> -p <Module_Port> -i 1 -t <time interval in sec>

For example: C:\> iperf.exe -c 192.168.1.220 -p 5001 -i 1 -t 30
```

Below is the iPerf image for reference:

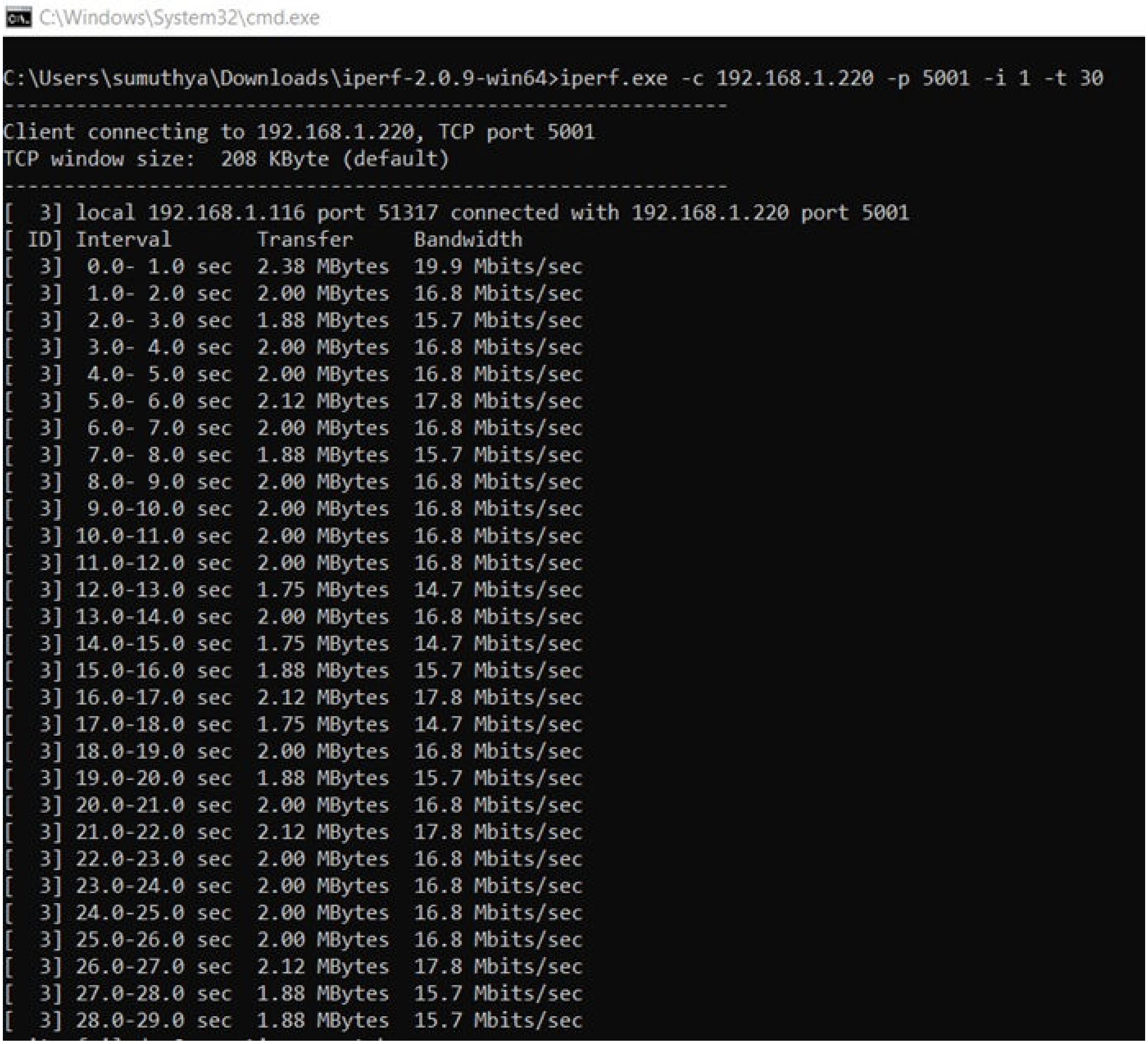


Figure 6.6. iPerf as TCP Client

6.3 Transport Layer Security (TLS)

SSL (Secure Sockets Layer) and its successor, Transport Layer Security, or TLS is a widely adopted security protocol designed to facilitate privacy and data security for communication over the Internet. A primary use case of TLS is encrypting the communication between web applications and servers. The three main components to what the TLS protocol accomplishes are encryption, authentication, and integrity.

