IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 20, Issue 5, Ver. III (Sep - Oct 2018), PP 54-63
www.iosrjournals.org

Establishing Point to Point Connections Among IoT Devices Over the Internet

Dr. Sefer Kurnaz[1],Abdullah Abed[2]
Corresponding Author: Dr. Sefer Kurnaz

Abstract: - *The rapid growth in the number of devices connected to the internet, other than computers, has proposed the Internet of Things (IoT) era. These devices make use of the high availability of internet connection to interchange data that enables them of achieving their tasks. However, as these devices usually belong to private networks, with private Internet Protocol (IP) addresses, it is not possible to reach these devices from outside their networks. To overcome this limitation, relay servers are used with public IP address to act like a midpoint in the communications among the IoT devices. This approach increases the path that the data travels through, from one IoT device to another, which increases the time required to reach the destination device and the power consumption at these devices, in addition to the security threat the extension of the path poses to the communications. In this study, a new approach is proposed to establish a direct connection between any two IoT device, using the porthole punching technique. A connections management server is used to detect and forward the socket information that allows one device to connect to another. This socket information is used by the requesting device to connect to the requested one, and communicate the data directly, without any need of the connections management server. The evaluation results show that the proposed method has been able to reduce the time and energy required to communicate the data, compared to the use of relay servers.*

Keywords: - *Cloud Computing, Efficiency, Internet Services,Internet of Things.*

Date of Submission: 13-10-2018 Date of acceptance: 28-10-2018

I. Introduction

Nowadays, the devices connected to the internet are not only computers and smartphones. Most of the modern devices have gained access to the internet send, receive data or both. Received data can be used for command execution, such as turning a light on or off, while transmission may be of data sensed or measured by the device, such as ambient temperature [1]. Each device on the network required an address, in order to send and receive data. These addresses as called Internet Protocol (IP) addresses, where each IP address consists of four bytes that represents the address of that device on the network. The size of the IP address limits the number of possible addresses [2].

To overcome the problem of the limited number of IP addresses, the available addresses are split into ranges, where each range is assigned to a specific usage. Two of the most important ranges, which are used for data transmission over the internet, are the private and public addresses. A public IP address is a unique IP address over the internet, which can be assigned to only one device over the entire network of the internet, while a private IP address may be assigned to many devices, each device belongs to a different subnet. The idea behind this distribution of the IP addresses is that devices that require remote access should be assigned with public IP address, so that, they can be accessible from anywhere on the internet, as this IP address is unique and the routers on the internet can figure out the route to that IP address. Moreover, to handle the increasing number of devices connected to the internet, private IP addresses are assigned to devices that need to access the internet but are not required to be accessed from it. Thus, devices that provide services over the internet are assigned with public IP address, so that, they can be reachable from any device connected to the internet, while client devices, such as computers and smartphones, may use private IP addresses, as no remote access required for these devices [3].

As it is possible to provide more than one service per each server, and it is also possible for each client to make use of more than one service on the server, it is important to distinguish the required service that the data is being sent for, or, the service that the client is trying to access on that server. For this purpose, each service uses a different virtual port, which is distinguished from other ports by assigning a different port number for each service being provided. Thus, to make a connection to a service on the server, the client needs to know the IP address of the server and the port number that the service is listening for connections on it. The combination of the IP address and the port number is known as the socket, where connections are made based on sockets [4].

Data transmission on the internet is usually bidirectional, where data is sent back and forth between the client and the server, even if the main aim of the connection is to send data in one direction. In Transmission Control Protocols (TCP), each received packet is acknowledged to the sender, so that, if no acknowledgment is received, the packet is transmitted again, as the delivery of the packet is considered failure. Thus, it is important for the server to know the route back to the clients IP address, even if the client has a private IP address. To do so, when a connection is established between a public and a private IP addresses, the device that represents the gateway of the private network to the public internet, which has at least one public IP address, maps the socket information of the device with the private address into a new socket, which has the public IP address of that gateway, so that, when the device with the public IP address sends information back to the client, it sends to the new socket and the device forwards it to the device that this socket is assigned for [5].

