QUIC10GC-IP Demo Instruction

Table of Contents

QUIC10GC-IP Demo Instruction

Rev1.00 2-Jul-2024

This document provides detailed instructions to demonstrate the use of the QUIC Client 10Gbps IP core (QUIC10GC-IP) on our reference design, referred to “QUIC10GC-IP Reference Design”, using the ZCU106 Evaluation Board. The QUIC10GC-IP is used as a medium to transfer data within a secure connection following the QUIC transport protocol version 1 standard (RFC9000). This process involves handling the TLS 1.3 handshake and dealing with data encryption and decryption.

The reference design uses the QUIC10GC-IP and manages the application layer of the IP. It is tailored to test the IP functionality, help users understand how to use the IP, and to offer flexibility for users in case they need to modify the design. There are two main applications demonstrated in this reference design: one is HTTP/3, as the application layer to streamline the HTTP data, and the other is a unique application protocol designed by an organization to use with their application.

This instruction will explain step-by-step how users can utilize the QUIC10GC-IP through our reference design for uploading and downloading data from two examples. aioquic is the first example, used to perform as a server using the HTTP/3, and the results are similar to those achieved by a web browser. Also, MsQuic is employed as a server to show the transfer performance using the unique application protocol.

Following our document guidelines, this document will describe how to set up the environment for the test, provide more details about our reference examples (aioquic and MsQuic), and instruct and show the results of the test, respectively.

1 Environment Setup

To run the QUIC10GC-IP demo, please prepare following test environment.

1) FPGA development boards (ZCU106 board).
2) Test PC with 10 Gigabit Ethernet or connecting with 10 Gigabit Ethernet card.
3) 10 Gb Ethernet cable:
   a) 10 Gb SFP+ Passive Direct Attach Cable (DAC) which has 1-m or less length
   b) 10 Gb SFP+ Active Optical Cable (AOC)
   c) 2x10 Gb SFP+ transceiver (10G BASE-R) with optical cable (LC to LC, Multimode)
4) Micro USB cable for JTAG connection connecting between ZCU106 board and Test PC.
5) Micro USB cable for UART connection connecting between ZCU106 board and Test PC.
6) Vivado tool for programming FPGA installed on Test PC.
7) Serial console software such as TeraTerm installed on PC. The setting on the console is Baudrate=115200, Data=8-bit, Non-parity and Stop=1.
8) Batch file named “QUIC10GCIPTest.bat” and its dependency files (To download these files, please visit our website at www.design-gateway.com).

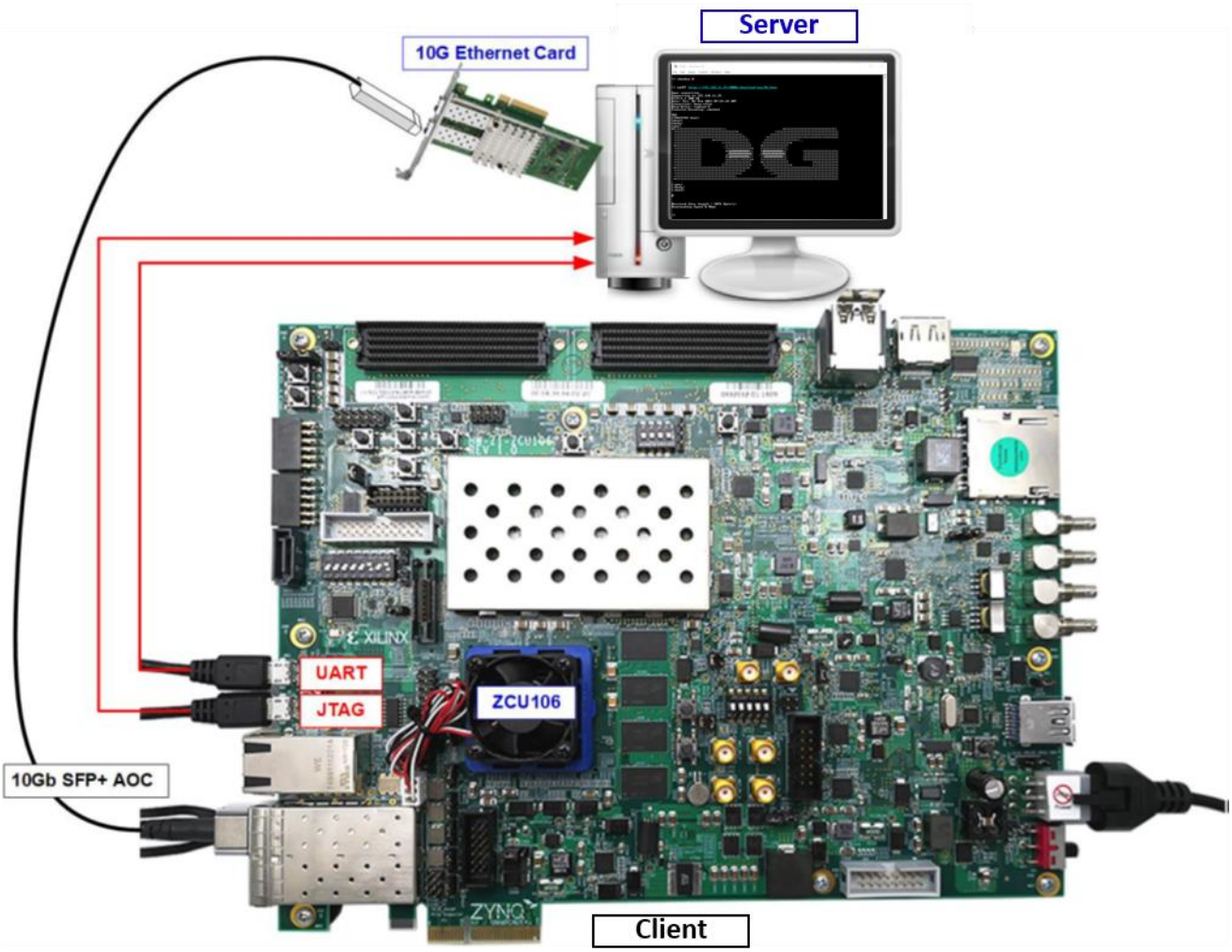

Figure 1-1 QUIC10GCIP demo environment on ZCU106 board

2 PC Setup

Before running demo, please check the network setting on PC. The example of setting 10 Gb Ethernet card is described as follows.

2.1 IP setting

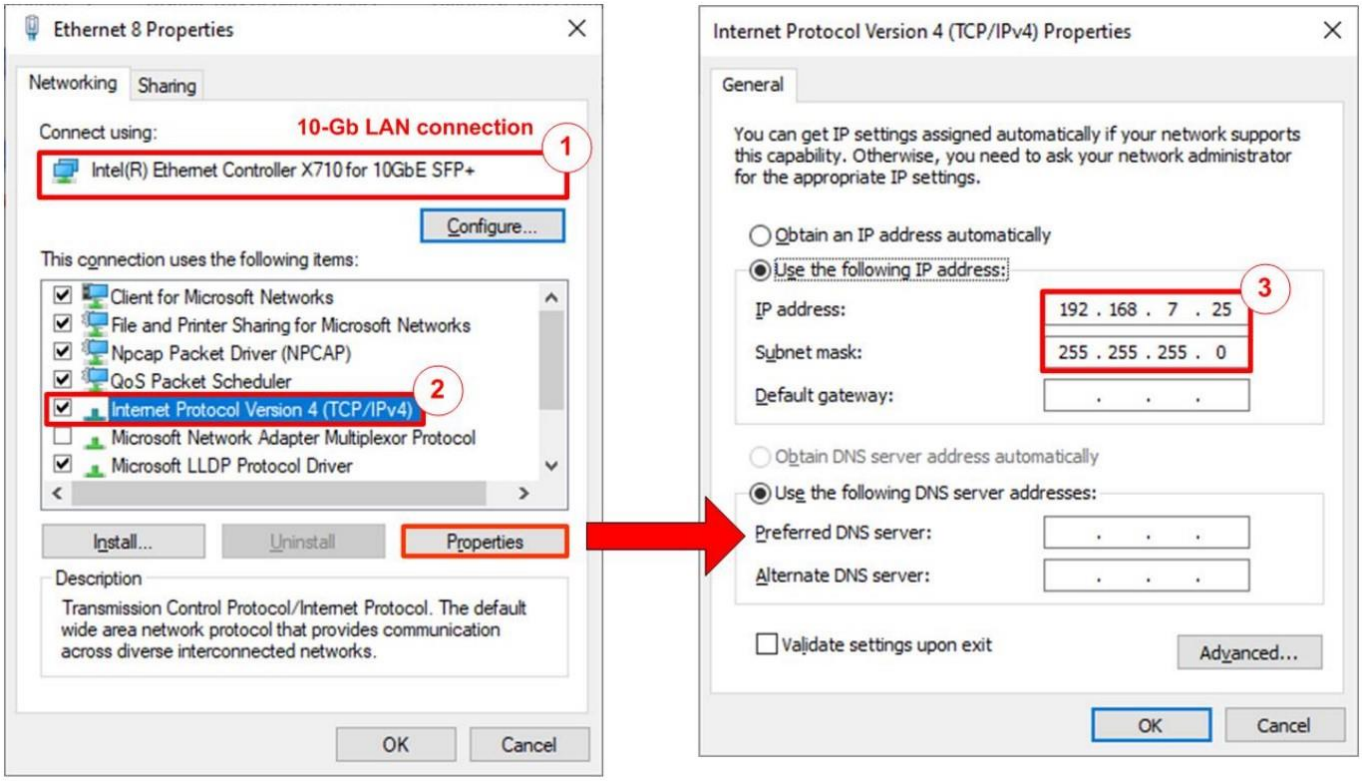

Figure 2-1 Setting IP address for PC

1) Open Local Area Connection Properties of 10 Gb connection, as shown in the left window of Figure 2-1.
2) Select “TCP/IPv4” and then click Properties.
3) Set IP address = 192.168.7.25 and Subnet mask = 255.255.255.0, as shown in the right window of Figure 2-1.