In order to provide a high degree of privacy, TLS encrypts data that is transmitted across the web. This means that anyone who tries to intercept this data will only see a garbled mix of characters that is nearly impossible to decrypt. TLS then initiates an authentication process called a handshake between two communicating devices to ensure that both devices are really who they claim to be. And then it digitally signs data in order to provide data integrity, verifying that the data is not tampered with before reaching its intended recipient.

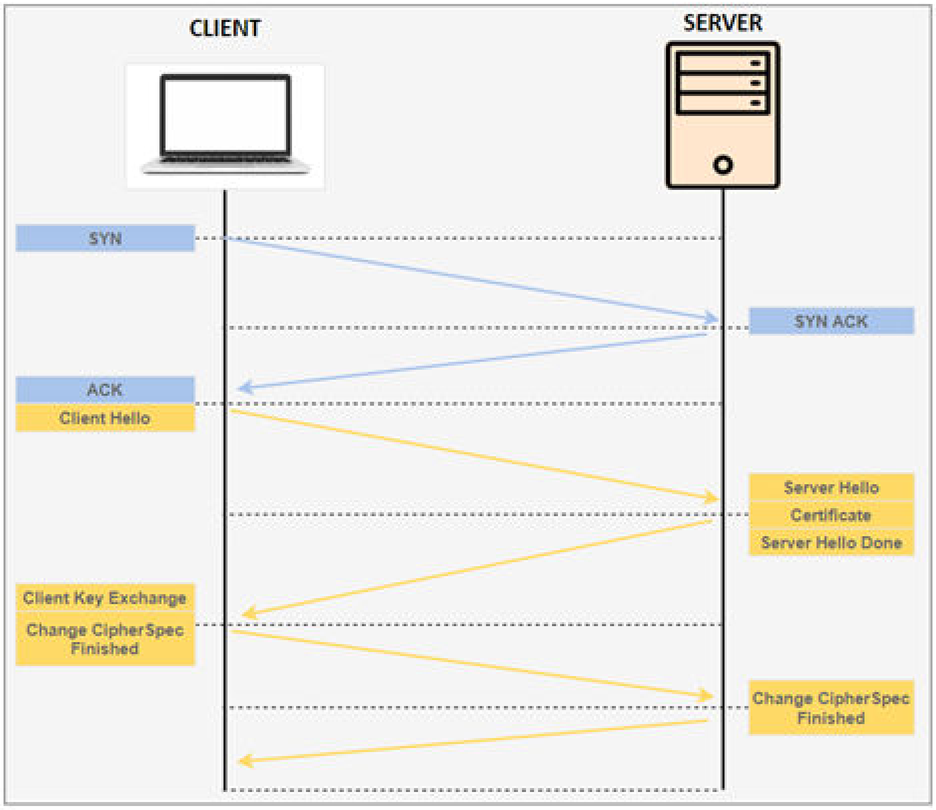


Figure 6.7. Transport Layer Security (TLS)

6.3.1 TLS Tx

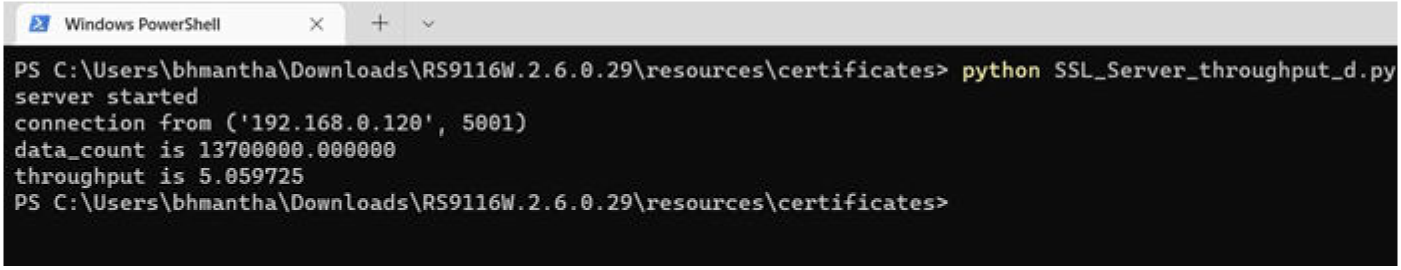


Figure 6.8. TLS Tx

6.3.2 TLS Rx

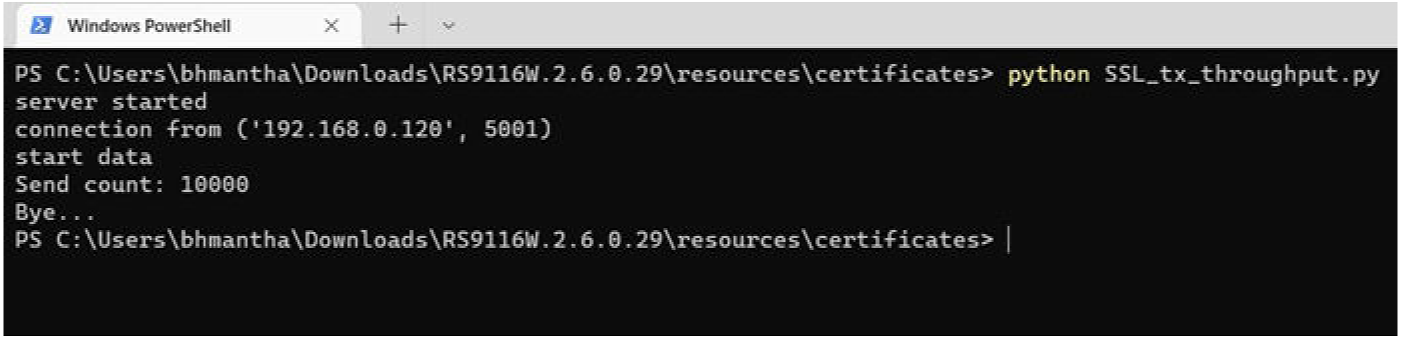


Figure 6.9. TLS Rx

Note: All the throughput numbers have been measured in an RF enclosure by running iPerf at the remote peer. Throughput numbers might