The only problem with private IP addresses is that devices with these addresses cannot receive connections from other devices on the internet, because if a connection is initiated toward a device with a private IP address, it is impossible to distinguish the required device among all the devices that have the same private IP address. To overcome such problem, it is possible to setup port forwarding, which represents a static Network Address Translation (NAT) rule that maps any incoming connections to the public IP address of the gateway on a specific port to the required device with the private IP address and the port setup in the rule [6].

Regardless of the type of the IP address assigned to a device, whether it is a public or private IP address, these IP address may be static or dynamic. Static IP addresses are assigned to specific devices and do not change under any circumstances, as they are added to the device. Dynamic IP addresses may change every time the device is connected to the network, as these addresses are assigned by specific servers on the network, using a protocol known as Dynamic Host Configuration Protocol (DHCP). Whenever the device loses connectivity to the network, it requests IP configurations from the DHCP server, which may or may not assign the same IP address assigned earlier to this device. This imposes the problem that devices on the network, as well as the internet, may not have the same IP address over different periods of time. These configurations are, sometimes, set by the internet service provider, who may apply extra fees to assign a static IP address [7].

II. Literature Review

Direct communication between devices connected to the internet is essential to ensure fast interchange of data, instead of using a proxy server that represents a midpoint between the communicating devices, where all the data must pass through that server in order to connect devices [8]. Although it is easier to implement communication using the proxy method, the fact that all data must go through that server slows down the communication speed, as it is limited by the network speed of the proxy server, and the amount of data being handled by the server from all the devices at that moment. Another disadvantage of using such topology id the security of the information being transferred, as all this information must go through that server in order to be delivered to the other party of the connection [9]. To overcome the security issues, it is possible to encrypt the data being transferred, so that, the proxy server does not have access to the actual contents of the information being transferred [10]. But, data encryption requires a lot of the limited available resource of the Internet of Things (IoT) devices.

Different techniques are implemented to achieve direct interchange of information between devices on the internet. These techniques ensure faster and more secure data transfer using different protocols depending on the requirements of the application that the connection is being made for. Peer to Peer (P2P) protocol is used by Voice Over IP (VOIP) applications, to setup a direct connection between devices to ensure faster data transfer in order to maintain the voice communication quality [11]. Some other applications use Virtual Private Networks (VPN), which are created between specific computers in a larger network to establish their own network for data transmission. These networks, mainly, use the Secure Socket Tunneling Protocol (SSTP) and Internet Key Exchange Version 2 (IKEv2) [12]. All these methods are difficult and expensive, from the resources point of view, to be implemented in IoT devices.

Each device on a network has a unique address, which is the IP address. However, a device may communicate with multiple devices simultaneously or with different services on the same device. Thus, sending information by only using the IP address does not specify the application of that device that the information is sent for. To overcome this problem, each application that requires access to the network is assigned with a port number, so that, when information is sent to a specified port number on the device, it is possible to redirect that information to the required application. The combination of the IP address and the port number is known as the socket. Applications that provide services on the network are assigned with a static port number, so that, when any other device tends to access that service, the port number that the service is listening on is known. However, the application that initiates the connection must have a port number, so that, the information sent back to the application, if exists, is assigned to that port number. These applications are the host applications and are usually assigned with dynamic port numbers. Each party in a connection knows the socket information of the other party, automatically [13].

Porthole punching is another approach to achieve direct connections between devices on the internet. Both devices are connected to a known server, with a known public IP address and port number, then, the socket information of the remote clients, from the server's point of view, are forwarded to the other device. As this socket information are already configured in the NAT configurations of the remote devices, these devices become reachable from different addresses. It is easier to implement port hole punching in services built over UDP protocol, as it uses simpler data encapsulation than in the TCP protocol, but it is possible to lose some of the information being transferred over the UDP protocol, as it requires no acknowledgment upon data arrival to the other end of the connection. Thus, in case of transmitting important information using the UDP protocol, it is important to take care of that issue by adopting other protocols that are built over the UDP protocol [14].

Figure 2.1 illustrates the basic steps of the porthole punching.

