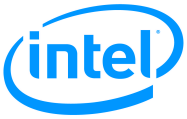
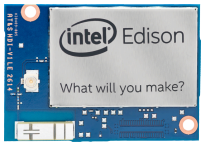
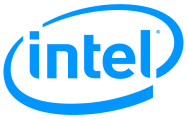




Intel® Edison Tutorial: TCP Socket Communications



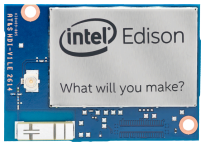
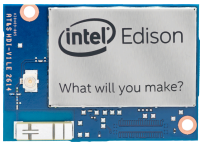


Table of Contents



Introduction

In this tutorial, users will:

1. Learn about TCP/IP.
2. Implement a single-threaded TCP/IP server-client system to enable data transfer.

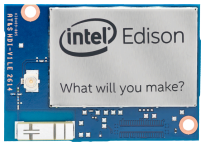
Prerequisite Tutorials

Users should ensure they are familiar with the documents listed below before proceeding.

1. Intel Edison Tutorial – Introduction, Linux Operating System Shell Access
2. Intel Edison Tutorial – Introduction to Linux
3. Intel Edison Tutorial – Introduction to Vim

List of Required Materials and Equipment

- 2x Intel Edison Kit.
- 4x USB 2.0 A-Male to Micro B Cable (micro USB cable).
- 2x powered USB hub OR an 2x external power supply.
- 1x Grove – Starter Kit for Arduino.
- 1x Personal Computer.
- 1x Wi-Fi network configured to access the internet.



Transmission Control Protocol (TCP)

Transmission **C**ontrol **P**rotocol (TCP) is one of the main protocols of the Internet Protocol (IP) suite. Another major protocol is **U**nified **D**atagram **P**rotocol (UDP). The table below highlights some of the key differences between these protocols.

	TCP	UDP
Reliability	Attempts to retransmit lost packets	Does not track if packets are lost
Ordering the packets	Yes	No
Latency	Higher latency – slower data transmission	Lower latency – faster data transmission
Notable attributes	Ordered data transfer, retransmission of lost packets, error-free data transfer, flow control, congestion control	Datagram, transaction-oriented, simple, stateless, does not retransmit data
Use cases	Most applications including: world wide web, email, remote administration	Network Time Protocol (NTP), IP tunneling, bootstrapping, streaming media, Voice over IP (VOIP), online games, broadcasting information

TCP/IP – General System Architecture

A block diagram illustrating the typical operation of a single-threaded TCP/IP client-server system can be found in Figure 1.

