AES
Automatisierungs- und Energiesysteme

Lehrstuhl für Automatisierung und Energiesysteme

Univ.-Prof. Dr.-Ing Georg Frey

UNIVERSITÄT DES SAARLANDES

Seminar Report

-Seminar on Advanced Topics in Automation and Energy Systems-

-Smart Grid Communication-

Submitted by

Anirudh Upadhya

2581460

Saarbrücken, 2020

SUPERVISOR

Prof. Dr.-Ing. Georg Frey
Lehrstuhl für Automatisierungs und Energiesysteme

Daud Mustafa Minhas, M.Sc.
Lehrstuhl für Automatisierungs und Energiesysteme

Table of Contents

Smart Grid Communication

1. Introduction:

Different inventive remote gadgets, for example, cell phones/tablets and other wearable short-run remote gadgets are installed in each comprehensible spot from shoes to vehicles to structures and have become a basic piece of regular day to day existence at both individual and business levels. These gadgets might be sensors to screen different conditions to improve or give knowledge into the usefulness of the installed framework, be it human or machine. Some different gadgets may likewise give simple admittance to capacities in scope of hardware from forced air systems to entryway locks. A consistently expanding number of these gadgets can speak with one another without the mediation of people. With the driving force towards creating brilliant urban communities everywhere on over the world, it is turning out to be obvious since savvy urban communities will rely upon a keen framework and microgrid for their power needs. Continuous vitality the executives robotization in microgrids may end up being a significant advance towards actualizing brilliant matrix objectives for savvy urban communities. Before, networking was done using a wired communication media, such as Ethernet and Power Line Communication (PLC). As of late, wireless communication media with a low power RF communication module, such as Wi-Fi, ZigBee, Bluetooth, and Z-Wave, have been widely exploited to provide network functions.

In this seminar, the communication strategies between master and slave nodes are discussed. The sensors which are directly coupled to the slave nodes carry information about the sensed parameter which is then transmitted to the master node. In the master node these values are conditioned and optimized. The optimized/predicted values are then conveyed to the slave nodes which actuate based on the same.

The main focus of this seminar is coming up with design to incorporate the above requirement making use of the existing infrastructure.

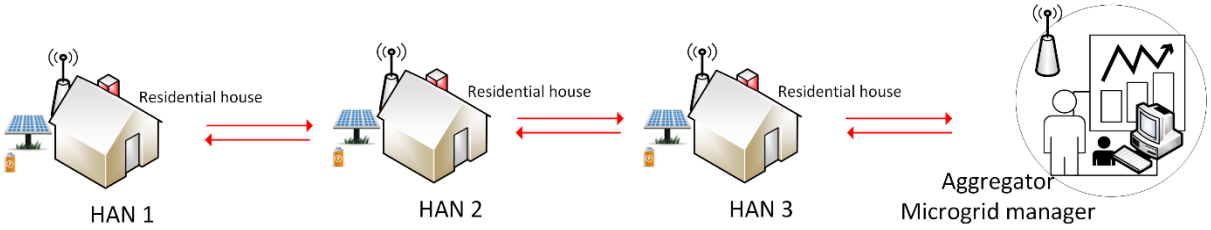


Figure1: The overall pictorial representation of the idea.

2. Client-Server Communication

In the current design we have used Raspberry Pis as master and slave nodes and have implemented a communication channel between them.