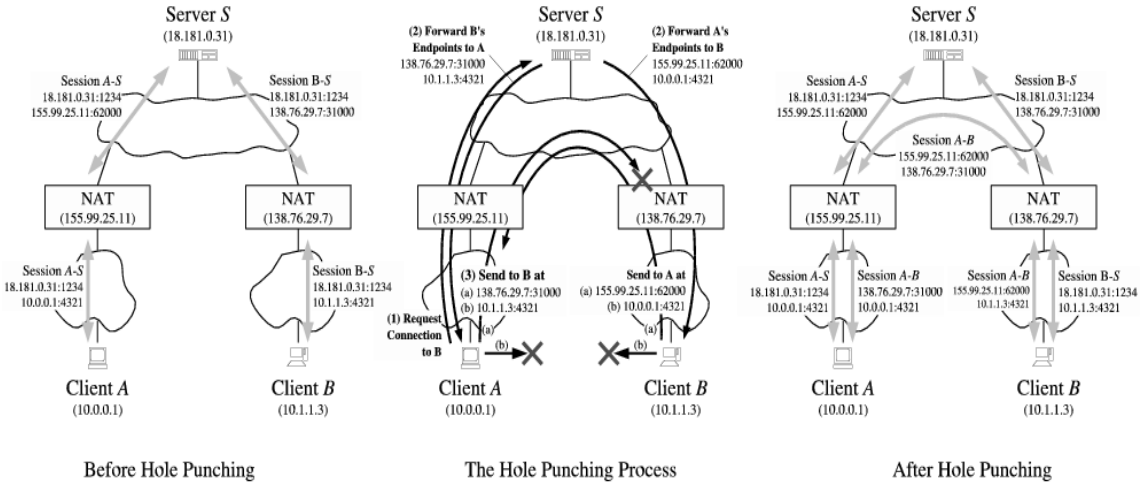

Figure 1: Porthole punching procedure [15].

TCP guarantees the recipient will receive the packets in order by numbering them. The recipient sends messages back to the sender saying it received the messages. If the sender does not get a correct response, it will resend the packets to ensure the recipient received them. Packets are also checked for errors. Packets sent with TCP are tracked so no data is lost or corrupted in transit. Moreover, UDP just sends data to the recipient and does not wait to make sure the recipient received the packet and just continues sending the next packets. If the recipient device misses some UDP packets, it cannot ask for those packets again. There is no guarantee that the recipient is getting all the packets and there is no way to ask for a packet again if it misses it but losing all this overhead means the computers can communicate more quickly.

III. Proposed Method

The proposed method includes a service that runs on a server with a known public IP address, and uses a specific port number, so that, all IoT devices can connect to this service. Each IoT device is assigned with an identification number, so that, it can be distinguished from other devices, regardless of its network configuration, which may vary over time. When an IoT device is connected to the service, it sends its local IP address and port number, so that, it is easier for IoT devices on the same subnet to reach it using this information. Moreover, the socket information of the device is known to the service, so that, if an IoT device from another network attempts to reach that device, this information can be used to access it. Each device must have an active connection to the service, so that, when a new connection is required to that device, it initiates a new connection to the service. The port of the new connection is punched by the device that requested the connection. Different approaches are evaluated in this study, such as the use of private socket information, in addition to the public to establish the communications, as well as implementing the IEC 870-5-101 protocol in the application layer, to ensure the arrival of the data in the same order they are sent.Algorithm 1 describes the main steps of the proposed method to achieve a point to point connection between two IoT devices.

Algorithm 1: Establishing point to point connection between IoT.