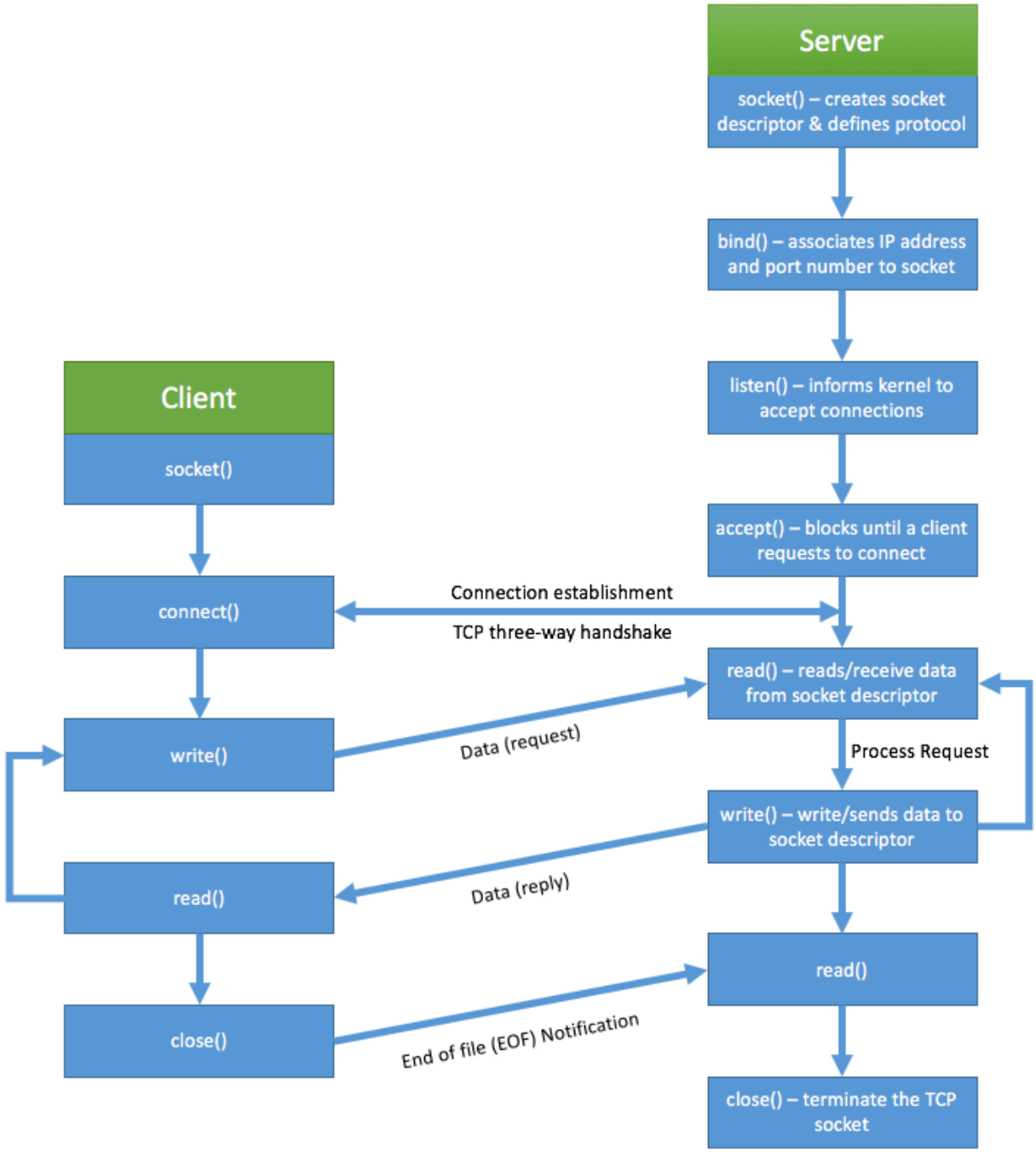


Figure 1: Block diagram of a typical TCP/IP client-server architecture

TCP/IP – Example System Implementation

This tutorial will cover a single-threaded client-server model for data transfer. Analyze the block diagram below before proceeding.