2.2 Speed and duplex settings

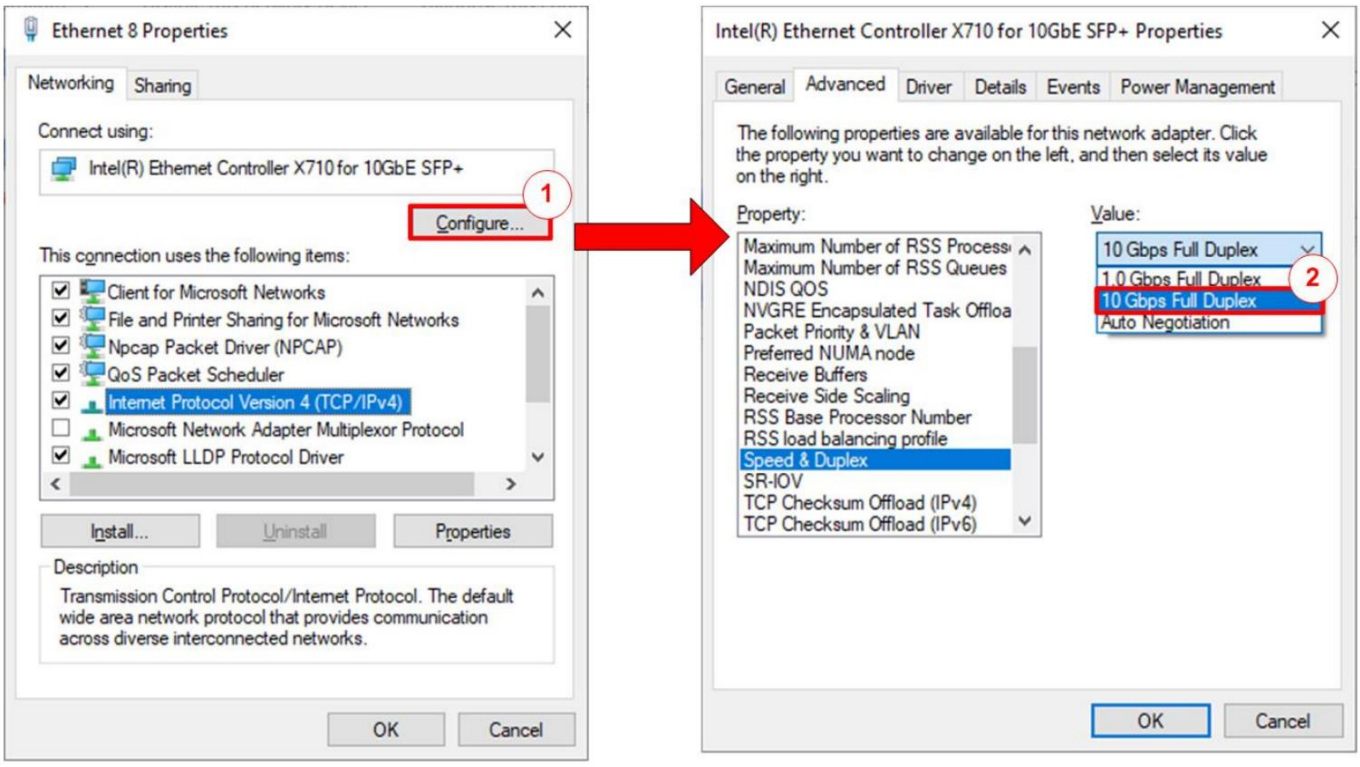

Figure 2-2 Set Link Speed = 10 Gbps

1) On Local Area Connection Properties window, click “Configure”, as shown in Figure 2-2.
2) On Advanced Tab, select “Speed and Duplex”. Set the value to “10 Gbps Full Duplex” for running 10 Gigabit transfer test, as shown in Figure 2-2.

2.3 Network properties settings

Some of network parameter settings may affect network performance. The example of network properties setting is as follows.

1) On “Interrupt Moderation” window, select “Disabled” to disable interrupt moderation which would minimize the latency during transferring data, as shown in Figure 2-3.

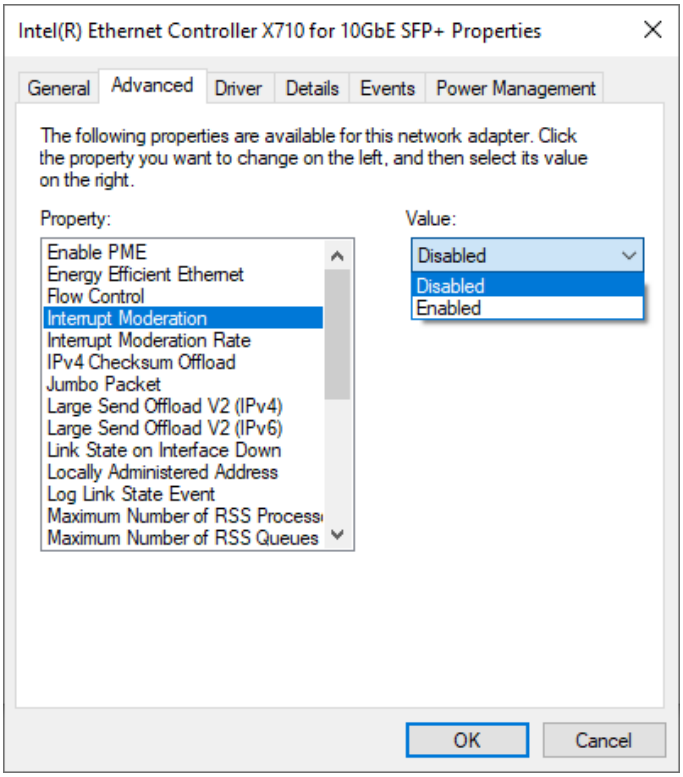

Figure 2-3 Interrupt Moderation

2) On “Interrupt Moderation Rate” window, set the value to “OFF”, as shown in Figure 2-4.

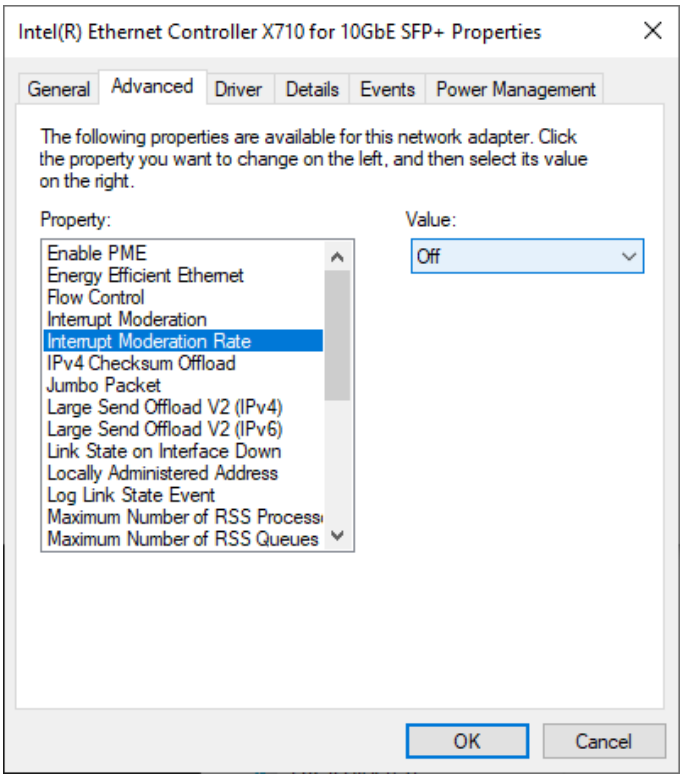

Figure 2-4 Interrupt Moderation Rate

3) On “Jumbo packet” window, set the value to “9014 Bytes”, as shown in Figure 2-5.

