AN1429: SiWx917 SoC Throughput

This document provides information about SiWG917 supported network protocols, socket configurations and the performance results along with the recommendations and guidelines to achieve the best performance.

Note:

This content may contain offensive terminology that is now obsolete. Silicon Labs is replacing these terms with inclusive language wherever possible.

For more information, see www.silabs.com/about-us/inclusive-lexicon-project.

KEY POINTS

- Setup Diagram
- Supported Protocols
- Socket Configuration
- Performance Results

Table of Contents

1. Setup

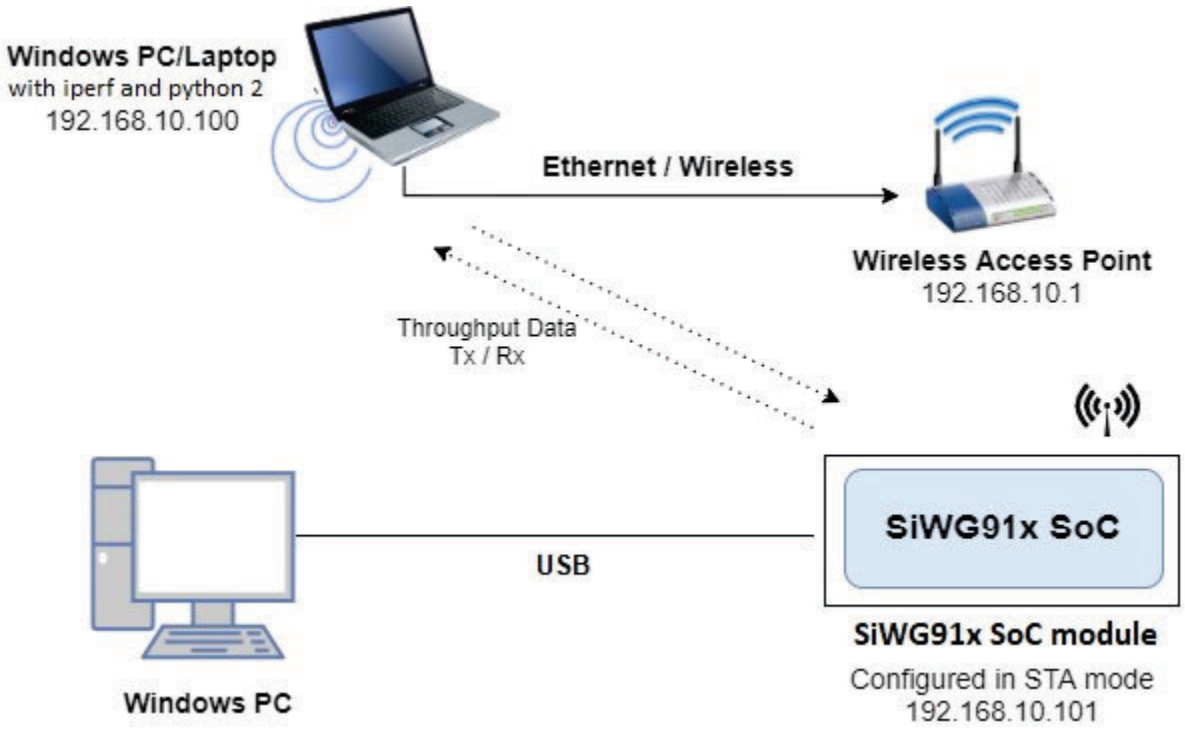

Figure 1.1. Throughput Setup Diagram

Note: It is recommended to use Ethernet between the Remote PC and Access Point for best throughput values.

Wireless connection creates the possibility of packet loss and retransmission due to network congestion, and the throughput might drop to 50% of what is observed with Ethernet connection.

2. Supported Protocols

SiWG917 supports the following protocols:

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

The following subsections explains these protocols in detail.

2.1 User Datagram Protocol (UDP)

UDP is a Transport Layer Protocol. It is a lightweight protocol with no support for handshaking, ordering, error recovery etc., UDP takes a datagram from Network Layer, attaches its header, and sends it to the user. So, it works fast.

Unlike TCP, UDP is an unreliable and connectionless protocol. So, there is no need to establish a connection prior to data transfer. The UDP helps to establish low-latency and loss-tolerating connections establish over the network.

For real-time services like computer gaming, voice or video communication, or live conferences, we need UDP. Since high performance is needed, UDP permits packets to be dropped instead of processing delayed packets. There is no error checking in UDP, so it also saves bandwidth. User Datagram Protocol (UDP) is more efficient in terms of both latency and bandwidth.