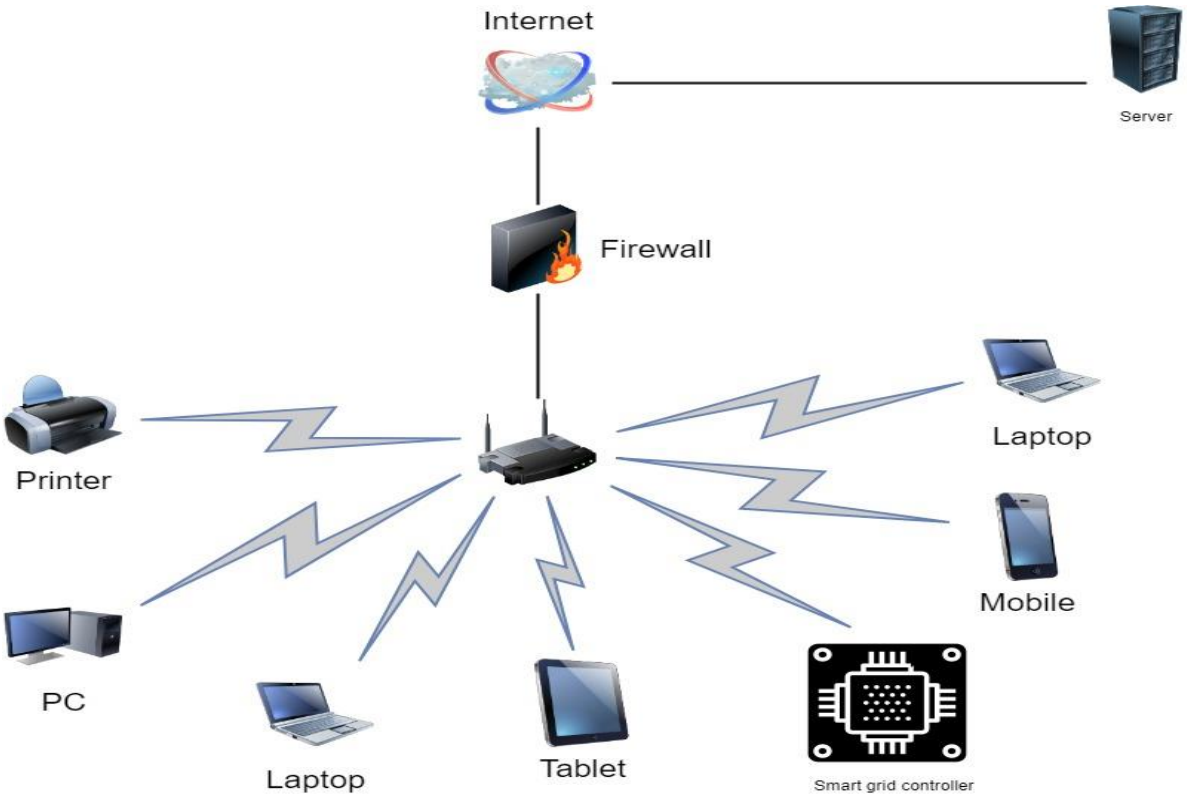
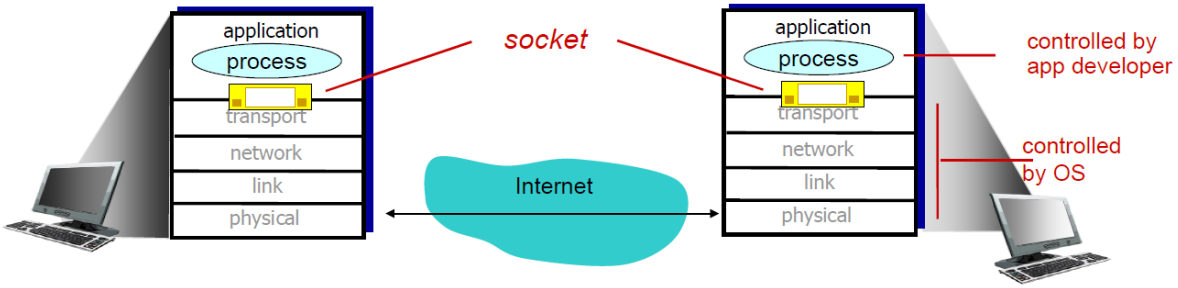


Figure2: Making use of existing infrastructure for communication.

The 2 Raspberry Pis acts as slave and read values fed to it from the devices/sensors and One Raspberry Pi acts as master which controls everything. Existing infrastructure of network stack in the Linux OS of Raspberry Pi is made use of for communication.

2.1 Sockets

Sockets are interfaces that can "plug into" each other over a network. Once so "plugged in", the programs so connected communicate. A "server" program is exposed via a socket connected to a certain /etc/services port number. A "client" program can then connect its own socket to the server's socket, at which time the client program's writes to the socket are read as stdin to the server program, and stdout from the server program are read from the client's socket reads. It's a door between application process and end-end transport protocol.



There are two types of sockets which can be used based on the transport layer services. UDP (Unreliable Datagram) and TCP (Reliable and byte stream oriented).

In UDP, no hand shaking before sending data and transmitted data maybe lost or received out-of-order. UDP is faster, simpler and more efficient than TCP. There is no retransmission of lost packets and it doesn't have flow control or congestion control mechanism.

In TCP, it's a connection-oriented protocol. TCP is reliable as it guarantees delivery of data to the destination router. Retransmission of lost packets is possible but is slower than UDP. It has both congestion and flow control.

In our use case, TCP with underlying IP as Network layer is used. This decision is done as data integrity was of more priority than the speed. Also, as we plan to make use of existing internet infrastructure and it would be difficult to allocate a certain Bandwidth to this process. Because of these reasons TCP/IP was used for communication between the master and slave nodes.

TCP	UDP
It is a connection-oriented protocol.	It is a connectionless protocol.
TCP is reliable as it guarantees delivery of data to the destination router.	The delivery of data to the destination cannot be guaranteed in UDP.
TCP provides extensive error checking mechanisms.	UDP has only the basic error checking mechanism using checksums.
TCP is comparatively slower than UDP.	UDP is faster, simpler and more efficient than TCP.
Retransmission of lost packets is possible in TCP, but not in UDP.	There is no retransmission of lost packets in User Datagram Protocol (UDP).
It has congestion and flow control	It doesn't have flow control mechanism
TCP is used by HTTP, HTTPs, FTP, SMTP and Telnet.	UDP is used by DNS, DHCP, TFTP, SNMP, RIP, and VoIP.