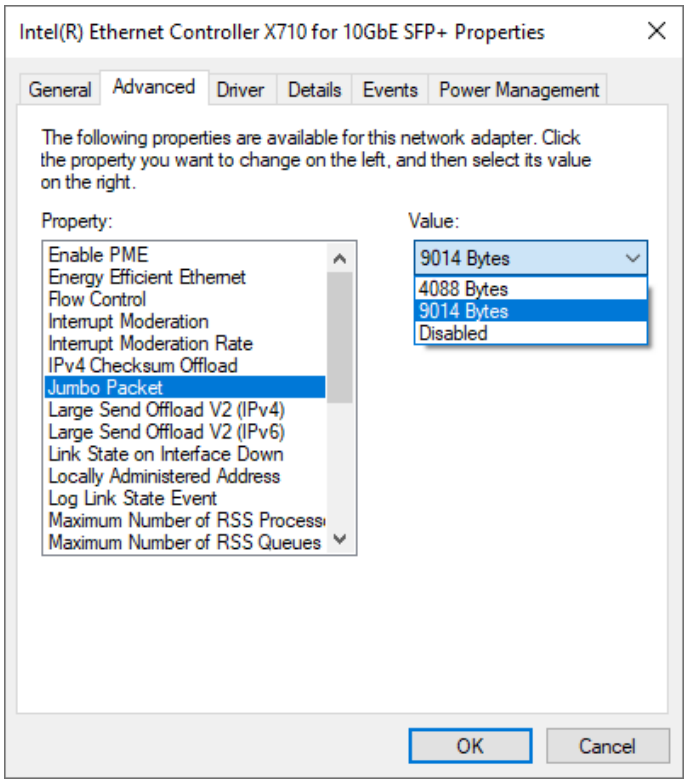

Figure 2-5 Jumbo packet

4) On “Receive Buffers” window, set the value to the maximum value, as shown in Figure 2-6.

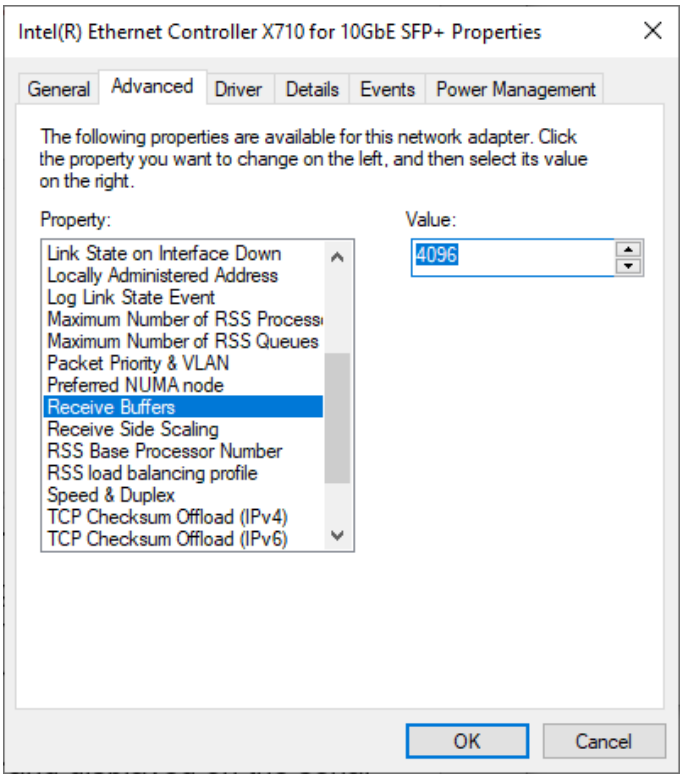

Figure 2-6 Receive Buffers

5) On “Transmit Buffers” window, set the value to the maximum value, as shown in Figure 2-7.

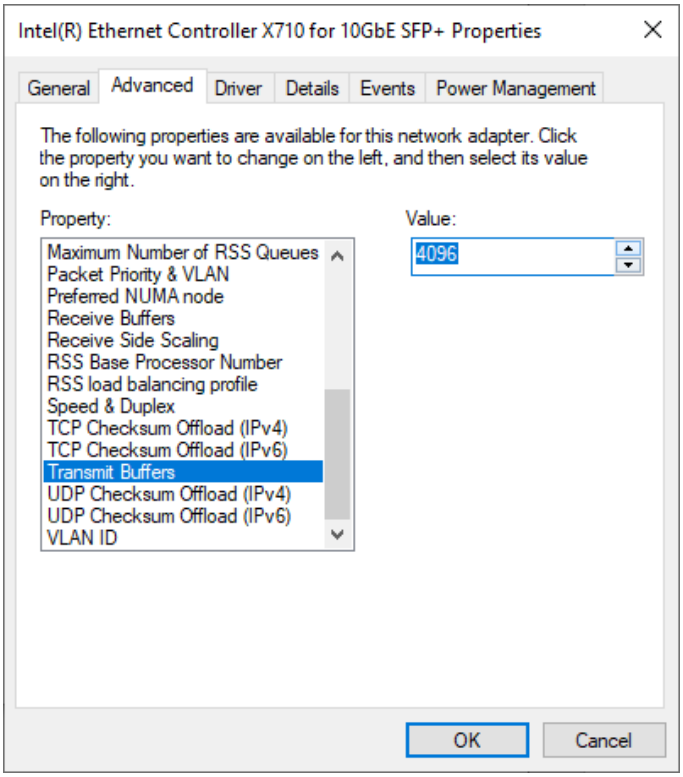

Figure 2-7 Transmit buffers

3 aioquic server

aioquic is an open-source implementation of QUIC and HTTP/3 in Python, which can be found using this link: github.com/aiortc/aioquic. However, this development requires some changes to run our demonstration. The following points highlight where the modifications are made.

1) Replaced 'examples/templates/index.html', which is the index html of aioquic, with Design Gateway's html file.
2) Added Design Gateway's key and certificate files, named tls_key.pem and tls_cert.pem, respectively, in 'tests/' directory.
3) Added an html file, called 'tests/httpEcho.html', to show one functionality of the aioquic.

This reference uses aioquic version 1.0.0, which we are grateful for the development and publication, and all of these modifications are included in our aioquic folk in this repository github.com/design-gateway/aioquic. If you have any questions regarding their core, please kindly direct them to the aioquic development team.

3.1 Run aioquic

'examples/http3_server.py' is used as an example to run a QUIC demo server. To run this server, certain parameters are required, all of which you can find using the help command, as shown in Figure 3-1. This information shows the available options and usage instructions for the aioquic HTTP/3 server, allowing users to configure parameters such as the TLS certificate, private key, host address, port, and other settings relevant to the QUIC server operation.

```
D:\Chromium\aioquic>py examples/http3_server.py -h
usage: http3_server.py [-h] -c CERTIFICATE [--congestion-control-algorithm CONGESTION_CONTROL_ALGORITHM] [--host HOST]
                       [--port PORT] [-k PRIVATE_KEY] [-l SECRETS_LOG] [--max-data MAX_DATA]
                       [--max-stream-data MAX_STREAM_DATA] [--max-datagram-size MAX_DATAGRAM_SIZE] [-q QUIC_LOG]
                       [--retry] [-v]
                       [app]

QUIC server

positional arguments:
  app                   the ASGI application as <module>:<attribute>

options:
  -h, --help            show this help message and exit
  -c CERTIFICATE, --certificate CERTIFICATE
                        load the TLS certificate from the specified file
  --congestion-control-algorithm CONGESTION_CONTROL_ALGORITHM
                        use the specified congestion control algorithm
  --host HOST           listen on the specified address (defaults to ::)
  --port PORT           listen on the specified port (defaults to 4433)
  -k PRIVATE_KEY, --private-key PRIVATE_KEY
                        load the TLS private key from the specified file
  -l SECRETS_LOG, --secrets-log SECRETS_LOG
                        log secrets to a file, for use with Wireshark
  --max-data MAX_DATA   connection-wide flow control limit (default: 1048576)
  --max-stream-data MAX_STREAM_DATA
                        per-stream flow control limit (default: 1048576)
  --max-datagram-size MAX_DATAGRAM_SIZE
                        maximum datagram size to send, excluding UDP or IP overhead
  -q QUIC_LOG, --quic-log QUIC_LOG
                        log QUIC events to QLOG files in the specified directory
  --retry               send a retry for new connections
  -v, --verbose         increase logging verbosity
```

Figure 3-1 aioquic console with the help command

As depicted in, an aioquic server is operated by running the python file on the terminal with certain options. This example uses the provided key and certificate files, binds to an existed address on the machine, and logs the secrets in a text file.

```
D:\Chromium\aioquic>py examples/http3_server.py --host 192.168.7.25 --certificate tests/tls_cert.pem --private-key tests
/tls_key.pem -l C:\Users\tan\sslkey\sslkeylog.txt
2024-05-29 10:31:06,329 INFO quic [37cc129c68e544ac] Duplicate CRYPTO data received for epoch Epoch.INITIAL
2024-05-29 10:31:06,330 INFO quic [37cc129c68e544ac] Duplicate CRYPTO data received for epoch Epoch.INITIAL
2024-05-29 10:31:06,330 INFO quic [37cc129c68e544ac] Duplicate CRYPTO data received for epoch Epoch.INITIAL
2024-05-29 10:31:06,331 INFO quic [37cc129c68e544ac] Duplicate CRYPTO data received for epoch Epoch.INITIAL
2024-05-29 10:31:06,331 INFO quic [37cc129c68e544ac] Duplicate CRYPTO data received for epoch Epoch.INITIAL
2024-05-29 10:31:06,344 INFO quic [37cc129c68e544ac] ALPN negotiated protocol h3
2024-05-29 10:31:06,350 INFO quic [37cc129c68e544ac] HTTP request GET /
2024-05-29 10:31:06,396 INFO quic [37cc129c68e544ac] HTTP request GET /style.css
2024-05-29 10:31:06,523 INFO quic [37cc129c68e544ac] HTTP request GET /favicon.ico
```

Figure 3-2 Example of running the aioquic HTTP/3 server

3.2 Run Chrome web browser

A typical web browser can be used to communicate with the aioquic server. The process involves using the GET method to download data from the server and using the POST method to upload data to the server. To access the demo server running on the local machine, launch Chromium or Chrome with the following command:

<path to chrome.exe> --enable-experimental-web-platform-features \
--ignore-certificate-errors-spki-list=<SPKI> \
--origin-to-force-quic-on=<server ip:server port> \
url path

```
C:\Users\tan>"c:\Program Files\Google\Chrome\Application\chrome.exe" --enable-experimental-web-platform-features --ignor
e-certificate-errors-spki-list=2kaKWhS2v9++0KSEDx5JfwzGs6QM3cBv0R90ZTM/GeI= --origin-to-force-quic-on=192.168.7.25:4433
https://192.168.7.25:4433/
```

Figure 3-3 Example command line to run Chrome with aioquic server

The RSA certificate used in this demonstration is self-signed, meaning it was not issued by a certification authority (CA). When attempting to access the server with a self-signed certificate, the web browser may generate a certificate unknown error and terminate the connection. To bypass certificate errors, users can run the Chrome browser from the command prompt with the '--ignore-certificate-errors-spki-list' flag, specifying the certificate's SPKI (Subject Public Key Info). This allows Chrome/Chromium to accept the self-signed certificate as valid. Users can generate the SPKI with the following command:

```
openssl x509 -noout -pubkey -in tls_cert.pem |^
openssl pkey -pubin -outform der |^
openssl dgst -sha256 -binary |^
openssl base64
```

```
D:\Chromium\aioquic\tests>openssl x509 -noout -pubkey -in tls_cert.pem | openssl pkey -pubin -outform der | openssl dgst
 -sha256 -binary | openssl base64
2kaKWhS2v9++0KSEDx5JfwzGs6QM3cBv0R90ZTM/GeI=
```

Figure 3-4 Example command line to calculate public key hash from certificate

When launching Chrome with the '--ignore-certificate-errors-spki-list' flag, users may encounter a message indicating that Chrome is running with an unsupported command-line flag. The flag '--ignore-certificate-errors-spki-list' is used to bypass SSL/TLS certificate errors by specifying a list of SPKI hashes. This can be useful for testing purposes but is not recommended for regular use due to potential security and stability risks.

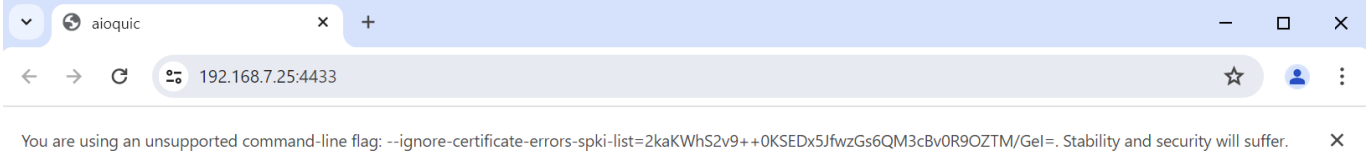

Figure 3-5 Example of encountering the --ignore-certificate-errors-spki-list flag

Remark: Our tested web browser is Google Chrome version 125.0.6422.113.

3.3 Download with method GET

The aioquic demo example supports HTTP/3, allowing users to download data from the aioquic server using a web browser via the GET method. To download data, the URL required as an input must follow this format:

https://ip:port/length

Where ip represents server's IP address in dot-decimal notation.

port represents server's port number.

length represents data length in byte (or leave blank to get the homepage).

For example, if the server's IP address is 192.168.7.25 and the port number is 4433, the URL must be https://192.168.7.25:4433/ to establish a secure connection and display the aioquic homepage in the web browser, as illustrated in Figure 3-7.

```
2024-05-27 16:00:03,396 INFO quic [95110ffd4223fd08] ALPN negotiated protocol h3
2024-05-27 16:00:03,410 INFO quic [95110ffd4223fd08] HTTP request GET /
2024-05-27 16:00:03,865 INFO quic [95110ffd4223fd08] HTTP request GET /favicon.ico
```

Figure 3-6 aioquic console when the client downloads the homepage

Figure 3-7 Download the index html using web browser

To download data with a specific length, user adds a size in byte unit at the end of the URL. For example, https://192.168.7.25:4433/500 is used as the URL to download 500-byte data, which will be displayed in the web browser, as shown in Figure 3-8.

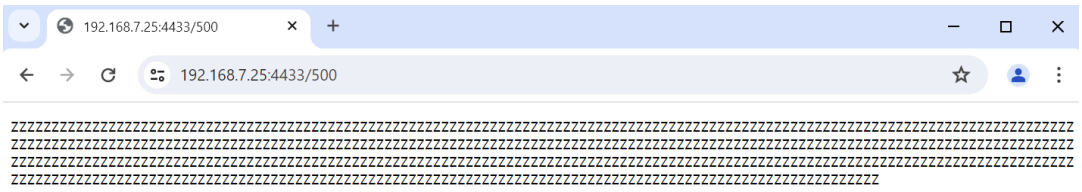

Figure 3-8 Download data pattern shown in the web browser

```
2024-05-27 16:01:28,719 INFO quic [05b3bbfb94816be4] HTTP request GET /500
2024-05-27 16:01:28,721 INFO quic [05b3bbfb94816be4] ALPN negotiated protocol h3
```

Figure 3-9 aioquic console when the client downloads data pattern

3.4 Upload with method POST

aioquic also offers a method to upload data to an aioquic server. By using a web browser, an application named 'httpEcho.html' is provided for uploading data in a secure connection with an aioquic server.

Users can open the webpage of this application by simply dragging and dropping the html file into a browser. At this point, a user interface appears which allows users to set parameters, such as server's IP address and port number, and to input the data message to be uploaded to the server, as shown in Figure 3-10. After setting the inputs, users press the "Send" button in order to send a POST command to the aioquic server with the URL endpoint "/echo".

Although the aioquic doesn't support a direct POST method by showing the data message at their side, it returns the data message back to the sender, the web browser in this case. As a result, the echoed data is displayed in the web browser, as depicted in Figure 3-11.

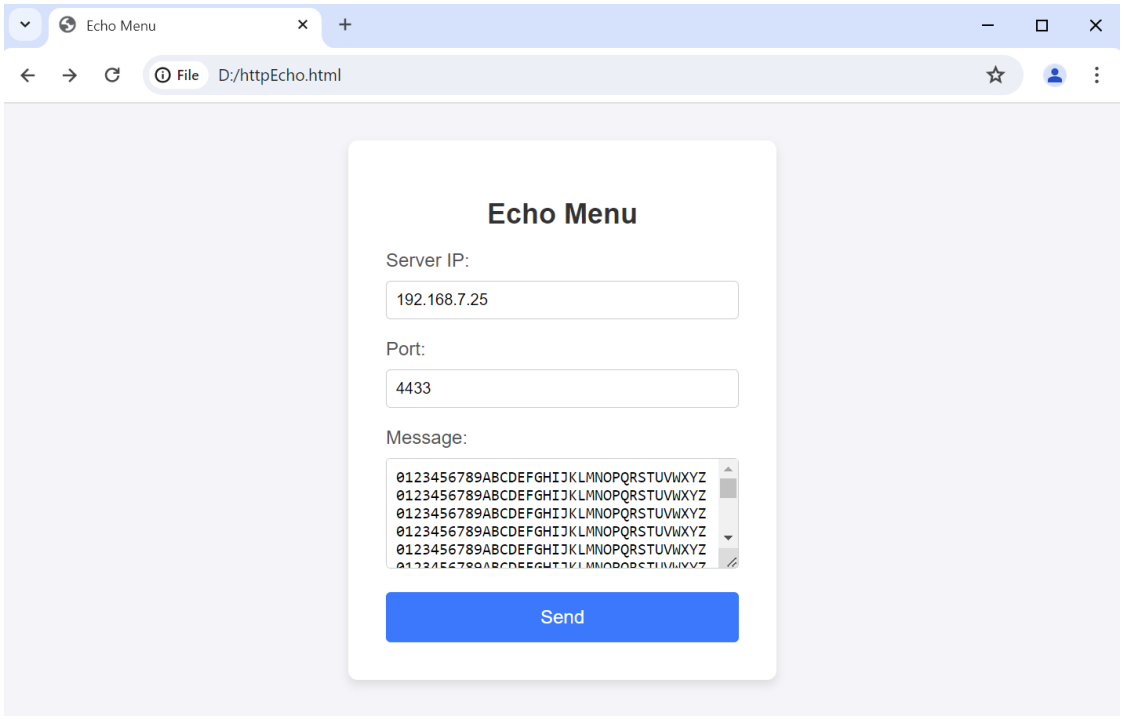

Figure 3-10 Upload a message via a web browser

Figure 3-11 Echo Data from the aioquic server shown in the web browser

```
2024-05-27 16:15:26,232 INFO quic [a9d17b5558b939aa] ALPN negotiated protocol h3
2024-05-27 16:15:26,239 INFO quic [a9d17b5558b939aa] HTTP request POST /echo
```

Figure 3-12 aioquic console when the client uploads data

4 MsQuic server

Another QUIC software implementation used in our reference is MsQuic, developed by Microsoft. Their work is an open-source software project, written in C and published in the following website: github.com/microsoft/msquic. We use MsQuic version 2.3.5 and would like to thank Microsoft and the MsQuic team for the development of MsQuic. There is no modification of MsQuic required to run with our demonstration, but we included a folk of MsQuic as a reference branch that we use in this repository: github.com/design-gateway/msquic. If you have any questions regarding their core, please kindly direct them to the MsQuic development team.