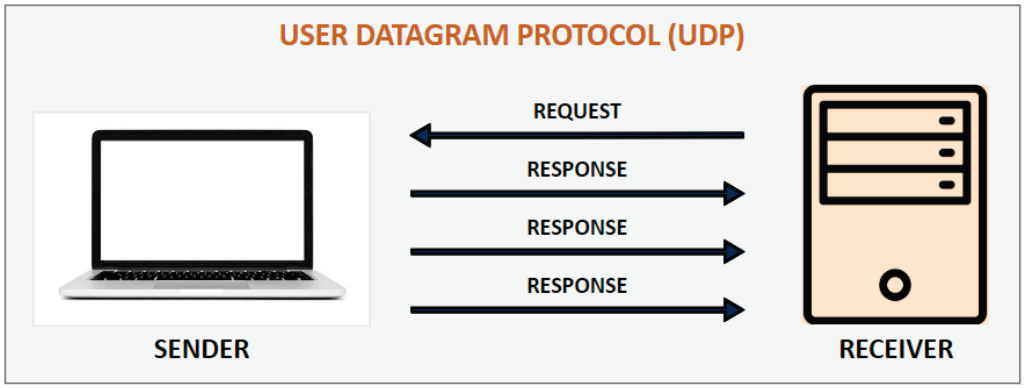

Figure 2.1. User Datagram Protocol (UDP)

UDP Tx:

To measure UDP Tx throughput, configure the SiWG917 as a UDP client and open UDP server at the remote peer using the following command:

```
C:\> iperf.exe -s -u -p <SERVER_PORT> -i 1
Example: C:\> iperf.exe -s -u -p 5000 -i 1
```

UDP Rx:

To measure UDP Rx throughput, configure the SiWG917 as a UDP server and open UDP client at the remote peer using the following command.

```
C:\> iperf.exe -c <Module_IP> -u -p <Module_Port> -i 1 -b <Bandwidth> -t <time interval in seconds>
Example: C:\> iperf.exe -c 192.168.50.91 -u -p 5005 -i 1 -b 40M -t 30
```

2.2 Transmission Control Protocol (TCP)

TCP is one of the main protocols of the Internet protocol suite. 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 protocol integrates a mechanism that checks that all packets are correctly delivered. This mechanism is called acknowledgment, and it consists of having the receiver transmit a specific packet or flag to the sender to confirm the proper reception of a packet.

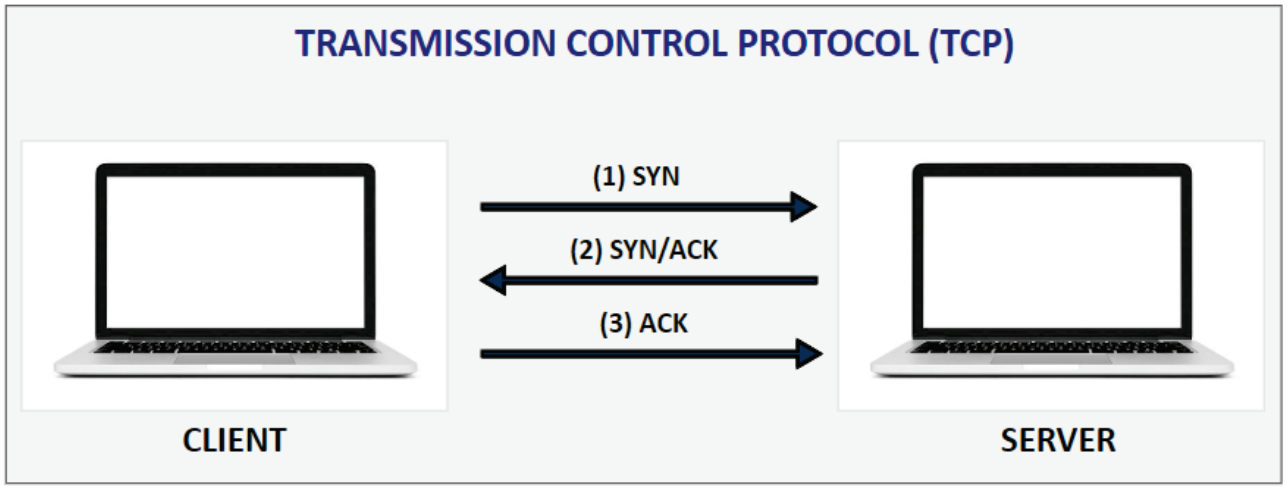

Figure 2.2. Transmission Control Protocol (TCP)

TCP Tx:

To measure TCP Tx throughput, configure the SiWG917 as a TCP client and open TCP server at the remote peer using the following command.

```
C:\> iperf.exe -s -p <SERVER_PORT> -i 1
Example: C:\> iperf.exe -s -p 5000 -i 1
```

TCP Rx:

To measure TCP Rx throughput, configure the SiWG917 as a TCP server and open TCP client at the remote peer using the following command.

```
C:\> iperf.exe -c <Module_IP> -p <module_PORT> -i 1 -t <time interval in sec>
Example: C:\> iperf.exe -c 192.168.50.91 -p 5005 -i 1 -t 30
```

2.3 Transport Layer Security (TLS)

Secure Sockets Layer (SSL) and its successor, Transport Layer Security (TLS) is a widely adopted security protocol designed to facilitate privacy and data security for communication over the Internet. The 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.