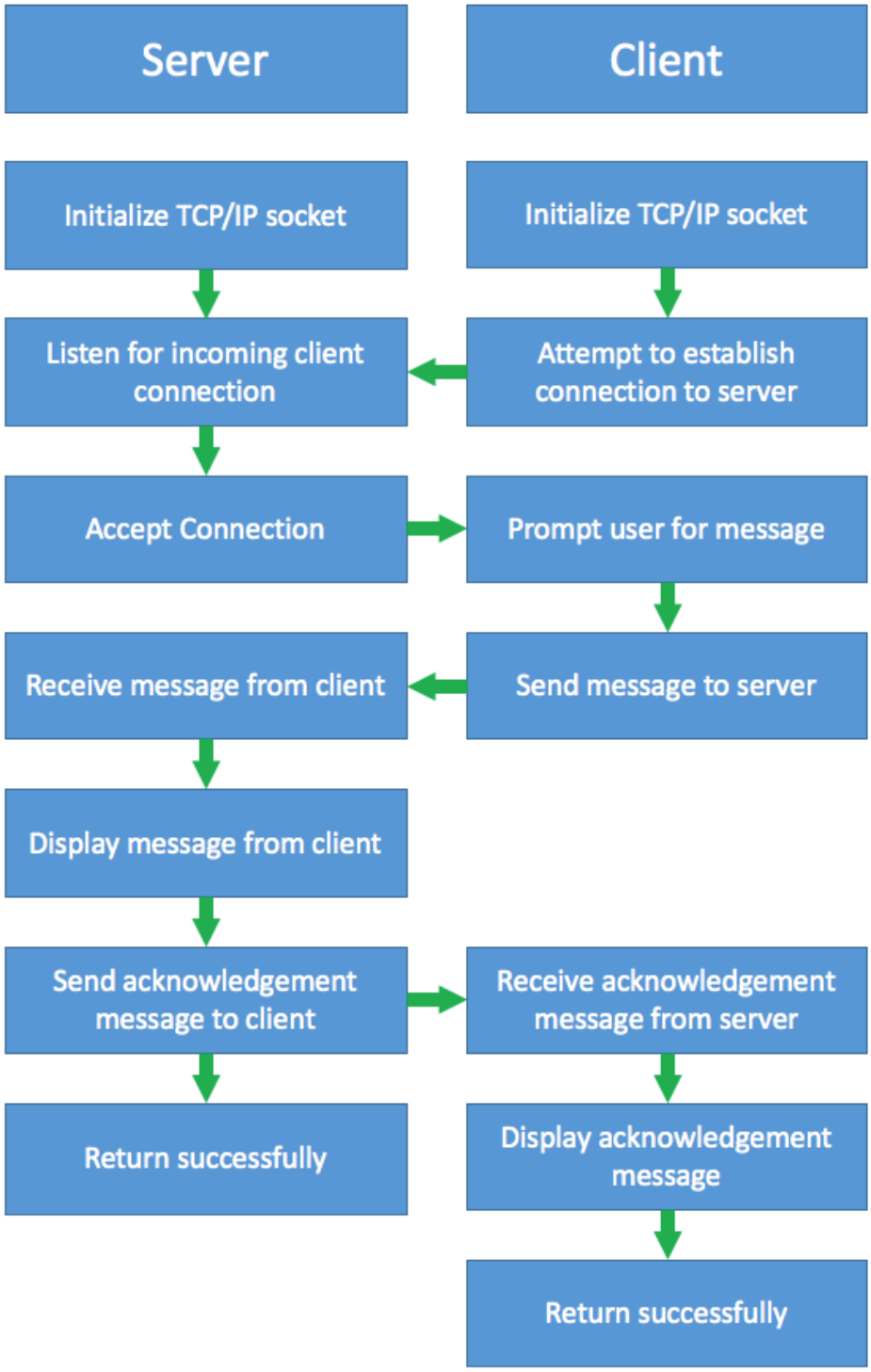
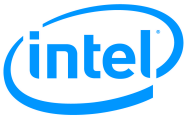
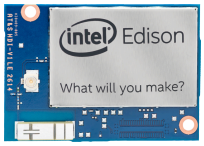


Figure 2: Block diagram of this particular TCP/IP client-server system





Software Retrieval and Other Notes

Follow the steps outlined below to configure the software for the Intel Edison's for this tutorial.

1. Open the following link on a web browser on a personal computer.

   https://drive.google.com/drive/folders/0B4NGslzPqDhveUR6eWNqLVo1S2s

2. Download and extract the contents of the zip archive labelled **FILES**. The archive should contain the two files listed below.

   client.c
   server.c

3. Upload the C code source file labelled **client.c** to one of the two Intel Edison's. This Intel Edison will be referred to as **the client** for the rest of this tutorial.

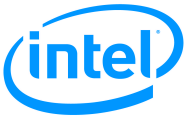
   For more information on how to transfer files between a personal computer and a remote computing device, refer to the document labelled ***Intel Edison Tutorial – Introduction, Linux Operating System Shell Access and SFTP***.