There are several application examples offered by MsQuic, but for our reference, an application called 'secnetperf' is applied to run with our demo due to the fact that it is optimized for the high-performance data transfer. For this reason, it uses its own application protocol rather than using the HTTP/3 protocol. Please follow MsQuic's guidelines to set up the application.

To run an MsQuic server using the secnetperf application, secnetperf.exe is called with two options – one is the IP address to bind the server to, and the other is the profile setting of this application, configured for maximum performance in this example. After running the binary file, the message displaying "Started!" is shown as a result, and there will be no other messages thereafter. The example of running the MsQuic is illustrated in Figure 4-1.

```
PS D:\37.QUIC\folks\msquic> ./artifacts/bin/windows/x64_Debug_openssl/secnetperf.exe -exec:maxtput -ip:192.168.7.25
Started!
```

Figure 4-1 MsQuic server application console

At this point, a client running the secnetperf application can be connected to the server. To run the client, four options are used in this example: "target" being the IP address of the server to be connected to, "exec" being the same setting as of the server, "up/down" being the length of the upload or download, and "ptput" being the setting to print throughput information. The example of the client console uploading data from the server is shown in Figure 4-2, while the example of downloading data to the server is shown in Figure 4-3.

```
PS D:\37.QUIC\folks\msquic> ./artifacts/bin/windows/x64_Debug_openssl/secnetperf -target:192.168.7.25 -up:1gb -ptput:1
-exec:maxtput
Started!

Result: Upload 1000000000 bytes @ 3413934 kbps (2343.337 ms).
```

Figure 4-2 MsQuic client application console uploading data

```
PS D:\37.QUIC\folks\msquic> ./artifacts/bin/windows/x64_Debug_openssl/secnetperf -target:192.168.7.25 -down:1gb -ptput:1
-exec:maxtput
Started!

Result: Download 1000000000 bytes @ 3593452 kbps (2226.271 ms).
```

Figure 4-3 MsQuic client application console downloading data

5 QUIC10GCdemo setup

1) Make sure the power switch is off and connect power supply to FPGA development board.
2) Connect two USB cables between the FPGA board and PC via micro-USB ports.
3) Power on the system.
4) Download the configuration file and firmware to the FPGA board by the following steps,
    a) open a Vivado TCL shell.
    b) change the current directory to the download folder where the demo configuration file is located.
    c) Type “QUIC10GCTest.bat” and press enter, as shown in Figure 5-1.

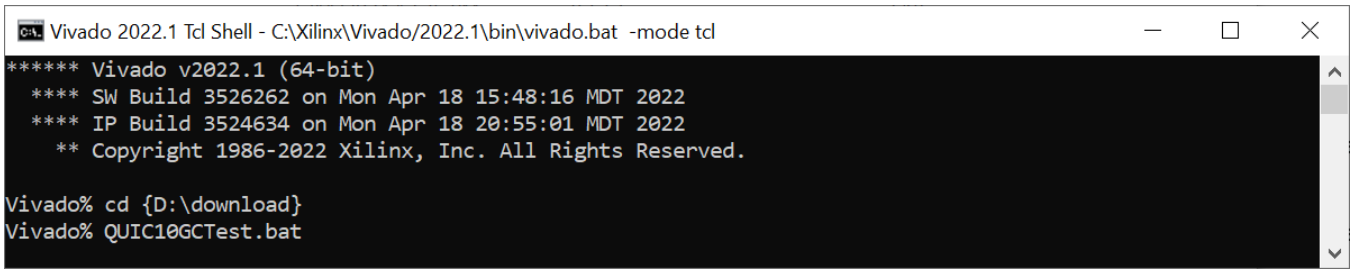

Figure 5-1 Example command script to download the configuration file

6 Serial Console

Users can set the parameters or run the download and upload applications by using the following commands. The QUIC10GCdemo commands and their usages will be displayed, as shown in Figure 6-1. Detailed information about each command is described in topic 7.

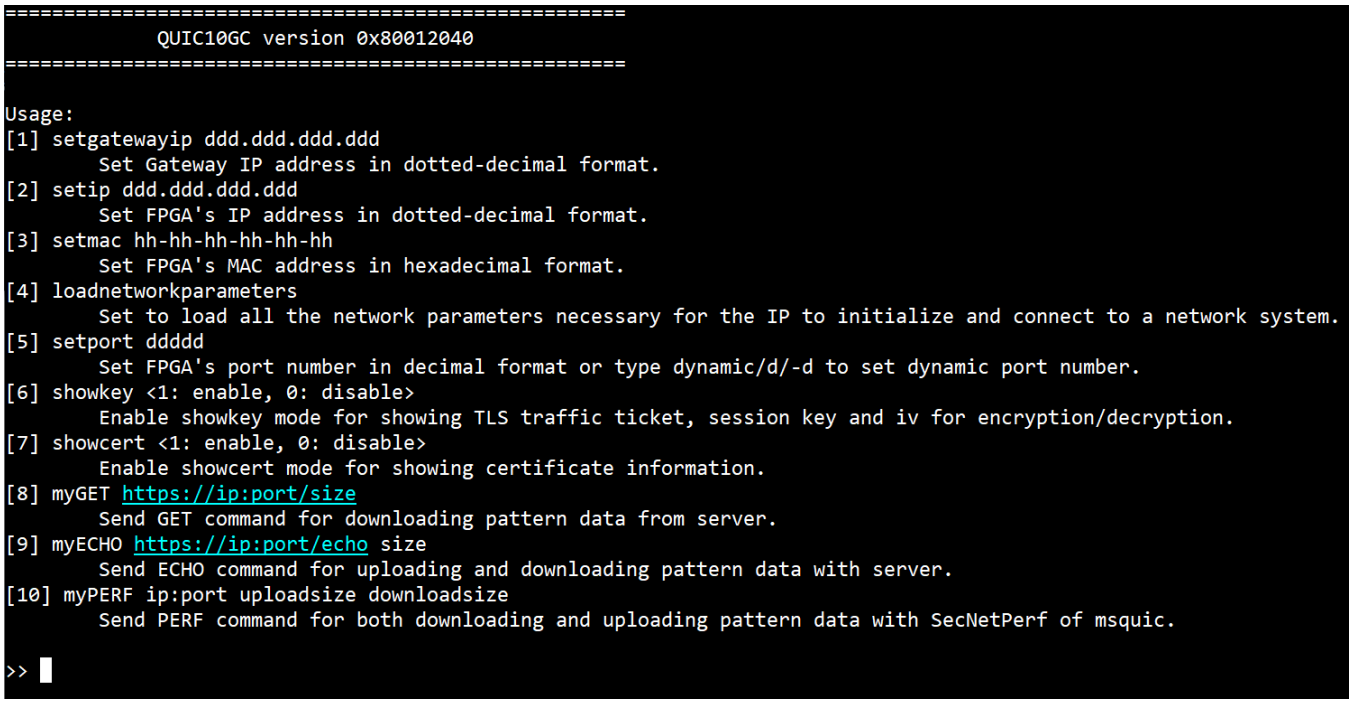

Figure 6-1 Serial console

7 Command detail

7.1 Set Gateway IP Address

command> setgatewayip ddd.ddd.ddd.ddd

This command is used to set the Gateway IP address in dotted-decimal format. The default Gateway IP address is 0.0.0.0, which indicates to the IP that there is no valid Gateway IP address. Users can input the setgatewayip command followed by a valid IP address, as shown in Figure 6-1.

7.2 Set FPGA's IP Address

command> setip ddd.ddd.ddd.ddd

This command is used to set the FPGA's IP address in dotted-decimal format. The default FPGA's IP address is 192.168.7.42. Users can input the setip command followed by a valid IP address, as shown in Figure 6-1.

7.3 Set FPGA's MAC address

command> setmac hh-hh-hh-hh-hh-hh

This command is used to set the FPGA's MAC address in hexadecimal format. The default FPGA's MAC address is 80-11-22-33-44-55, which is a unicast MAC address.

7.4 Load FPGA's network parameters

command> loadnetworkparameters

This command is used to load all the network parameters necessary for the IP to initialize and connect to a network system. These include the Gateway IP address, FPGA's IP address, and FPGA's MAC address. The QUIC10GC-IP must have all network parameters loaded at least once after a power-on reset or when the IP core system reset is active.

7.5 Set FPGA's Port Number

command> setport ddddd

This command is used to set the static port number of FPGA in decimal format. By default, the FPGA's port number is set to be dynamic. Dynamic ports range from 49152 to 65535. Users can enable dynamic port again after specifying a port number by using "setport dynamic" command, as shown in Figure 6-1. It is worth noting that this command is not listed in the necessary network parameters, and therefore, it can be used to change the port number at any time before opening connection.

7.6 Enable showkey mode

command> showkey <1: enable, 0: disable>

This command is used to enable the showkey mode. When the showkey mode is enabled, the TLS traffic ticket for encryption/decryption is displayed on the serial console, as shown in Figure 7-1. Users can utilize the TLS traffic ticket in the (Pre)-Master-Secret log file for Wireshark*, enabling them to decrypt transferred data between the client and server.