vary depending on the environment and the wireless traffic on air. It would also differ based on the host interface speeds, host processor capabilities (CPU frequency, RAM, etc.), wireless medium, physical obstacles, distance, etc. Make sure the AP and Server running in PC are connected over ETHERNET cable as it is generally defined as the maximum bandwidth without any packet loss. The SPI clock configurations may be changed depending on the MCU. For configurations, refer to the data sheet of the respective MCU.

7. Wireshark I/O Graphs

The Wireshark I/O graph displays the traffic present in a capture file, which is measured in bytes per second. The x-axis represents the time in seconds, and the y-axis represents the Bytes per second.

Steps to Trace an I/O Graph:

1. Open the required Wireshark capture and apply the filter like TCP or UDP.
2. Now click on Statistics→ I/O Graph menu or toolbar item.

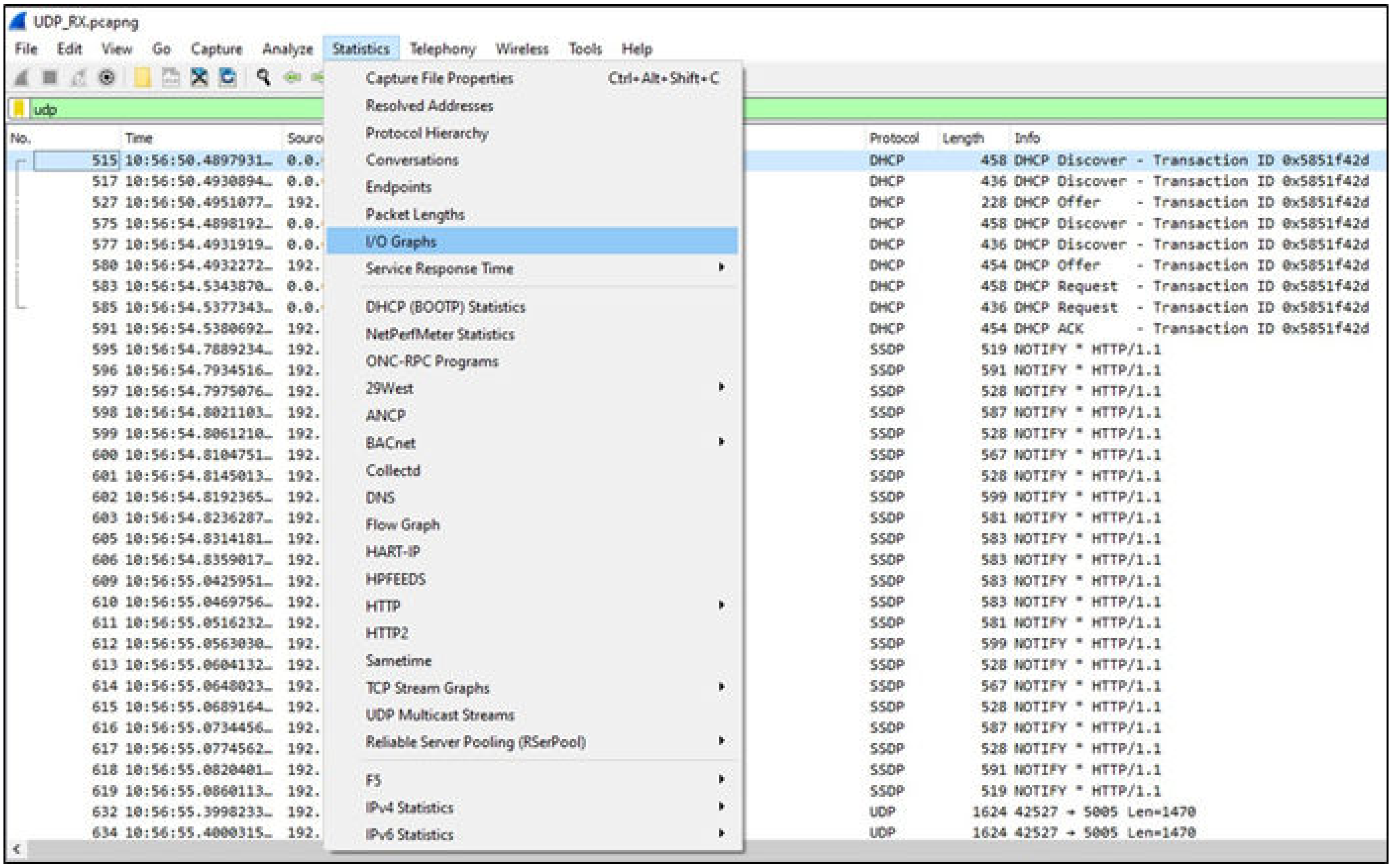


Figure 7.1. Wireshark I/O Graph

If UDP_Rx capture is considered, the following graph conveys that the transfer rate of the UDP packets is ~16 Mbps per second.

7.1 UDP Tx

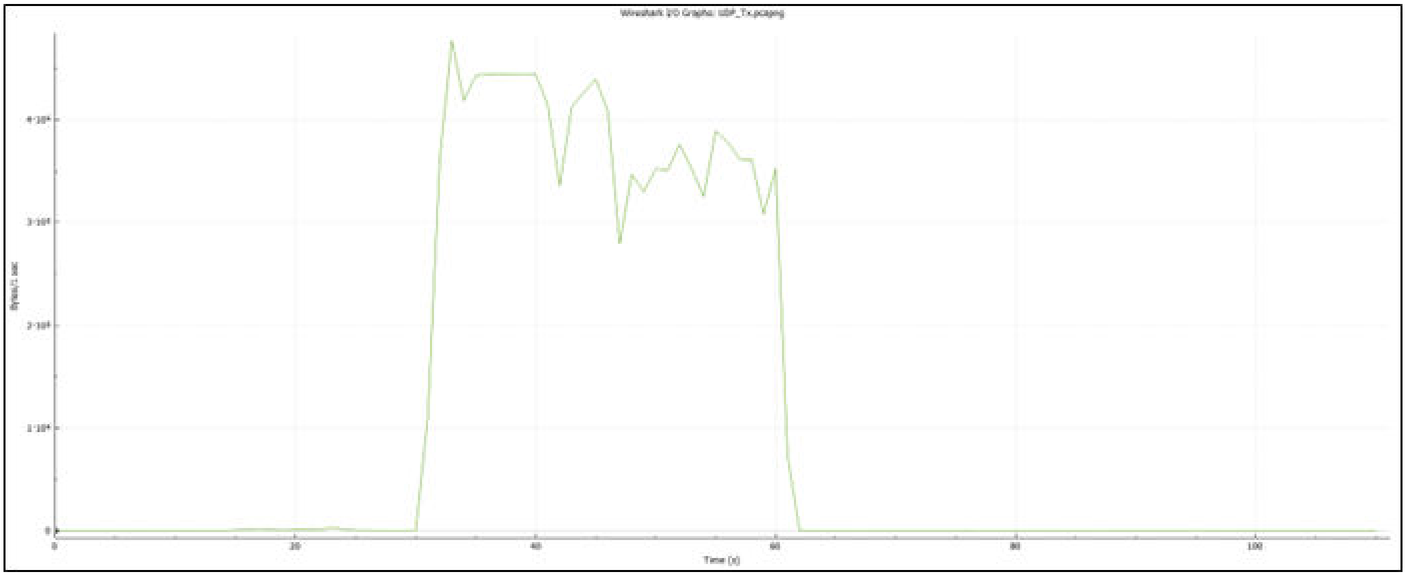


Figure 7.2. UDP Tx – Wireshark I/O Graph

7.2 UDP Rx

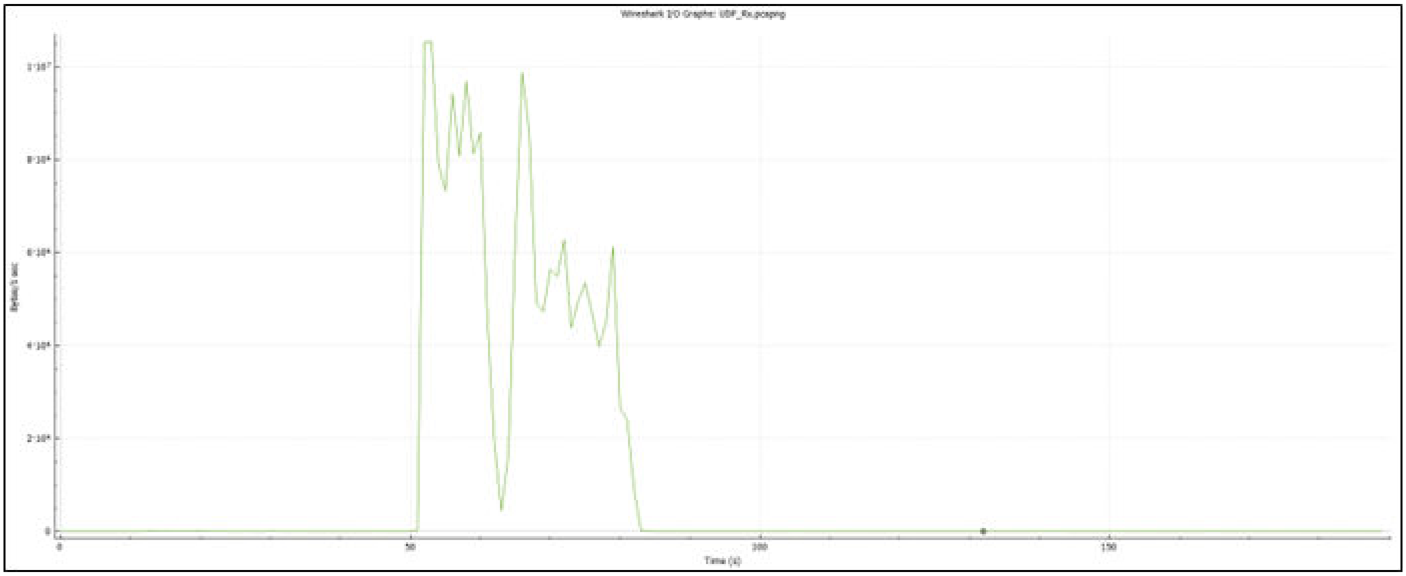


Figure 7.3. UDP Rx – Wireshark I/O Graph

7.3 TCP Tx

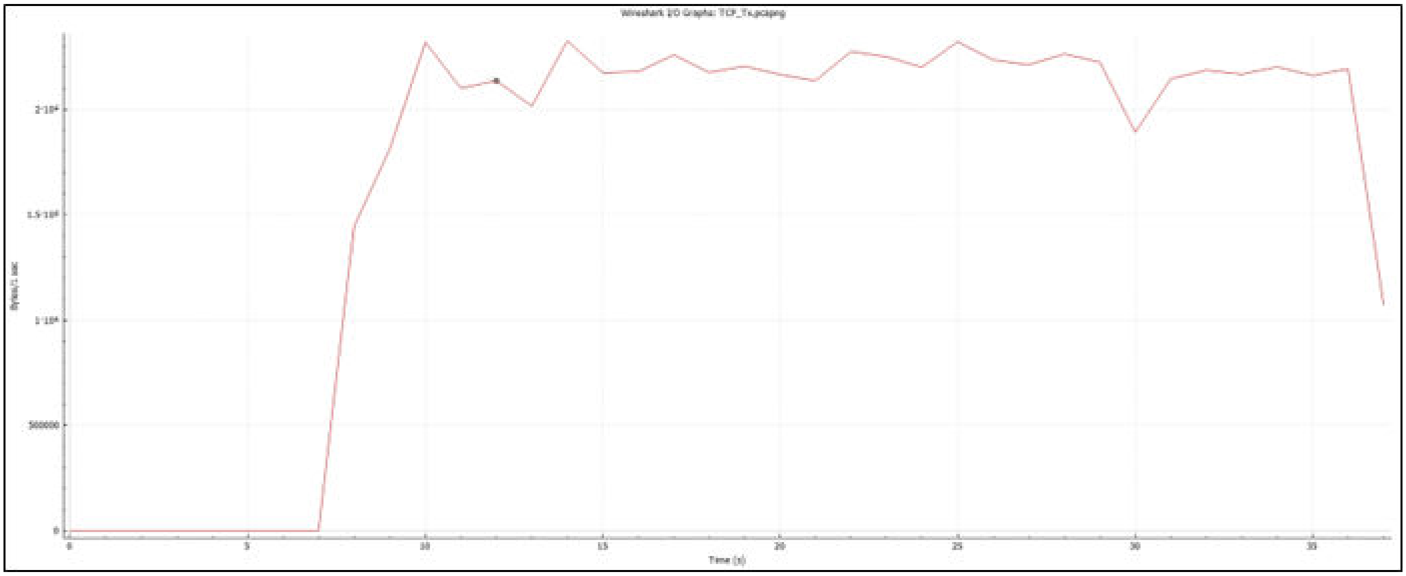