Figure3: Differences between TCP and UDP

Congestion Control in Raspberry Pi:

Raspberry Pi uses BBR congestion control. The BW is probed by using an exponential increase till congestion window threshold and linear increase after that. If in case a loss is identified due to duplicate ACKs then the window is decreased to threshold and again it increases linearly. If in case its due to time out then the congestion window starts from beginning and increases exponentially till threshold.

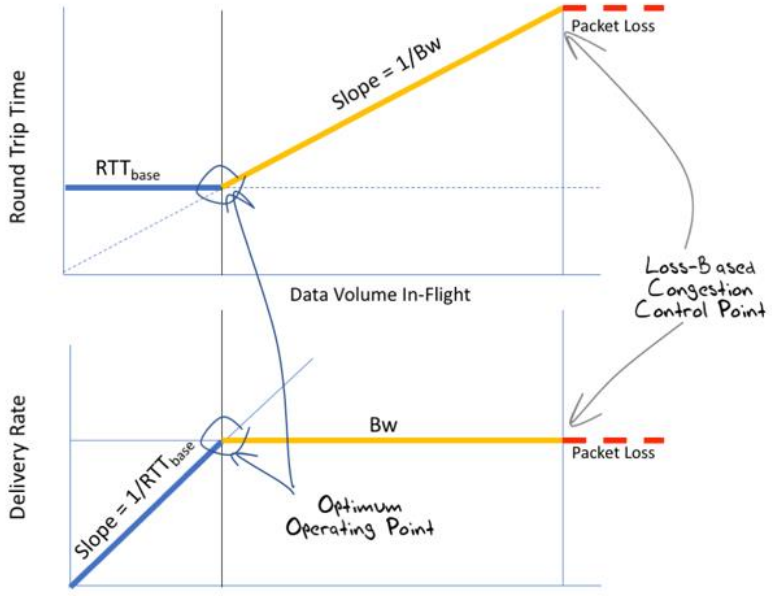


Figure4: Congestion control BBR in Raspberry Pi

Server:

All of the 3 nodes (Raspberry Pis) act as both client and server. The server process must be running first to start communication between client and server. In the server a socket is created which welcomes clients contact. As discussed earlier, a TCP socket is created specifying the IP address and port number of the server process. When contacted by client, server TCP creates new socket for server process to communicate with that particular client. This allows server to talk to multiple clients and server port number is used to distinguish between clients.

Firstly, in the server process the port and IP address for a particular client is initialized. The transport layer is configured (in our case TCP). Secondly, the receive buffer size is specified and now the server is ready to receive.

In this seminar, Excel is used for data logging. The server provides an excel, the received data from the client is segregated based on the parameter and then the values are populated to the excel. This process is similar for all the 3 Raspberry Pis as all of them act as server and client.

In the raspberry pi coupled to sensors (slave) and devices would read the values from them and store it in a buffer. This is values from this buffer are extracted and then populated to the excel. In the master node the values from the slave which are logged are transmitted periodically. These values are then received in master and logged using same method as slave. These values are then used to run prediction and conditioning algorithms which will eventually be transmitted back to the slaves.

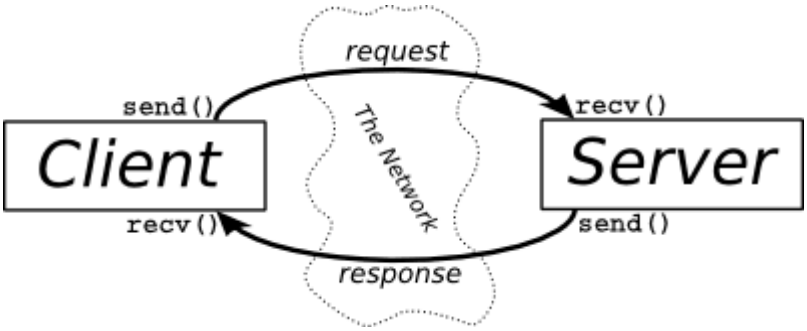


Figure5: Client-Server representation

Client:

The client process must be started after the server is up and running. In this process the server port and IP address which we want to communicate to specified. When client creates socket, Client TCP establishes connection to server TCP.

In the client process, the socket is created and the server IP and port are configured. The data to be transmitted is sent via TCP/IP. The data from the sensors and devices are directly logged to the excel file provided. These data which are logged to excel are the extracted and transmitted to the server. The connection between them can be closed if the intended operation is complete. Since this is a continuous process both client and server are running all the time.