*Wireshark, a network packet analyzer tool used for network troubleshooting, analysis, and security purposes.

```
>> myPERF 192.168.7.25:4433 0 200000000
=========================================
Start IP initialization process

Traffic Secret
CLIENT_HANDSHAKE_TRAFFIC_SECRET 4D314C190E18AF16EF15DF13D711D110400E080D331832123212311730913 7D2 C768ABCF63A478580398994381FE6A8C9DBFAB2EE6E34C2B23A80547925E2F56
SERVER_HANDSHAKE_TRAFFIC_SECRET 4D314C190E18AF16EF15DF13D711D110400E080D331832123212311730913 7D2 4A90CC5D5333DDF4CF1E8EB41F86FFD734AE3387E193C722B37472A50E5F6A1F
CLIENT_TRAFFIC_SECRET_0 4D314C190E18AF16EF15DF13D711D110400E080D331832123212311730913 7D2 BF5B7F490E0505C5AEB2AF8896C55700721C5087B1A347C7A86713AFB6C4A6D1
SERVER_TRAFFIC_SECRET_0 4D314C190E18AF16EF15DF13D711D110400E080D331832123212311730913 7D2 4FD2848282D4DAC0405A279E31C7E5AC6733827D18A0E097023623B09D147496

IP initialization is complete

 Running... Done
=========================================
Pattern data has been verified, and
Data content is too large so only the transfer speed is displayed
=========================================
Total transfer size = 8 Byte(s)
Upload Speed 0.000 Mbps
Total transfer size = 200000000 Byte(s)
Download Speed 7692 Mbps
```

Figure 7-1 Serial console when the showkey mode is enabled

7.7 Enable showcert mode

command> showcert <1: enable, 0: disable>

This command is used to enable the showcert mode. When the showcert mode is enabled, the server's certificate stored in RAM called CertRam is displayed on the serial console, as shown in Figure 7-2. The certificate information is displayed in hexadecimal format, which corresponds to the result obtained by using openssl command: openssl x509 -in tls_cert.pem -outform der | hexdump -C, as shown in Figure 7-3.

```
Certificate information
 0B 00 03 60 00 00 03 5C  00 03 57 30 82 03 53 30
 82 02 3B A0 03 02 01 02  02 14 50 10 DD BC F4 A8
 3C 39 69 76 11 E8 B2 A0  CA 2B C5 67 89 8A 30 0D
 06 09 2A 86 48 86 F7 0D  01 01 0B 05 00 30 38 31
 0B 30 09 06 03 55 04 06  13 02 54 48 31 10 30 0E
 06 03 55 04 08 0C 07 42  61 6E 67 6B 6F 6B 31 17
 30 15 06 03 55 04 0A 0C  0E 44 65 73 69 67 6E 20
 47 61 74 65 77 61 79 30  20 17 0D 32 33 30 32 32
 34 30 39 32 38 31 30 5A  18 0F 32 31 32 33 30 31
 33 31 30 39 32 38 31 30  5A 30 38 31 0B 30 09 06
 03 55 04 06 13 02 54 48  31 10 30 0E 06 03 55 04
 08 0C 07 42 61 6E 67 6B  6F 6B 31 17 30 15 06 03
 55 04 0A 0C 0E 44 65 73  69 67 6E 20 47 61 74 65
 77 61 79 30 82 01 22 30  0D 06 09 2A 86 48 86 F7
 0D 01 01 01 05 00 03 82  01 0F 00 30 82 01 0A 02
 82 01 01 00 C0 36 8C 0A  DC 4F BF 0B 1C 40 3C 77
 17 EF BB 81 F3 C5 02 D2  F7 CA CA 96 CA D0 CD 3F
 0B 48 C1 87 FC F3 B7 13  5E 29 B6 C9 96 19 F4 ED
 BC C2 8D EB AF F6 92 0A  A2 B9 93 5C CF 34 BD 1B
 3C D1 24 54 7F 59 6B 75  9F F7 00 EE 38 4A 13 60
 72 96 23 97 21 6B 01 5A  22 40 94 63 8F 2B 24 4F
 07 64 36 D7 AF 55 14 B8  98 EB F7 DF 8F 03 07 2E
 EB 97 E8 64 78 73 17 18  A4 7B 79 2A FB 5E 4D 75
 06 C4 43 62 BA C7 5F A9  72 E5 8E 74 C5 AE B5 FE
 98 65 49 D3 7F C0 DE 39  31 9D 06 38 AC FA AD 68
 64 D0 3A B9 51 D6 24 53  7C 81 67 FD DB 19 A9 A8
 95 34 00 7E 83 F1 68 6C  59 CA 49 1D 99 D7 34 4C
 56 01 2A 83 D1 5C 12 CB  C8 83 4B AA 53 58 11 E6
 33 C0 BD A2 89 1E 4E 59  75 91 54 78 9D 85 3C FB
 C8 72 69 1F D1 97 E1 95  AA 25 D2 CB E8 90 A1 53
 48 34 29 7D B8 6F B3 80  AA CC 29 A8 D5 9C 82 47
 DB 75 8F 9F 02 03 01 00  01 A3 53 30 51 30 1D 06
 03 55 1D 0E 04 16 04 14  DA 27 4C 41 24 45 7B 02
 D8 58 0B 6C 13 EC 74 F9  6E FF DF AE 30 1F 06 03
 55 1D 23 04 18 30 16 80  14 DA 27 4C 41 24 45 7B
 02 D8 58 0B 6C 13 EC 74  F9 6E FF DF AE 30 0F 06
 03 55 1D 13 01 01 FF 04  05 30 03 01 01 FF 30 0D
 06 09 2A 86 48 86 F7 0D  01 01 0B 05 00 03 82 01
 01 00 3D C9 31 35 35 85  B8 84 0D 61 A0 25 C0 47
 A8 56 EC A3 A3 09 13 28  50 EE 2C 32 35 0F 33 C5
 A9 32 42 74 4D 54 28 28  6A C8 D7 4C B2 80 CC 90
 D0 A9 5B 06 E6 60 14 25  91 18 ED E1 EF 31 42 1E
 86 72 F2 4D 1B 9D 14 0C  6F 0C 96 DE FF D8 9E 85
 D6 89 7E 49 A8 59 6A 8A  21 28 F7 36 15 10 E7 11
 E3 78 48 4C A2 30 BF B4  93 F0 38 27 99 CE D1 73
 DE 42 FC 02 25 3C F2 1F  BD AA 32 02 2F EB 21 CB
 78 C0 CF C2 EE 84 E9 BF  EB 35 AB F4 C8 71 6C 23
 E8 F5 61 E6 03 8C 2D 43  1C 0A BF E8 E1 99 E8 B2
 93 A0 45 DA 58 15 ED 35  A2 0A A1 E2 75 EE EA C8
 8A 9F B9 D0 46 D9 7A 76  44 FB F1 FA 9B AB A8 79
 DC 40 7F 15 8D 57 A7 0B  D4 30 EB 2A 29 AE F6 70
 B2 F4 A3 61 5D B8 6C E0  CD FB 51 96 7A 01 18 12
 1C 3F 76 C4 84 D2 A8 9E  6F 65 FB 07 29 D9 24 C0
 FD 10 E4 98 3A B3 AB B4  76 4D C0 DE 44 00 4E E1
 37 62 00 00
```