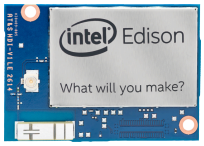
4. Upload the C code source file labelled **server.c** to the other Intel Edison. This Intel Edison will be referred to as **the server** for the rest of this tutorial.

5. Access the shell on each Intel Edison through SSH. This will ensure the Intel Edison device is configured to access the internet. Ensure that all SSH sessions remain open while performing this tutorial.

   Terminating an SSH session will issue the **sig**nal **h**ang **up** (SIGHUP) signal to all process started in the session. This signal will cause a process to terminate. To override this behavior, developers can explicitly ignore the SIGHUP signal in their source code, use the **nohup** prefix to a command, or follow an alternative strategy. For more information about **nohup**, refer to the link below.

   http://man7.org/linux/man-pages/man1/nohup.1.html





Server Software Configuration

1. Access the shell program on the server Intel Edison.
2. Issue the following command on the server, and record the IP address printed to standard out.

   $ configure_edison --showWiFiIP

3. **$ gcc -o server server.c**

```
root@ucla_iot_dev:~/SUSP/TCP# gcc -o server server.c
root@ucla_iot_dev:~/SUSP/TCP#
```

Figure 3: Successful compilation of the C code source file server.c

4. The syntax for the executable binary file labelled **server** is as follows.

   $ ./server <PORT>

   Where the value of <PORT> should be between 1000 and 8000.

5. Issue the command below to start the server.

   $ ./server 5000

   The server is functioning correctly if the cursor is on a blank line in the terminal session.

```
root@ucla_iot_dev:~/SUSP/TCP# ./server 5000
```

Figure 4: Successful execution of the server

   If the server prints the "ERROR on binding: Address already in use" message to standard error, try a different port number such as 8000.

   $ ./server 8000

```
root@ucla_iot_dev:~/SUSP/TCP# ./server 5000
ERROR on binding: Address already in use
root@ucla_iot_dev:~/SUSP/TCP# ./server 8000
```

Figure 5: Error message displayed when the desired port has not been released, and successful execution after selecting another port

Client Software Configuration

1. Access the shell program on the client Intel Edison.
2. Issue the following command.

   $ gcc -o client client.c

```
root@ucla_iot:~/SUSP/TCP# gcc -o client client.c
root@ucla_iot:~/SUSP/TCP#
```

Figure 6: Successful compilation of the C code source code file client.c

3. **$ ./client <IP_ADDR> <PORT>**

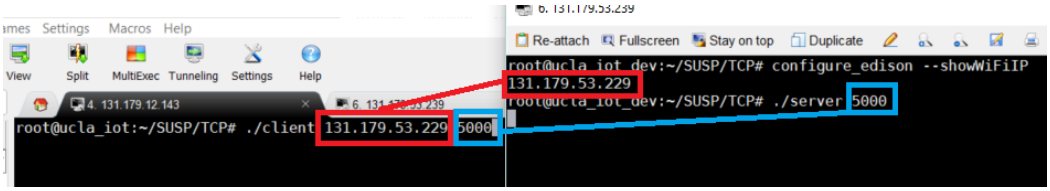


Figure 7: Determination of command line arguments for correct system operation

```
root@ucla_iot:~/SUSP/TCP# ./client 131.179.53.229 5000
Please enter the message:
```

Figure 8: Client process prompting the user for input after establishing a connection to the server

Type a message and press [Enter].

```
root@ucla_iot:~/SUSP/TCP# ./client 131.179.53.229 5000
Please enter the message: hello
I got your message
root@ucla_iot:~/SUSP/TCP#
```

Figure 9: Client process displaying the acknowledgement string sent from the server

4. Examine the output on the shell of **the server** Intel Edison.

```
root@ucla_iot_dev:~/SUSP/TCP# ./server 5000
Here is the message: hello

root@ucla_iot_dev:~/SUSP/TCP#
```

Figure 10: Server process displaying the user input string sent from the client using the TCP/IP protocol

Notice how the server exits as soon as one message has been received.