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 to provide data integrity, verifying that the data is not tampered with before reaching its intended recipient.

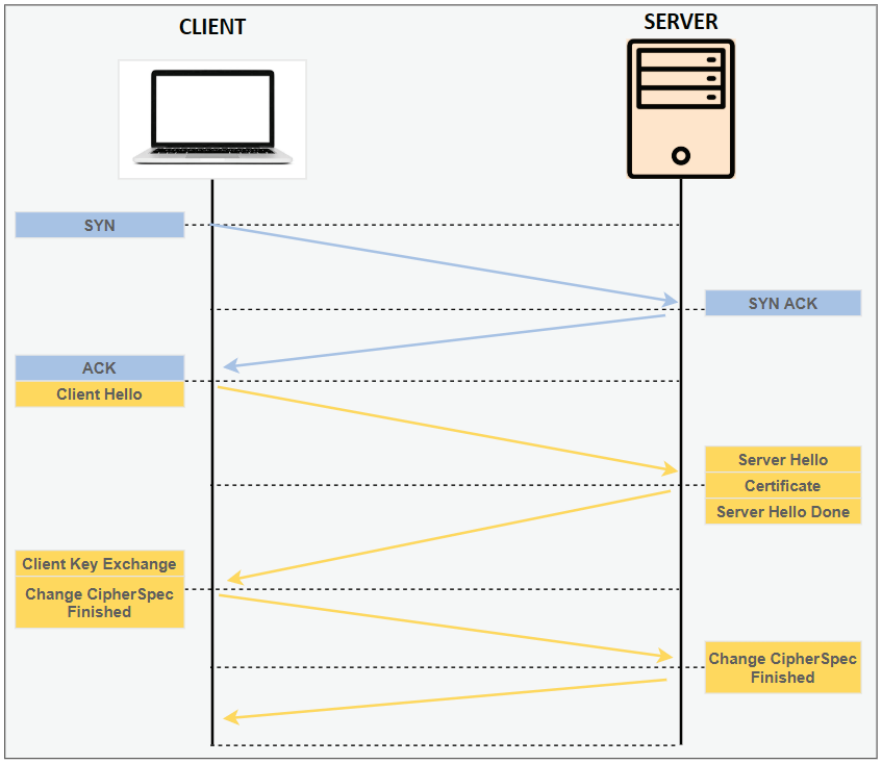

Figure 2.3. Transport Layer Security (TLS)

TLS Tx:

To measure TLS Tx throughput, configure the SiWG917 as a TLS client and open TLS server at the remote peer using the below command.

```
IPv4:
C:\SiWG917_Release\resources\scripts>python SSL_Server_throughput_d.py
IPv6:
C:\SiWG017_Release\resources\scripts>python SSL_Server_throughput_v6.py
```

TLS Rx:

To measure TLS Rx throughput, configure the SiWG917 as a TLS server and open TLS client at the remote peer using the below command.

```
IPv4:
C:\SiWG917_Release\resources\scripts>python SSL_tx_throughput_d.py
IPv6:
C:\SiWG917_Release\resources\scripts>python SSL_tx_throughput_v6.py
```

3. Network Stack

SiWG917 network stack modes are as shown below.

1. Offloaded mode
2. Hosted mode

By default, the TCP/IP stack runs on the TA which is the Offloaded mode. If you want to bypass it and run on the M4 which is the Hosted mode, you can refer to the LWIP TCP client example in the release.