Figure 7.4. TCP Tx – Wireshark I/O Graph

7.4 TCP Rx

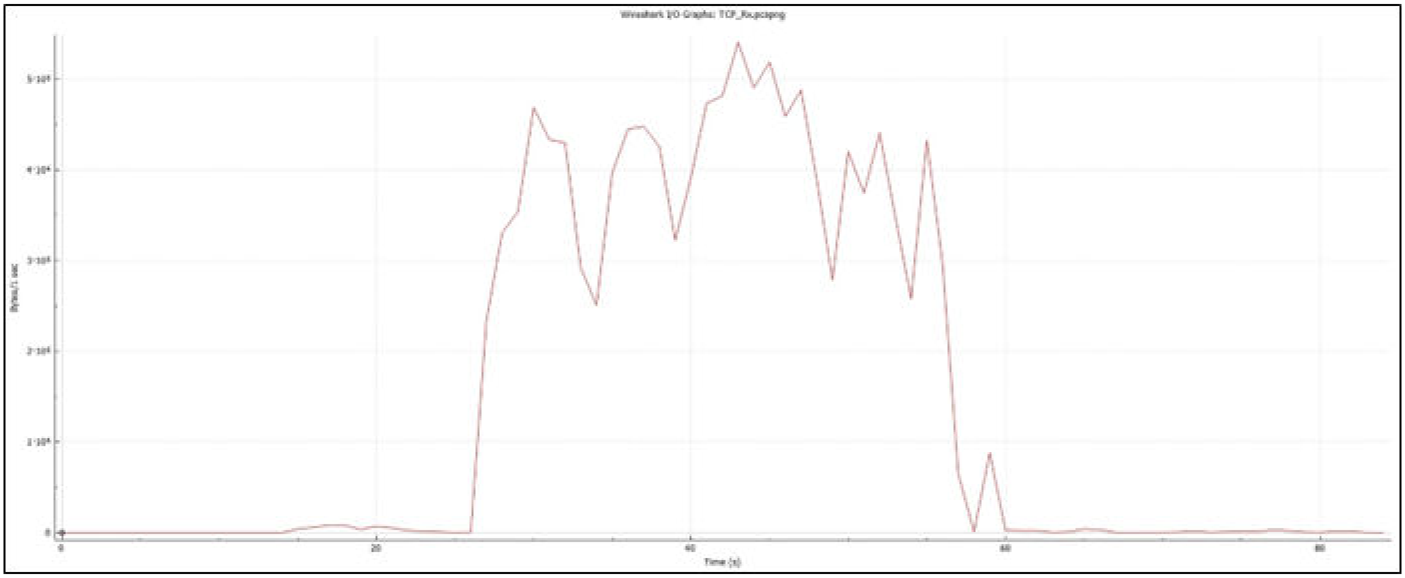


Figure 7.5. TCP Rx – Wireshark I/O Graph

7.5 TLS Tx

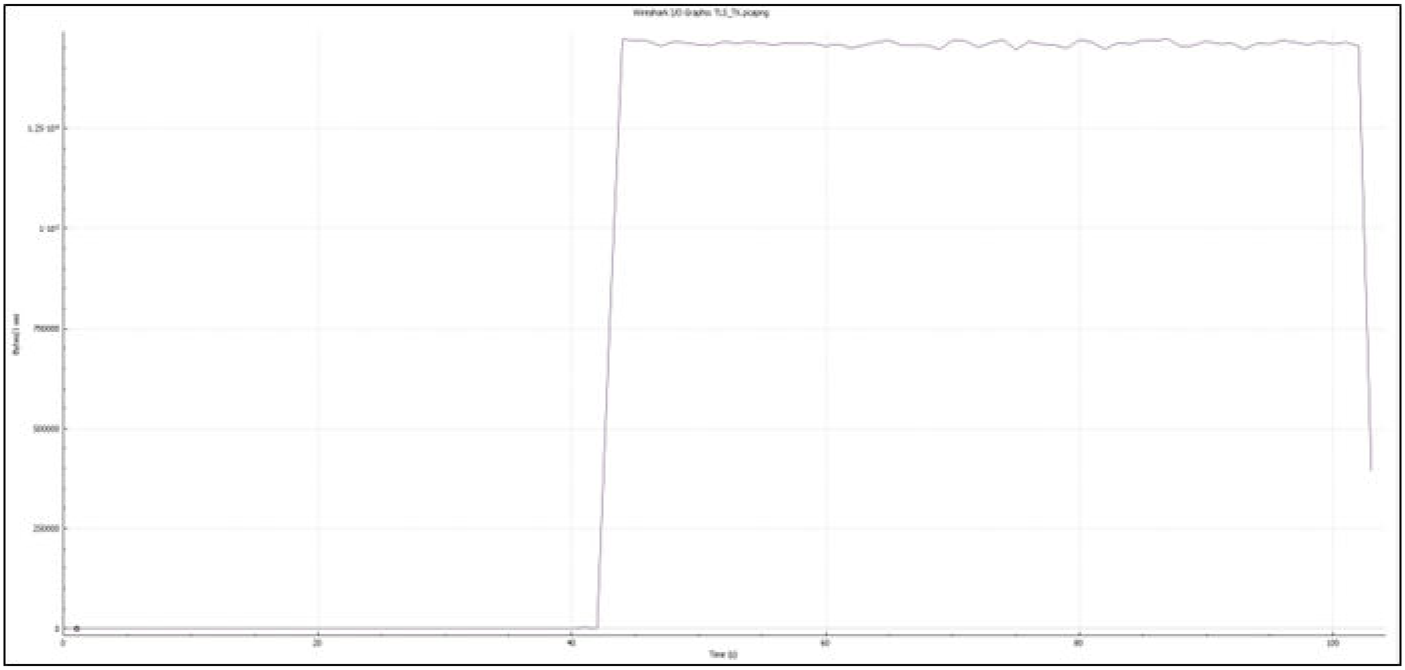


Figure 7.6. TLS Tx – Wireshark I/O Graph

7.6 TLS Rx

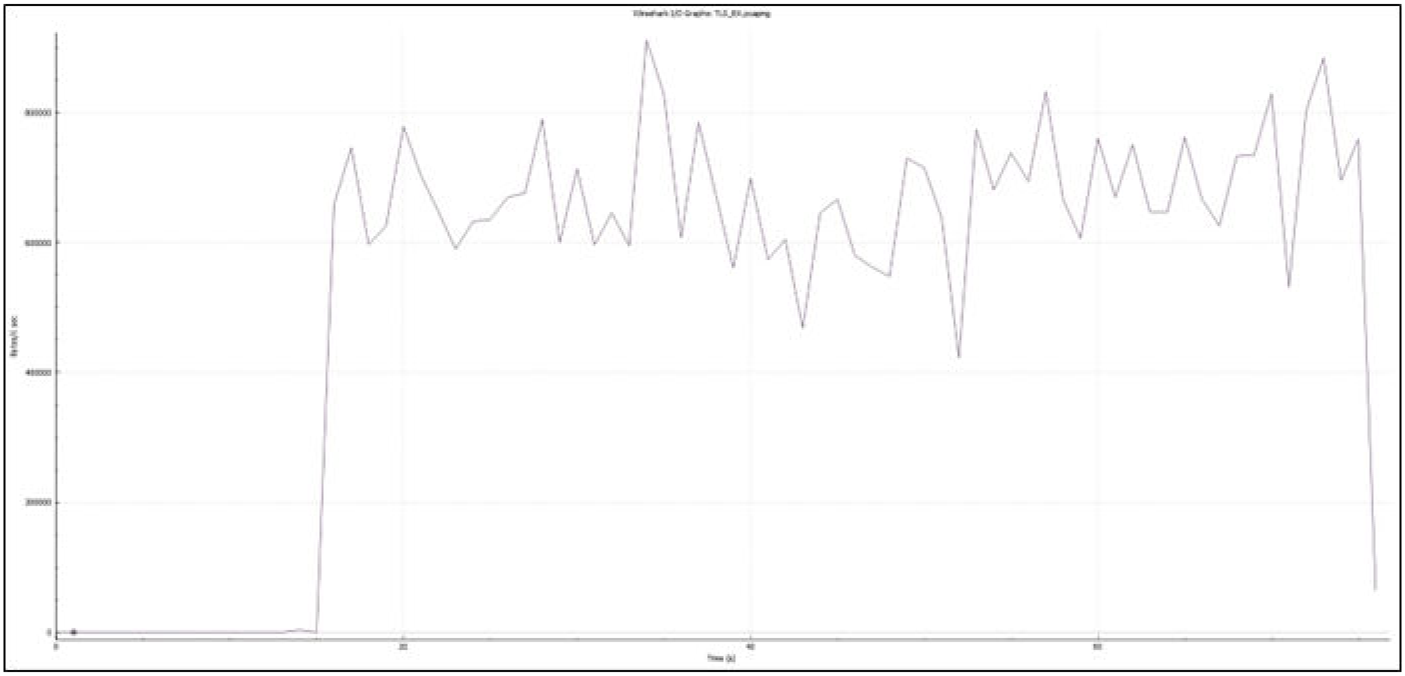


Figure 7.7. TLS Rx – Wireshark I/O Graph

All the throughput numbers mentioned below are with the SiWN917 NCP module using STM32F411RE as the host MCU and are for reference purposes only.