2.2 Security

The internet can be malicious. Several types of malware and cyber hacking codes can infect networks and dramatically damage them instantaneously. As a result, firewall network security features diverse types of methods to combat damaging malware. The method for intrusion prevention depends on the type of firewall in effect and the predicted dangers.

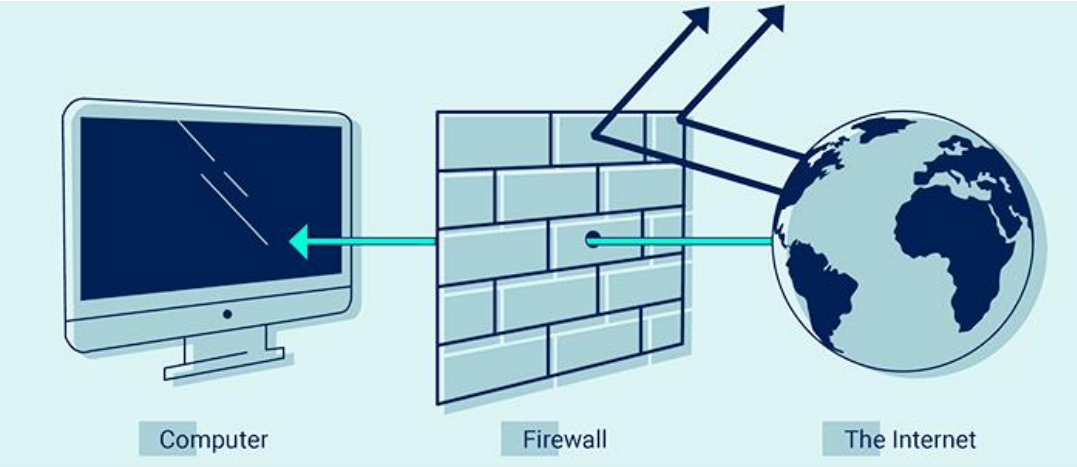


Figure6: Firewall protection from the internet

In our project we would configure it in way that its only accessible by the known devices and sensors which it is supposed to communicate and we will block all the rest of the connections. This can be done by installing firewall security application to Pi OS and configuring it manually to the desired IP addresses.

2.3 Implementation

In the current seminar, Raspberry Pi 4 Model B is used as both client and server. Total of three Pis are used out of which two acts as slave and one act as master.

The entire client-server process and programs are coded in python.

The program flow is as depicted in the Figure7 below:

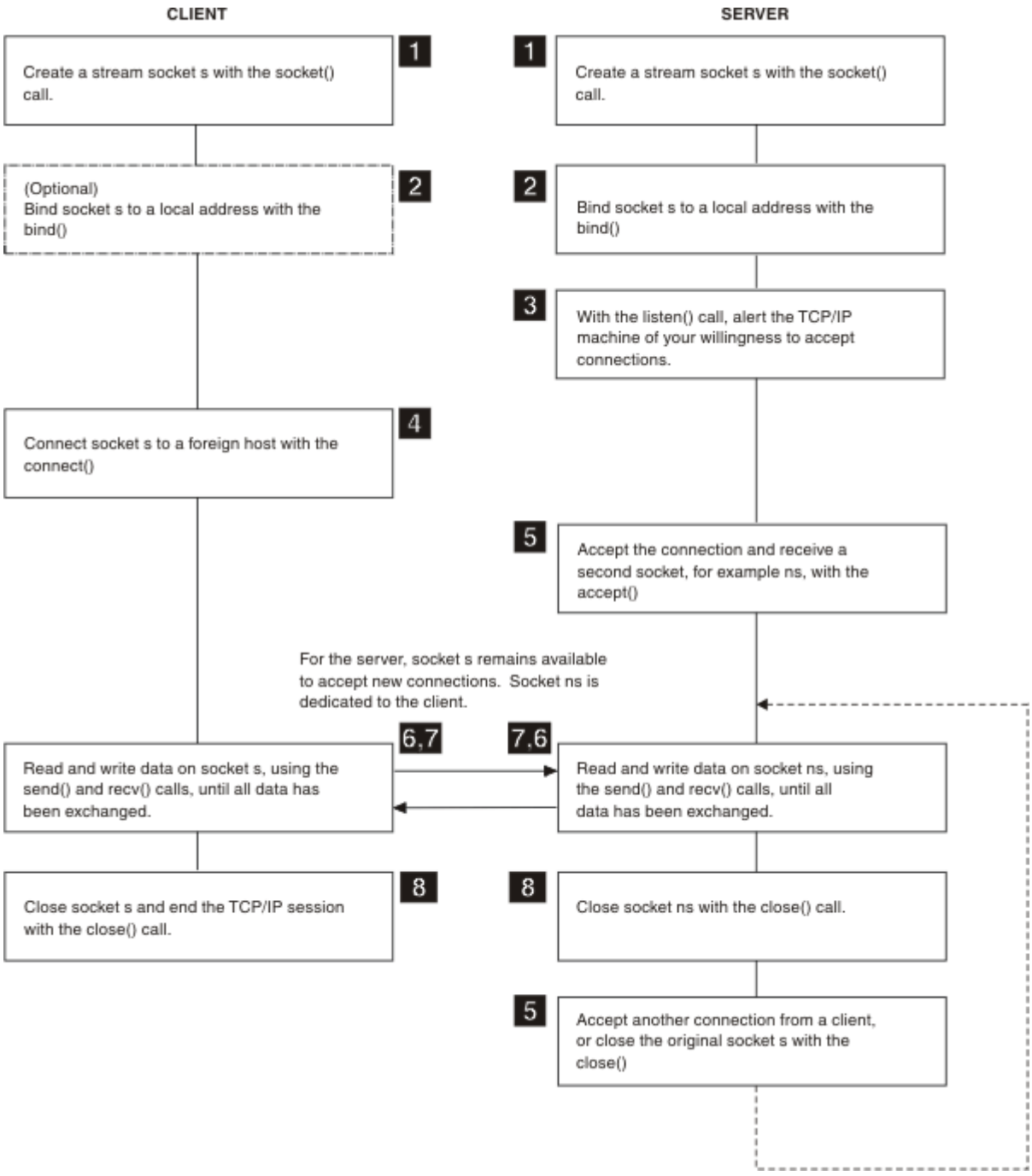


Figure7: Program flow of Client and Server python programs

System Specification and Parameters: In this seminar, Raspberry Pi 4 Model B is used, the technical specification concerning WLAN and under the test condition are

WLAN: 802.11ac; Uplink: 10Mbit/s; Downlink: 50Mbit/s

2.4 Level 2 Monitoring – Failure and Failsafe

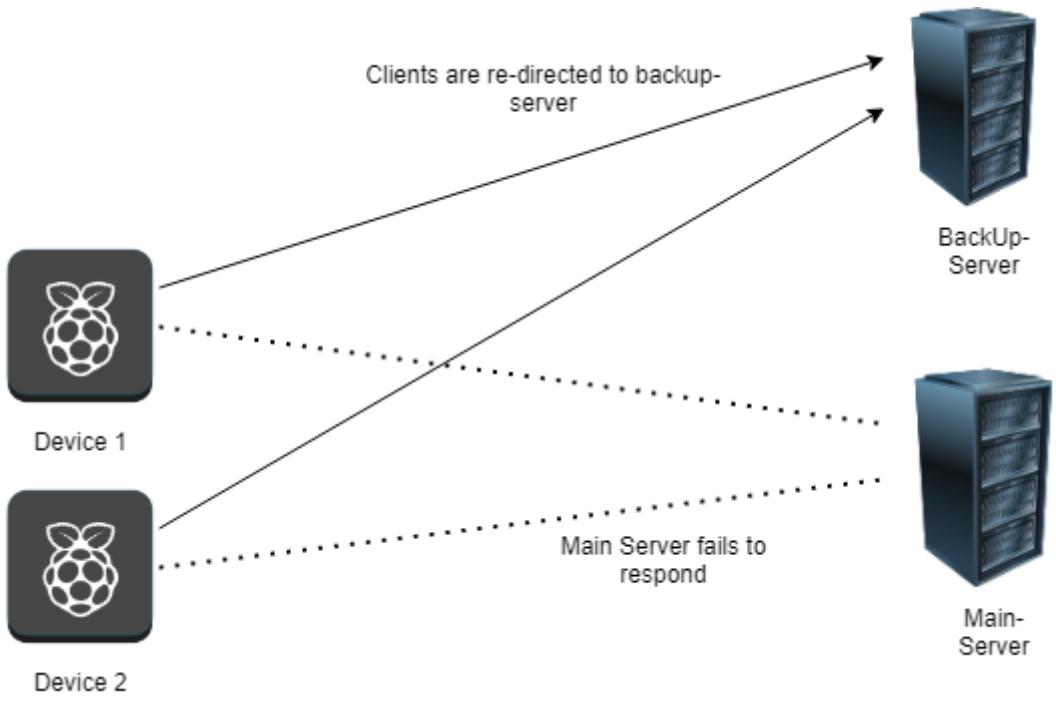
If the server fails to respond to the client then the situation of no response is debounced to a particular time as specified in the TCP/IP stack.

If even after the debounced period the situation of no error persists then one of the clients takes over and becomes the main server.

This is done based on the priority ID assigned to each client. The client with the highest priority number takes over as the main server.

Clients are re-directed to backup server if there is no response from the main server. All the errors from the main server sockets are handled and these exceptions are logged. With any exception leading to un-successful transmission between client and server is overcome by connecting to backup server and continuing the transmission action.