Algorithm: IoT point to point connection establishment.//This algorithm creates connections between IoT devices over the internet. *X* variable represents the requested IoT device, while *y* is the requesting device. //All devices are required to maintain a connection to the connections management server (CMS) to exchange information and commands. **Input:** Connection request to IoT device with ID=*x* from IoT device with ID=*y*. These devices are expected to be connected to the server using connections denoted as C_x and C_y.	
Step1:	*//Initialization* *Socket* as string*//To store the socket information of the requested device, to be sent to the requesting one.* *//Check whether the requested device is connected to the connection management server.* If *x* is reachable through C_x String AS ← random unique string.
Step2:	*//Instruct the requested device to initiate a new connection using the existing connection* Send a command to device*x* to initiate a new connection to the CMS on any available port (*P*) on the CMS using AS for authentication.
Step3:	*//Wait for the requested device to establish a new connection with the CMS using the new port assigned by the CMS.* While (no new connection from *x* on port *P*) Wait for new connection from *x* on port *P*.
Step4:	*//Read the remote socket information of the new connection at the server on port P.* *Socket* ← Read the remote socket information for the connection established over port *P*. Validate the *Socket* and
Step5:	//Send this information from the CMS to the device requesting the connection. Send *Socket* to the IoT device *y* using C_y.
Step6:	*//The new connection is now being used to communicate with the other IoT device. Thus, the CMS server no longer needs to maintain that connection.* Terminate the new connection initiated by *x* on port *P* from the server side.
Step 7:	Else*://Inform the requesting device that the requested device is not reachable.* Send "*Unreachable"* to device*y* through C_y.

The main difference between the proposed method and the use of the proxy server is that the information exchanged between two IoT devices that use the proposed method are exchanged directly and the need of the server is only mandatory to establish the direct connection between the devices. After the connection is established, no further need of the server exists. However, proxy server receives every bit of the information being sent from one IoT device sot another, stored temporarily, and then forwarded to the receiving device. Thus, instead of sending the information directly from one point to another, all the traffic must be directed to another point, which is the proxy server. This addition affects both the security and exchange speed of the information being transmitted from one IoT device to another. As shown in Figure 2, each IoT device is required to maintain a connection with the connections management server, so that, requests for new connections can be sent to the device using this connection. This connection is used to only send commands from the CMS to the IoT device and connection requests from the IoT devices to the CMS, where no data are sent over it. As soon as a device receives a request for a new connection, it initiates one to the connections management server. The socket information is known to the server, which are normally used to send information back using the same connection. Thus, this information is retrieved and forwarded to the IoT device requesting the connection, i.e., device *y* in the example. As soon the IoT device that requested the connection receives the connection information, it can contact the requested device directly, creating a point to point connection to send the intended data.

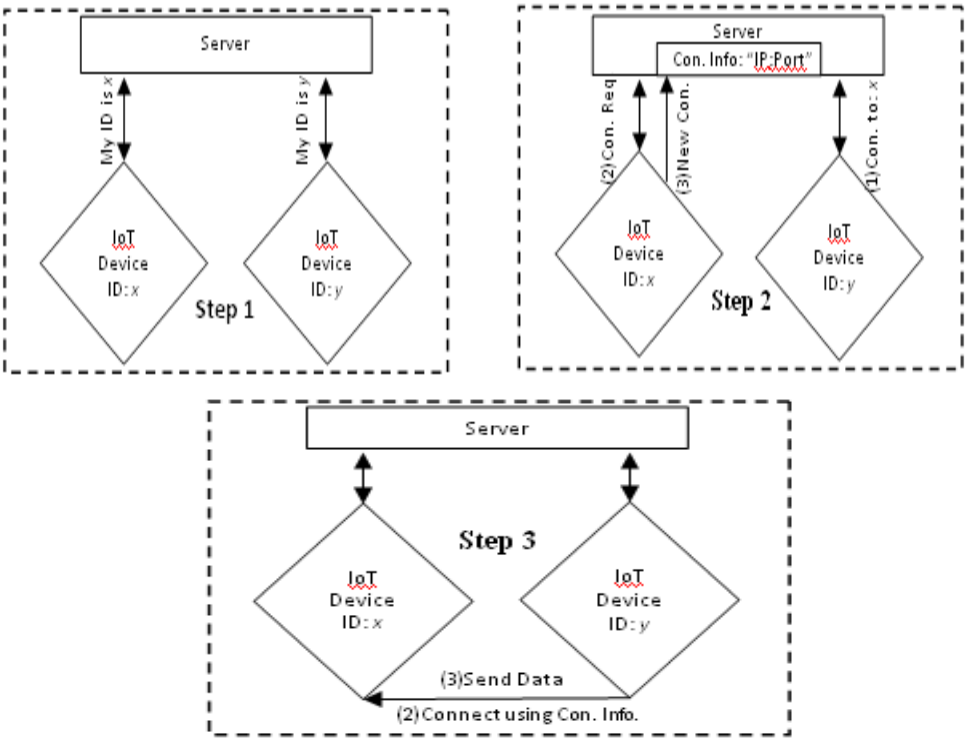

Figure 2: Illustration of the steps required to establish a direct connection between IoT devices.