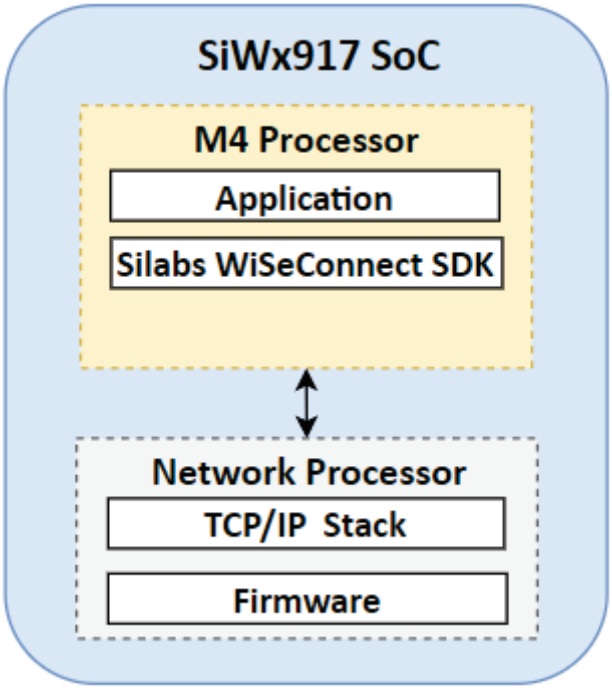

Figure 3.1. Offloaded Mode

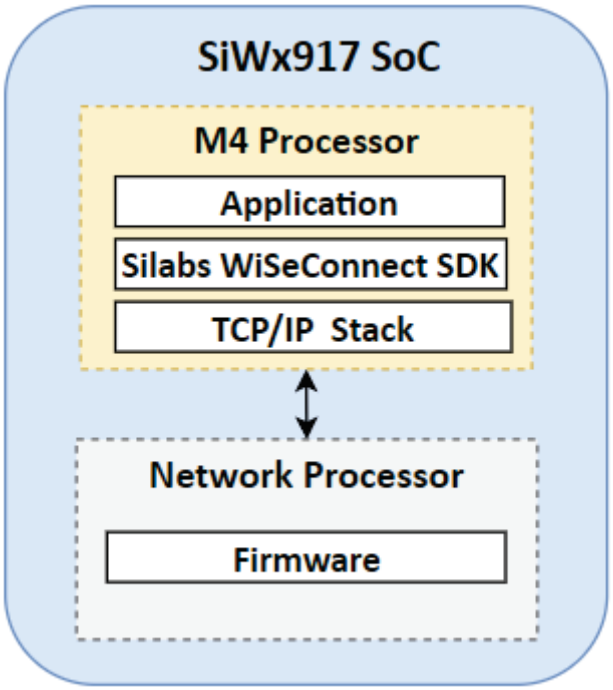

Figure 3.2. Hosted Mode

4. Socket Configuration

SiWG917 supports two modes of socket configurations:

1. BSD Socket Configuration
2. Asynchronous Socket Configuration

4.1 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 SiWG917 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.

4.2 Asynchronous Socket

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.

In 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 SiWG917 Asynchronous socket configuration, the data is sent or received asynchronously with a callback registered.

Note: The throughput results for asynchronous socket configuration are higher than that of BSD socket configuration.

5. Performance Results

The following table shows the test environment details.

Device	SiWG917 BRD4338A radio board
Memory Config	• NWP: 480K • M4:192K
AP Used	• ASUS AX3000 • WPA2-PSK
Environment	RF Shield room

The following table shows the 11ax throughput results for the above mentioned test environment with SiWG917 in station mode.

S.No	Protocol	Throughput (in Mbps)
		Asynchronous
1	UDP Tx	66
2	UDP Rx	62
3	TCP Tx	56
4	TCP Rx	48
5	TLS Tx	20
6	TLS Rx	13

The following table shows the throughput results with SiWG917 in access point mode.

S.No	Protocol	Throughput (in Mbps)
		Asynchronous
1	UDP Tx	22.7
2	UDP Rx	46
3	TCP Tx	24.7
4	TCP Rx	20.2

6. Guidelines and Recommendations

- Throughput applications are recommended to be run while there is minimal traffic.
- 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.
- 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.
- For higher performance, enable PLL_MODE (in sl_si91x_protocol_types.h file) in sl_si91x_feature_frame_request and also enable SL_SI91X_CUSTOM_FEAT_SOC_CLK_CONFIG_160MHZ (in .custom_feature_bit_map). By default, these configurations are present in the throughput example.
- 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
- SiWG917 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 SiWG917 firmware, if the Access point is MU0MIMO supported, the user can see the throughput improvement.

7. Troubleshooting

- For best throughput result, 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.
- Make sure that the station is connected to the AP, before giving the UDP/TCP/TLS related APIs.
- For running iPerf, the server should first begin listening, and then the client can send the data.
- If more than one socket needs to be opened at a time, then the total throughput shall be divided among all the sockets. For example, if four TCP sockets need to be opened, the SiWG91x is configured in station mode and acts as a TCP server to measure TCP Rx throughput. The throughput obtained on each socket would be ~3.5 Mbps, assuming the actual throughput is 14 Mbps in an ideal environment when a single socket is opened.

8. Future Scope

- We plan to include the performance setup and results for HTTP, WLAN + BLE (Coex), Access Point mode, Concurrent mode (Access Point +Station)
- Throughput results in 802.11 ax mode

9. Revision History

Revision 1.1

November 2024

- Updated: 5. Performance Results.
- Editorial changes throughout the document.

Revision 1.0

December 2023

- Initial release.

Smart. Connected. Energy-Friendly.

IoT Portfolio
www.silabs.com/products

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

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

www.silabs.com