Thus, the problem of a node going missing is resolved using backup designated nodes.



3. Results and Outcome

The results and outcome of the client-server setup are as intended. The set of data that are transmitted from the client are received intact. The entire setup is logged using packet sniffer Wireshark.

The number of packets transmitted depends on the traffic on the network. In Figure8 we can see the number of packets transmitted from client to the server with respect to time in seconds.

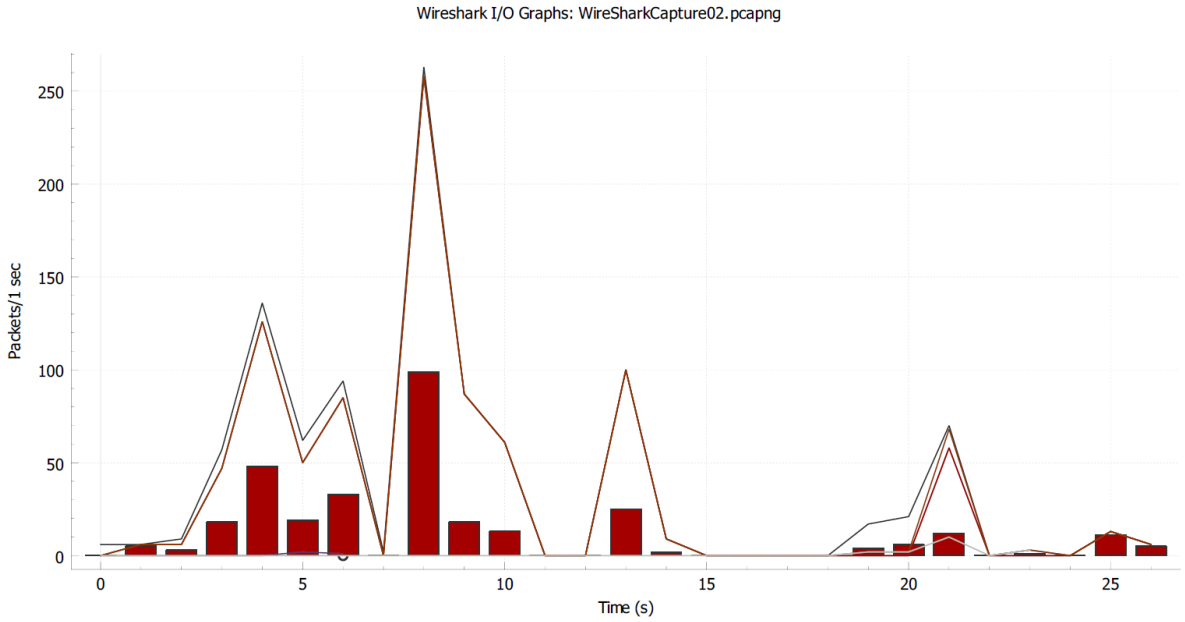


Figure8: The number of packets/second transmitted from client to the server with respect to time in seconds

In one of the client-server combinations out of all 3 raspberry pis, the server IP address is 192.168.1.14 with port 1200 and the client IP is 192.168.1.249. The client-server communication can be seen clearly in the Figure9.