The use of proxy, or relay, servers require the IoT devices to send their data to that server as a midpoint that both devices can reach. The relay server must have a public IP address to make it reachable by all the devices connected to the internet, so that, when a device needs to communicate with another, the data are sent to the relay server in order to be delivered to the destination device. The addition of the relay server increases the path that the data passes through in order to reach the destination device, as well as the security concerns that the use of such server imposes, as all data must travel through, and revealed to, the proxy server to reach their destination.

As illustrated in the previous section, an IoT device can establish a direct connection to another device in two steps, using the proposed method. After connecting to the connections management server, an IoT device can request connection information of the other IoT device and use it to send data of any size, without the need of that server. However, using a proxy server, the data being transferred must always be sent to the proxy server, which redirects these data to the destination IoT device. Thus, if the proxy server is lost, all communication among IoT devices are terminated immediately, while using the proposed method, the connections management server is not needed, after establishing a direct connection, unless the IP address of one of the IoT devices, connected to each other, is changed.

To illustrate the difference between the proposed method and the existing proxy method, algorithm 2 shows the steps required to send a set of packets from device *x* to device *y*, from the IoT device's point of view, while algorithm 3 show the steps required from the IoT device to send the same set of packets using the existing proxy method.

Algorithm 2: Sending data packets from IoT device *x* to IoT device *y* using port punching.	
Step1:	*//Send a new connection request for the required device to the CMS server* *Info* ← Request connection information of device (*y*) from the CMS.
Step2:	*//Connect to the requested device using the information received from the CMS.* Connect to IoT device *y* using *Info*.
Step3:	*//Send all the data directly to the requested device using the connection established using the socket information retrieved from the CMS, without any further communications to the CMS.* For each packet in the packets set: Send the packet directly to *y*.
Step4:	*//Close the connection to the requested device.* Terminate connection with *y*.

Algorithm 3: Sending data packets from IoT device *x* to IoT device *y* using Proxy Server.	
Step1:	Connect to the proxy server
Step2:	//Sent data one by one to the proxy server in order to deliver them to the remote IoT device. For each packet in the packets set: Specify the destination device *y* to the proxy server Send the packet to the proxy server If confirmation is required: Wait for delivery confirmation from the proxy server.
Step3:	//After all data is sent, the connection is no longer required. Terminate it. Terminate connection with the proxy server.

Moreover, the path that the data travel through, when a proxy server is used, is equal to the summation of the path from the transmitter to the proxy server and the path from the proxy server to the receiver, even if the devices are located in the same network. Using the proposed method, the path is minimized to the minimum number of hops that directly connect the public IP addresses the connect these devices to the internet, or directly if the devices are located in the same network. Moreover, the packets communicated using the proxy server are queued at the proxy server, to be processed according to their designated destination, which imposes extra delay to the delivery of these packets, especially when the proxy server is handling an enormous number of IoT devices.

Another important advantage of the proposed method is the security of the information being transferred. As this information is communicated directly between the sender and the receiver IoT devices, no information is revealed to the connections management server, while each information communicated between two IoT devices using the proxy server must be revealed, stored and redirected by the proxy server. Moreover, the minimization of the path that the information travels through, to reach from the sender to the receiver, minimizes the risk of external attacks, where attackers may listen to the communication between the IoT devices.

For example, if two devices in the same network intend to exchange information without knowing each other's IP address, the use of proxy server requires each packet to reach to the proxy server to be redirected back to the receiving device, where losing connectivity with the proxy server terminates all the communication between these device, despite that each device is reachable by the other one. However, using the proposed method, the connections management server is no longer needed after the direct connection is established between the devices, and losing connectivity to that server does not affect the communications between the IoT devices as long as the connections information is the same for both devices.