Figure 7-2 Serial console when the showcert mode is enabled


```
D:\37.QUIC\folks\aioquic\tests>openssl x509 -in tls_cert.pem -outform der | hexdump -C
000000  30 82 03 53 30 82 02 3b a0 03 02 01 02 02 14 50  0..S0..;.......P
000010  10 dd bc f4 a8 3c 39 69 76 11 e8 b2 a0 ca 2b c5  .....<9iv.....+.
000020  67 89 8a 30 0d 06 09 2a 86 48 86 f7 0d 01 01 0b  g..0...*.H......
000030  05 00 30 38 31 0b 30 09 06 03 55 04 06 13 02 54  ..081.0...U....T
000040  48 31 10 30 0e 06 03 55 04 08 0c 07 42 61 6e 67  H1.0...U....Bang
000050  6b 6f 6b 31 17 30 15 06 03 55 04 0a 0c 0e 44 65  kok1.0...U....De
000060  73 69 67 6e 20 47 61 74 65 77 61 79 30 20 17 0d  sign Gateway0 ..
000070  32 33 30 32 32 34 30 39 32 38 31 30 5a 18 0f 32  230224092810Z..2
000080  31 32 33 30 31 33 31 30 39 32 38 31 30 5a 30 38  1230131092810Z08
000090  31 0b 30 09 06 03 55 04 06 13 02 54 48 31 10 30  1.0...U....TH1.0
0000a0  0e 06 03 55 04 08 0c 07 42 61 6e 67 6b 6f 6b 31  ...U....Bangkok1
0000b0  17 30 15 06 03 55 04 0a 0c 0e 44 65 73 69 67 6e  .0...U....Design
0000c0  20 47 61 74 65 77 61 79 30 82 01 22 30 0d 06 09   Gateway0.."0...
0000d0  2a 86 48 86 f7 0d 01 01 01 05 00 03 82 01 0f 00  *.H.............
0000e0  30 82 01 0a 02 82 01 01 00 c0 36 8c 0a dc 4f bf  0.........6...O.
0000f0  0b 1c 40 3c 77 17 ef bb 81 f3 c5 02 d2 f7 ca ca  ..@<w...........
000100  96 ca d0 cd 3f 0b 48 c1 87 fc f3 b7 13 5e 29 b6  ....?.H......^).
000110  c9 96 19 f4 ed bc c2 8d eb af f6 92 0a a2 b9 93  ................
000120  5c cf 34 bd 1b 3c d1 24 54 7f 59 6b 75 9f f7 00  \.4..<.$T.Yku...
000130  ee 38 4a 13 60 72 96 23 97 21 6b 01 5a 22 40 94  .8J.`r.#.!k.Z"@.
000140  63 8f 2b 24 4f 07 64 36 d7 af 55 14 b8 98 eb f7  c.+$O.d6..U.....
000150  df 8f 03 07 2e eb 97 e8 64 78 73 17 18 a4 7b 79  ........dxs...{y
000160  2a fb 5e 4d 75 06 c4 43 62 ba c7 5f a9 72 e5 8e  *.^Mu..Cb._.r..
000170  74 c5 ae b5 fe 98 65 49 d3 7f c0 de 39 31 9d 06  t.....eI....91..
000180  38 ac fa ad 68 64 d0 3a b9 51 d6 24 53 7c 81 67  8...hd.:.Q.$S|.g
000190  fd db 19 a9 a8 95 34 00 7e 83 f1 68 6c 59 ca 49  ......4.~..hlY.I
0001a0  1d 99 d7 34 4c 56 01 2a 83 d1 5c 12 cb c8 83 4b  ...4LV.*..\....K
0001b0  aa 53 58 11 e6 33 c0 bd a2 89 1e 4e 59 75 91 54  .SX..3.....NYu.T
0001c0  78 9d 85 3c fb c8 72 69 1f d1 97 e1 95 aa 25 d2  x..<..ri......%.
0001d0  cb e8 90 a1 53 48 34 29 7d b8 6f b3 80 aa cc 29  ....SH4)}.o....)
0001e0  a8 d5 9c 82 47 db 75 8f 9f 02 03 01 00 01 a3 53  ....G.u........S
0001f0  30 51 30 1d 06 03 55 1d 0e 04 16 04 14 da 27 4c  0Q0...U.......'L
000200  41 24 45 7b 02 d8 58 0b 6c 13 ec 74 f9 6e ff df  A$E{..X.l..t.n..
000210  ae 30 1f 06 03 55 1d 23 04 18 30 16 80 14 da 27  .0...U.#..0....'
000220  4c 41 24 45 7b 02 d8 58 0b 6c 13 ec 74 f9 6e ff  LA$E{..X.l..t.n.
000230  df ae 30 0f 06 03 55 1d 13 01 01 ff 04 05 30 03  ..0...U.......0.
000240  01 01 ff 30 0d 06 09 2a 86 48 86 f7 0d 01 01 0b  ...0...*.H......
000250  05 00 03 82 01 01 00 3d c9 31 35 35 85 b8 84 0d  .......=.155....
000260  61 a0 25 c0 47 a8 56 ec a3 a3 09 13 28 50 ee 2c  a.%.G.V.....(P.,
000270  32 35 0f 33 c5 a9 32 42 74 4d 54 28 28 6a c8 d7  25.3..2BtMT((j..
000280  4c b2 80 cc 90 d0 a9 5b 06 e6 60 14 25 91 18 ed  L......[..`.%...
000290  e1 ef 31 42 1e 86 72 f2 4d 1b 9d 14 0c 6f 0c 96  ..1B..r.M....o..
0002a0  de ff d8 9e 85 d6 89 7e 49 a8 59 6a 8a 21 28 f7  .......~I.Yj.!(.
0002b0  36 15 10 e7 11 e3 78 48 4c a2 30 bf b4 93 f0 38  6.....xHL.0....8
0002c0  27 99 ce d1 73 de 42 fc 02 25 3c f2 1f bd aa 32  '...s.B..%<....2
0002d0  02 2f eb 21 cb 78 c0 cf c2 ee 84 e9 bf eb 35 ab  ./.!.x........5.
0002e0  f4 c8 71 6c 23 e8 f5 61 e6 03 8c 2d 43 1c 0a bf  ..ql#..a...-C...
0002f0  e8 e1 99 e8 b2 93 a0 45 da 58 15 ed 35 a2 0a a1  .......E.X..5...
000300  e2 75 ee ea c8 8a 9f b9 d0 46 d9 7a 76 44 fb f1  .u.......F.zvD..
000310  fa 9b ab a8 79 dc 40 7f 15 8d 57 a7 0b d4 30 eb  ....y.@...W...0.
000320  2a 29 ae f6 70 b2 f4 a3 61 5d b8 6c e0 cd fb 51  *)..p...a].l...Q
000330  96 7a 01 18 12 1c 3f 76 c4 84 d2 a8 9e 6f 65 fb  .z....?v.....oe.
000340  07 29 d9 24 c0 fd 10 e4 98 3a b3 ab b4 76 4d c0  .).$.....:...vM.
000350  de 44 00 4e e1 37 62                              .D.N.7b
```

Figure 7-3 Certificate information from the openssl command

7.8 GET method

command> myGET protocol://ip:port/length

This command simulates the GET method of HTTP/3 to download data from the aioquic server. To run with an aioquic server, the protocol of the URL must be 'https', the server's IP address and port number are separated by a colon, and the length of the download data can be specified at the end of the URL. If the length parameter is left blank, the aioquic server will return the homepage, and the received data will be displayed on the serial console, as shown in Figure 7-4.

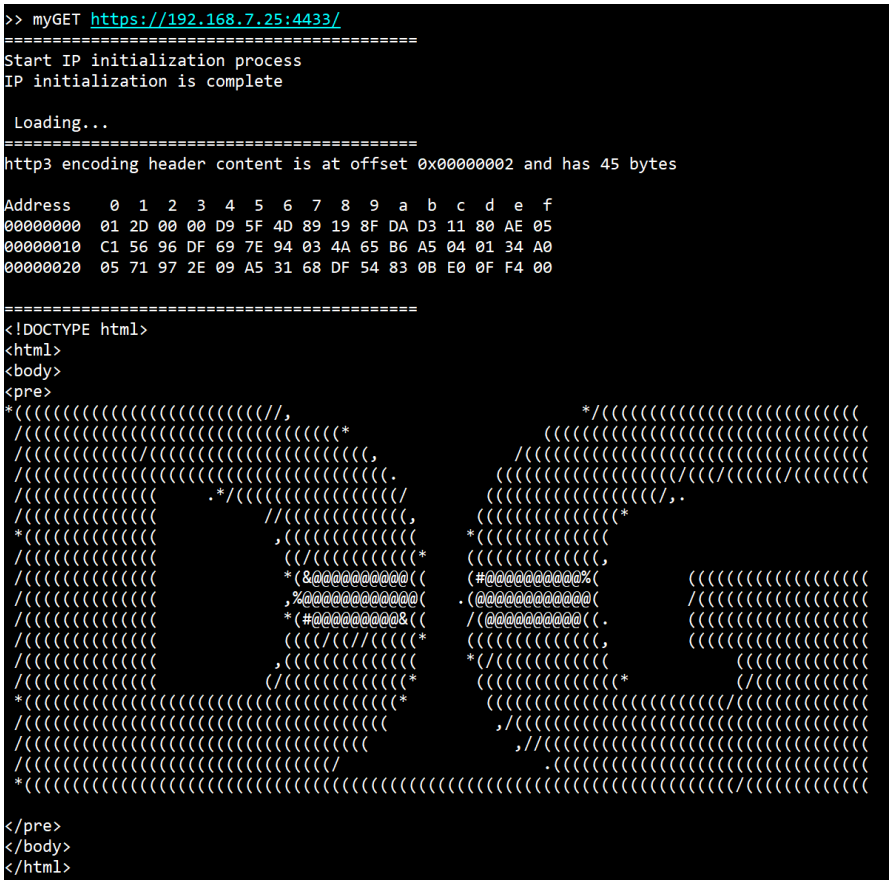

Figure 7-4 Serial console when downloading the homepage

On the other hand, if the length parameter is not blank, the aioquic server will be requested to return the data. Once the data is downloaded, it will be displayed in the serial console, shown in Figure 7-5, with one exception that if the length of the data exceeds 16kB, the message "Data Length is too large, Show only Transfer speed" will be displayed instead, as shown in Figure 7-6. In both cases, the total length of the received data and the download speed will be displayed on the serial console.

Due to the aioquic specification, the maximum data length is capped at 50MB for testing with the aioquic server. If users request to download data exceeding this limit, the software will retrieve only 50MB from the server.