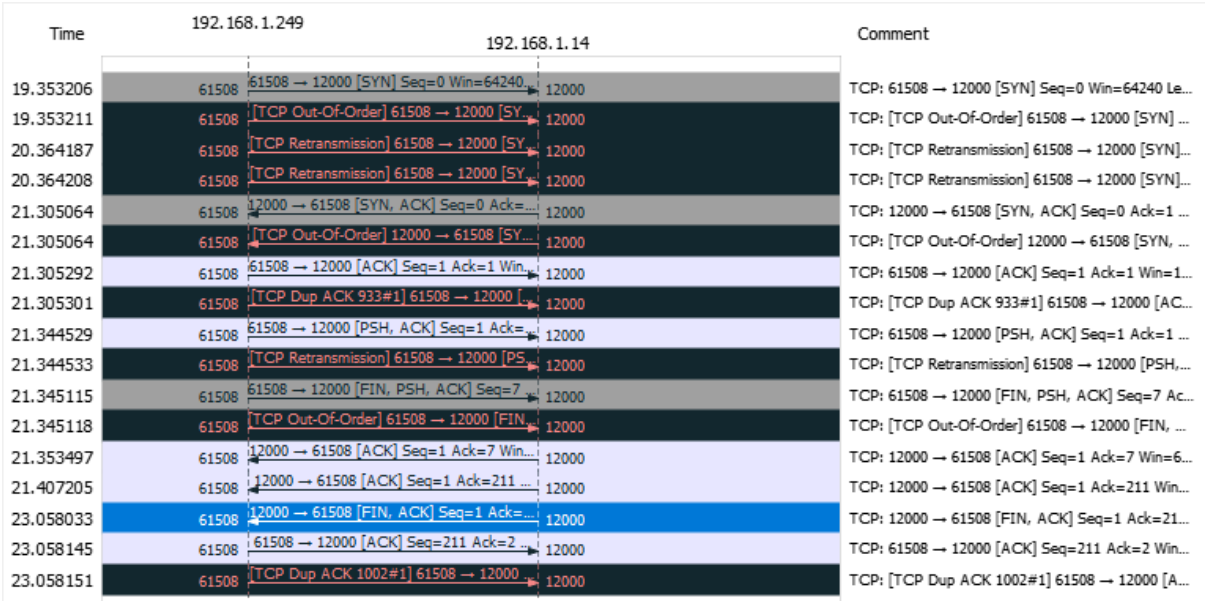


Figure9: Client-Server communication with TCP

The decision of choosing TCP over UDP is also justified here as no packet is lost and all the packets which are lost are retransmitted.

The data transmitted from the client and received at the server are evaluated and they are identical to each other. There are 4 sets of data which are transmitted and they are Time, Power Rating of Device 1, Power Rating of Device 2 and Power Rating of Device 3. These devices are household devices. The client obtained these data from the sensors and the devices themselves. These data are then logged on to the excel sheet provided by the client. These data are then plotted for better understanding which can be seen below.

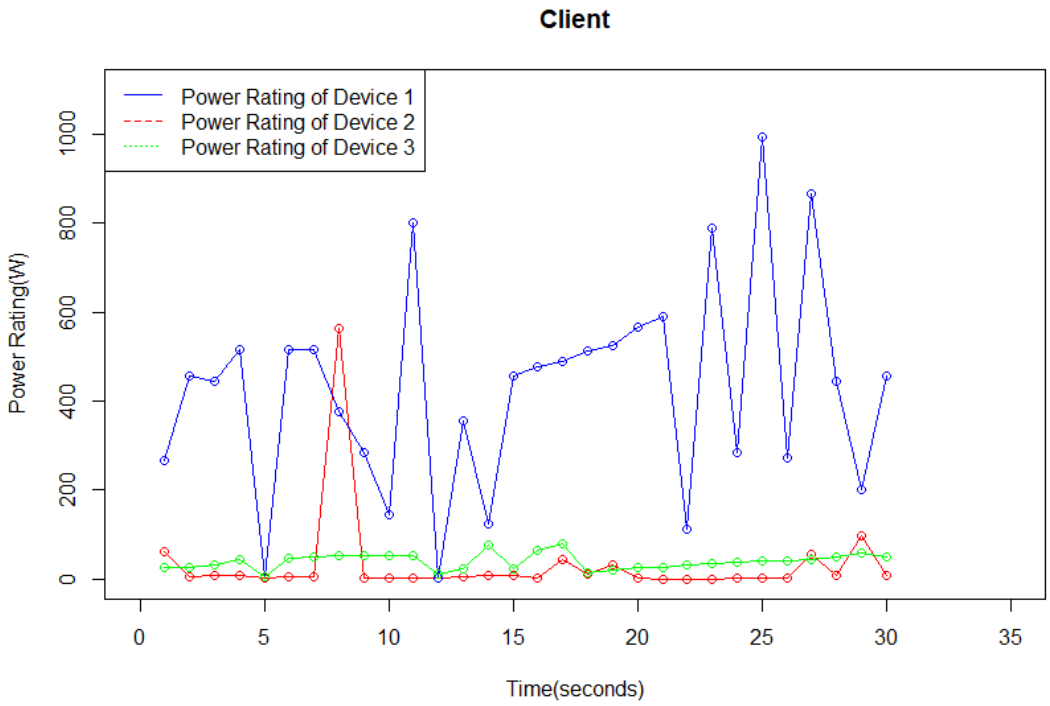


Figure10: Power rating of various devices which are transmitted from the client

Figure11: Power rating of various devices which are received at the server

From the above 2 plots it is clear that they are identical and all the data transmitted is received at the receiver.

4. Resources

All the necessary documents concerning code and resources are placed in the GitHub repository.