IV. Experimental Results

In order to evaluate the performance of the IoT devices when using the proposed method to send data from one to another, the existing method, of using relay server, is implemented to use the collected data as reference measurements for comparisons. The energy and time consumed by the IoT device to send the data to another is measures as well as the sent, received and total data communicated by the sender in order to deliver the payload data to the destination device.

USING RELAY SERVER

In this experiment, the existing method of using a relay, also known as proxy, server to exchange information between two IoT devices is evaluated. The topology used in this experiment is shown in Figure 3, where the results of this experiment are used as control values for comparison with the results of the proposed methods. These results are summarized in Table 1.

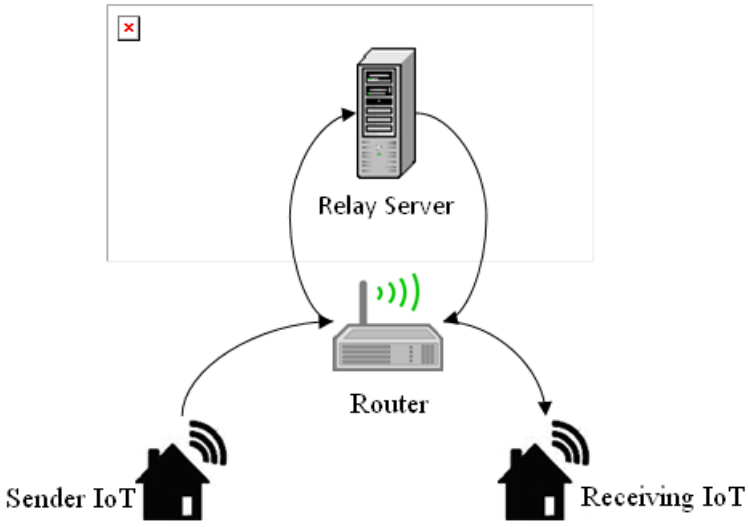

Figure 3: Illustration of the topology used to transfer data from one IoT device to another using a relay server [16].

Table 1: Measurements of IoT device transferring data using a relay server.

	Data (Bytes)				
	Sent	**Received**	**Total**	**Energy(mWh)**	**Time (Sec)**
1K	1863	572	2435	0.734	6.478
100K	103615	2403	106018	3.886	44.244
1M	2E+06	14538	2E+06	12.716	122.34

PORT PUNCHING USING PUBLIC IP ADDRESS

In this experiment, the port punching is achieved using the socket information retrieved from the connection at the Connections Management Server (CMS) side, i.e., no information is available about the local socket information of the IoT device. Thus, even if the sending and receiving IoT devices belong to the same private subnet, the communications are achieved through the gateway of the network to the internet, i.e., the default router that connects the local network to the internet. In this case, if the gateway loses the public IP address, after a connection reset, for example, the communications are interrupted and the information, of how to reach the destination server, needs to be retrieved from the CMS again. Figure 4 shows the topology used in the second step of this experiment, where the first step, of retrieving the destination device's information from the CMS server, is executed once.

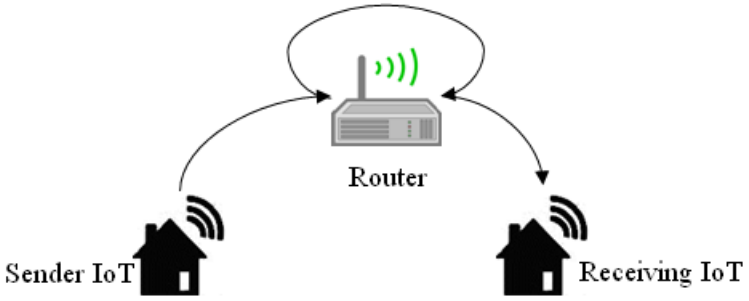

Figure 4: Illustration of the topology used to transfer using a public socket port punching server.

The measurements of IoT device using this topology are shown in Table 2.

Table 2: Measurements of IoT device transferring data using port punching on the public socket.