Note: Currently, we don't have BRD8045C Adapter boards for STM32 host MCU. Please reach out to sales team for more information on board availability.

S. No	Throughput Type	Test Tool	Throughput (Mbps)
1	UDP Tx	iPerf	16 Mbps
2	UDP Rx	iPerf	16 Mbps
3	TCP Tx	iPerf	16 Mbps
4	TCP Rx	iPerf	16 Mbps
5	TLS Tx	iPerf	15 Mbps
6	TLS Rx	iPerf	12 Mbps

Note: We also have a Throughput example with EFR32xG24 as the host MCU available in the SDK at \examples\featured \wlan_throughput

The throughput observed with EFR32xG24 host MCU are less compared to STM32F411RE host MCU.

8. Summary

By following the above procedure, configure the SiWN917 NCP module in station mode as a UDP/TCP/TLS server/client, connect to the iPerf server/client, and send/receive data for configured intervals. While the module is transmitting or receiving the data, the application prints the throughput numbers on the serial console.

9. Guidelines and Recommendations

- Throughput applications are recommended to be run while there is minimal traffic.
- Configure the TCP Rx window size (TCP_RX_WINDOW_SIZE_CAP) and TCP Rx window division factor (TCP_RX_WINDOW_DIV_FACTOR) to 44 to achieve high throughputs for TCP_Rx and TLS_Rx
- To get the maximum possible throughput make sure below are enabled. By default, these configurations are present in the throughput example.
  - Configure the SiWN917 in '672k memory configuration mode' and 'enable aggregation' via the opermode command.
  - Configure the SiWN917 SoC clock to 160MHz.
  - Enable the TCP Window division factor.
  - Enable the PLL mode in the feature frame.
- To achieve the maximum throughput values, the packet lengths for different networking protocols like TCP, UDP and, TLS are by default set to maximum in the throughput example application file. Throughput values would differ if the below buffer lengths were modified.
  - TCP_BUFF_SIZE 1460
  - UDP_BUFF_SIZE 1470
  - TLS_BUFF_SIZE 1370
- SiWN917 Multiuser-MIMO (Wi-Fi 6) helps in improving the overall network bandwidth. The sum of throughput across all devices when MU-MIMO is enabled will be approximately n (number of devices connected) times the sum of throughputs across all devices when MU-MIMO is disabled. The MU-MIMO is enabled by default in SiWN917 firmware, if the Access point is MU-MIMO supported, the user can see the throughput improvement.

10. References and Related Documents

- For other throughput examples, refer to the following examples in the WiSeConnect 3 SDK:
  - WLAN - refer to the WLAN Throughput example and the readme for detailed information about the throughput example and the execution steps
  - WLAN with IPv6 - refer to the WLAN Throughput IPv6 example
  - For BLE alone - refer to the BLE Throughput app example
- Refer to the WiSeConnect Documents for all SiWN917-related documents.
- Refer to the Release notes for latest Release Notes to see the enhancements done for the throughput application.

11. Troubleshooting

- For best throughput results, ensure that the AP and server running on the PC are connected over an Ethernet cable.
- Ensure that IP addresses are assigned to both client and server before running the iPerf test.
- Ensure that the station is connected to the AP.
- For running iPerf, the server should first begin listening, and then the client must send the data.

12. Revision History

Revision 1.0

February, 2024

- Initial Revision

IoT Portfolio
www.silabs.com/products

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com