Anirudh Upadhya, GitHub - https://github.com/anirudh-upadhya/AS2020_Seminar

5. References

[1] Keith W. Ross and James F. Kurose Retrieved August 14th 2020 from https://www.net.t-labs.tu-berlin.de/teaching/computer_networking/02.06.htm

[2] Mustafa Ali Retrieved August 14th 2020 from https://www.fieldengineer.com/blogs/what-is-firewall-important-network-security

[3] MiMove Retrieved August 14th 2020 from https://mimove.inria.fr/zefxis/

[4] Jianhua Zhang, Adarsh Hasandka, Jin Wei, S. M. Shafiul Alam, Tarek Elgindy, Anthony R. Florita 1 and Bri-Mathias Hodge Retrieved August 14th 2020 from https://www.mdpi.com/1996-1073/11/4/871

[5] Xiaodong Yang, Yanyan Wang, Youbing Zhang, Denghui Xu, Retrieved August 14th 2020 from https://ieeexplore.ieee.org/document/8582157

[6] Gaia Maselli, Mauro Piva, and John A. Stankovic, Retrieved July 14th 2020 from https://ieeexplore.ieee.org/document/8698889

[7] Silvia Marzal, Raul González-Medina, Robert Salas-Puente, Gabriel Garcerá and Emilio Figueres, Retrieved July 14th 2020 from https://ieeexplore.ieee.org/document/8708194

[8] M. Kuzlu, M. Pipattanasomporn and S. Rahman, Retrieved July 14th 2020 from https://ieeexplore.ieee.org/document/7437036

[9] Charalampos Kalalas, Francisco Vazquez-Gallego and Jesus Alonso-Zarate, Retrieved July 14th 2020 from https://ieeexplore.ieee.org/document/7778794

[10] Hyoung-Jun Park and Dongik Lee, Retrieved July 14th 2020 from https://www.mdpi.com/2079-9292/9/1/170

[11] Ikbal Ali, and S. M. Suhail Hussain, Retrieved July 14th 2020 from https://ieeexplore.ieee.org/document/7560616

[12] V. G. Tharinda Nishantha Vidanagama, Daisuke Arai and Tomohiko Ogishi, Retrieved July 14th 2020 from https://ieeexplore.ieee.org/document/7112092

[13] S. K. Datta, C. Bonnet, and N. Nikaein, ‘‘An IoT gateway centric architecture to provide novel M2M services,’’ in Proc. IEEE World Forum Internet Things (WF-IoT), Mar. 2014, pp. 514–519.

[14] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. D. Johnson, ‘‘M2M: From mobile to embedded Internet,’’ IEEE Commun. Mag., vol. 49, no. 4, pp. 36–43, Apr. 2011.

[15] Karnaouskos, S. The Cooperative Internet of Things enabled Smart Grid. In Proceedings of the 14th IEEE International Symposium on Consumer Electronics (ISCE), Braunschweig, Germany, 7–10 June 2010.

[16] Cataliotti, A.; Cosentino, V.; di Cara, D.; Guaiana, S.; Panzavecchia, N.; Tin, G.; Gallo, D.; Landi, C.; Landi, M.; Luiso, M. Experimental Evaluation of an Hybrid Communication System Architecture for Smart Grid Application. In Proceedings of the IEEE International Workshop on Applied Measurements for Power Systems (AMPS), Aachen, Germany, 23–25 September 2015; pp. 96–101.

[17] Rajalingham, G.; Ho, Q.-D.; Le-Ngoc, T. Evaluation of an Efficient Smart Grid Communication System at the Neighbor Area Level. In Proceedings of the 11th IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA, 10–13 January 2014; pp. 426–431.

[18] Gordon McMillan, Retrieved July 14th 2020 from https://docs.python.org/3/howto/sockets.html

[19] Budka, K.; Deshpande, J.; Thottan, M. Communication Networks for Smart Grids Making Smart Grid Real, 1st ed. Springer: London, UK, 2014.

[20] Nikhale, S.; Mankar, C.; Auti, D. Implementation of 802.16 using NS-3 Simulator. In Proceedings of the IEEE 9th International Conference on Intelligent Systems and Control (ISCO), Coimbatore, India, 9–10 January 2015.

[21] Schneider, K.; Chen, Y.; Chassin, D.; Pratt, R.; Engel, D.; Thompson, S. Modern Grid Initiative: Distribution Taxonomy Final Report; Pacific Northwest National Laboratory: Richland, WA, USA, 2008.

[22] Doug Hellmann, Retrieved July 14th 2020 from https://pymotw.com/2/socket/tcp.html

[23] Salvadori, F.; Gehrke, C.S.; de Oliveira, A.C.; de Campos, M.; Sausen, P.S. Smart Grid Infrastructure Using a Hybrid Network Architecture. IEEE Trans. Smart Grid 2013, 4, 1630–1639.

[24] S. Shukla, Y. Deng, S. Shukla, and L. Mili, "Construction of a microgrid communication network," in Innovative Smart Grid Technologies Conference, 2014, pp. 1–5.

[25] IBM Knowledge Centre, Retrieved July 14th 2020 from https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.4.0/com.ibm.zos.v2r4.halc001/o4ag1.htm

[26] Wireshark User's Guide, Retrieved July 14th 2020 from https://www.wireshark.org/docs/wsug_html/

[27] Plotting graph using R, Retrieved July 14th 2020 from https://www.rdocumentation.org/packages/graphics/versions/3.6.2/topics/plot