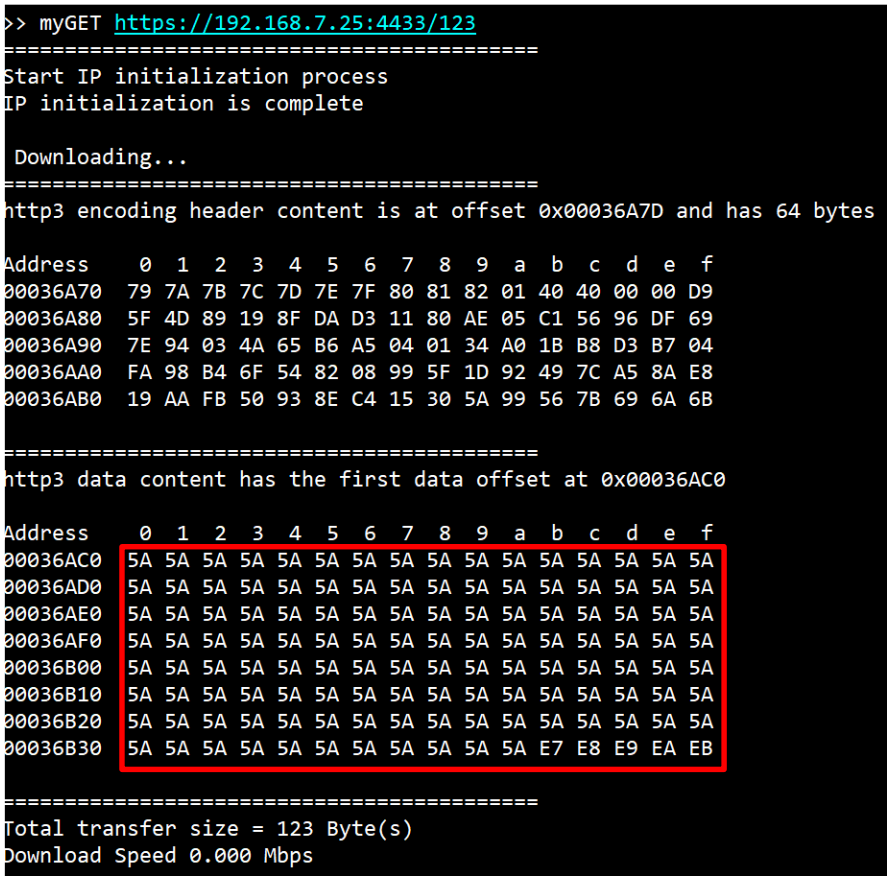

Figure 7-5 Serial console when downloading small data

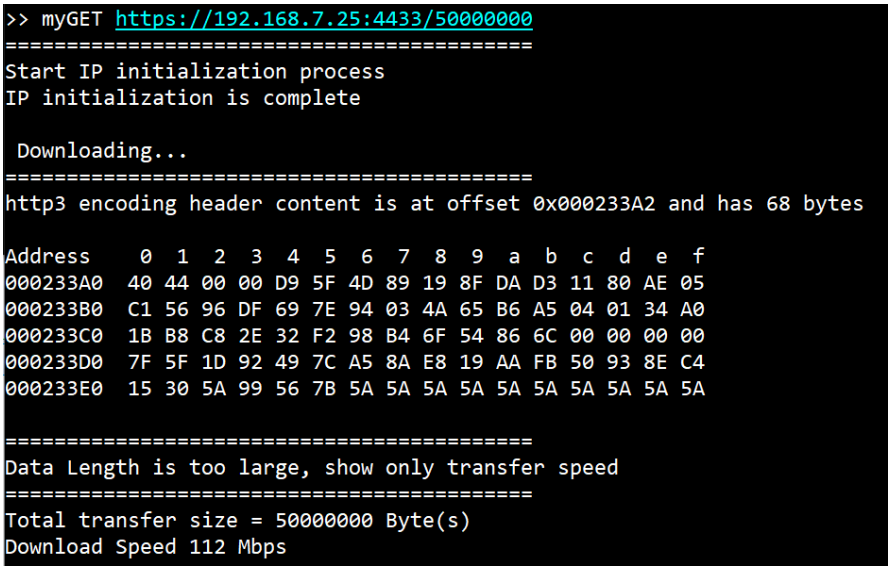

Figure 7-6 Serial console when downloading large data

7.9 POST method

command> myECHO protocol://ip:port/echo length

This command simulates the POST method of HTTP/3 to upload data to the aioquic server. The URL structure for running the POST method is similar to the GET method with the exception that the transfer length is replaced by a string of “echo”. Users can instead specify the length of the uploading data, which is an 8-bit counting pattern, in another option. After the upload is completed, the aioquic server will return what it has received. This allows the users to verify the received data from the aioquic server. By enabling the verification feature, it monitors whether the received data matches the expected pattern or not, and after verifying it, the data content, transfer length, and transfer speeds are displayed, as shown in Figure 7-7 and Figure 7-8.

```
>> myECHO https://192.168.7.25:4433/echo 123
==========================================
Start IP initialization process
IP initialization is complete

 Uploading... Downloading...
==========================================
http3 encoding header content is at offset 0x000029DF and has 44 bytes

Address    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
000029D0  0C 00 01 02 03 04 05 06 07 08 09 0A 0B 01 2C 00
000029E0  00 D9 5F 4D 89 19 8F DA D3 11 80 AE 05 C1 56 97
000029F0  DF 3D BF 4A 08 0A 65 B6 A5 04 01 34 A0 1F B8 D3
00002A00  37 1A 6D 4C 5A 37 FF 54 82 08 99 00 00 00 00 00

==========================================
Echoed data has been verified, and
Showing Rx data content with the first data offset at 0x00002A0E

Address    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00002A00  37 1A 6D 4C 5A 37 FF 54 82 08 99 00 40 7B 00 01
00002A10  02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11
00002A20  12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21
00002A30  22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31
00002A40  32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40 41
00002A50  42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51
00002A60  52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61
00002A70  62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71
00002A80  72 73 74 75 76 77 78 79 7A 00 00 00 00 00 00 00

==========================================
Total transfer size = 123 Byte(s)
Upload Speed 0.984 Mbps
Total transfer size = 123 Byte(s)
Download Speed 0.000 Mbps
```

Figure 7-7 Serial console when uploading small data

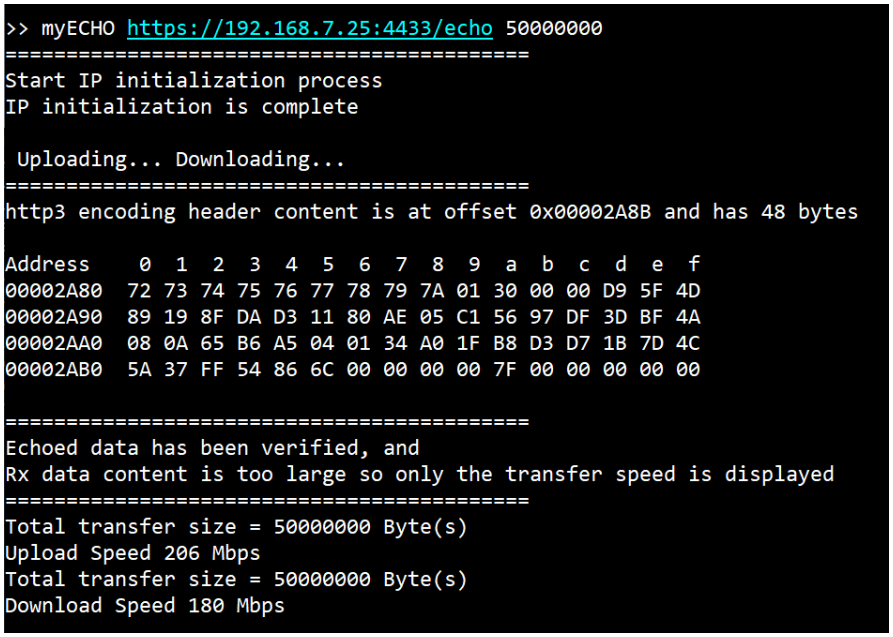

Figure 7-8 Serial console when uploading large data

7.10 PERF method

command> myPREF ip:port uploadlength downloadlength

This command is designed to run with the “secnetperf” example of an MsQuic server. There are three parameters required to run this command – the first option is the server’s IP address and server’s port number separated by a colon, the second is the length of the upload data, and the last one is the length of the download data.

Specifically, this application protocol is designed to have the client transferring the upload data firstly, after which the client receives the download data from the server. Similar to the POST method, the verification feature is used to monitor the received data, and the results, such as the download content, the transfer length, and the transfer speeds, are presented, as shown in Figure 7-9. It is also important to note that the performance of this operation depends on the network system and the resources available on the test machine.

```
>> myPERF 192.168.7.25:4433 0 123
=========================================
Start IP initialization process
IP initialization is complete

 Running... Done
=========================================
Pattern data has been verified, and
Showing Rx data content with the first data offset of 0x00036954

Address    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00036950  FC FD FE FF 00 00 00 00 00 00 00 00 08 09 0A 0B
00036960  0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B
00036970  1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B
00036980  2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B
00036990  3C 3D 3E 3F 40 41 42 43 44 45 46 47 48 49 4A 4B
000369A0  4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B
000369B0  5C 5D 5E 5F 60 61 62 63 64 65 66 67 68 69 6A 6B
000369C0  6C 6D 6E 6F 70 71 72 73 74 75 76 77 78 79 7A 7B

=========================================
Total transfer size = 8 Byte(s)
Upload Speed 0.000 Mbps
Total transfer size = 123 Byte(s)
Download Speed 0.000 Mbps
```

Figure 7-9 Serial console when downloading small data

Figure 7-9 is not a good example to represent the transfer performance because the operation time is too small for accurate calculation. Figure 7-10, however, can be used to show the transfer speeds for both upload and download because the transfer size settings are large enough.

```
>> myPERF 192.168.7.25:4433 16000000000 16000000000
=========================================
Start IP initialization process
IP initialization is complete

 Running... Done
=========================================
Pattern data has been verified, and
Data content is too large so only the transfer speed is displayed
=========================================
Total transfer size = 16000000000 Byte(s)
Upload Speed 3537 Mbps
Total transfer size = 16000000000 Byte(s)
Download Speed 9052 Mbps
```

Figure 7-10 Serial console when downloading large data

8 Revision History

Revision	Date	Description
1.00	2-Jul-24	Initial version release