	Data (Bytes)				
	Sent	**Received**	**Total**	**Energy(mWh)**	**Time (Sec)**
1K	1161	24	1185	0.53	3.15
100K	102537	24	102561	2.33	26.12
1M	1E+06	24	1E+06	6.36	61.07

Moreover, as the port punching relies on UDP protocol, which does not have the ability to detect any packets lost in the transmission, the IEC 870-5-101 protocol is implemented over the UDP protocol, so that, any loss in transmitted data can be detected and retransmitted. The results of using this protocol on top of the UDP protocol are shown in Table 3.

Table 3: Measurements of IoT device transferring data using port punching on public socket with IEC protocol (conducted by me).

	Data (Bytes)				
	Sent	**Received**	**Total**	**Energy(mWh)**	**Time (Sec)**
1K	1186	49	1235	0.63	5.21
100K	104947	2434	107381	3.51	35.72
1M	1073386	24696	1098082	10.58	95.18

PORT PUNCHING USING PUBLIC AND PRIVATE IP ADDRESSES

In this experiment, the private socket information of the IoT device is sent to the CMS server, so that, this information becomes known to the CMS server, as they cannot be retrieved by the CMS. This information is forwarded to the requesting device alongside with the public socket information, so that, the requesting device attempts to connect to the destination device using the local socket information first, as shown in Figure 5. If the requested device is not reachable using this information, the public socket information is used to establish the connection.

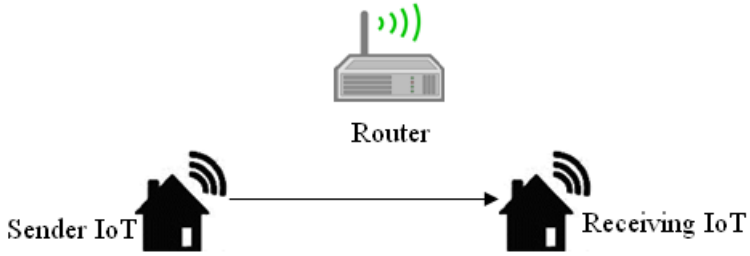

Figure 5: Illustration of the topology used to transfer using a private socket port punching server (proposed by me).

The results of this experiment are shown in Table 4.

Table 4: Measurements of IoT device transferring data using port punching on private socket (conducted by me).

	Data (Bytes)				
	Sent	**Received**	**Total**	**Energy(mWh)**	**Time (Sec)**
1K	0.17	1.48	1161	38	1199
100K	0.81	11.08	102537	38	102575
1M	3.2	24.87	1E+06	38	1E+06

Again, the implementation of IEC 870-5-101 communication protocol is used on top of the UDP protocol to detect any data lost in the network. The results of this implementation are summarized in Table 5.

Table 5: Measurements of IoT device transferring data using port punching on private socket with IEC protocol (conducted by me).

	Data (Bytes)				
	Sent	**Received**	**Total**	**Energy(mWh)**	**Time (Sec)**
1K	1186	63	1249	0.24	1.71
100K	104947	2448	107395	1.31	13.68
1M	1073386	24710	1098096	4.25	38.97

V. Conclusion

Most of the modern devices are connected to the internet. Some of these devices need to interchange data among them. Some of the data exchange must be fast and secure, which requires a direct connection between the communicating devices. The use of proxy servers reduces the speed and endangers the security of the information being transferred while most of those devices have private IP addresses that cannot be reached from outside the network, which makes it impossible to initiate direct connections without any relaying point.

In this study, a method that allows IoT devices to connect to each other in order to exchange information among them without the use of any relay or proxy servers. This approach allows faster and more secure communication, as the data being transferred is going directly from the sender to the receiver IoT devices. Moreover, the resource required for the server that handles the connections' establishments are expected to be extremely low, compared to those required for a proxy server. Thus, it is possible to use a low-cost cloud server to handle an enormous number of IoT devices, as none of the information being transferred pass through the server, and only tiny pieces of information are transferred through the server, which is the connections information of the IoT devices.

The results of the experiments conducted using the existing methods, based on relay servers, and the proposed method using two approaches, the uses the public IP address of the router to loop back the communications, while in the other approach the requested IoT device send its local socket information, so that, the transmitting device attempts to connect using this information to use the local network, if both devices are located in the same network. The results show that the use of the proposed method has improved the performance of the IoT devices by reducing the time and energy required to transfer a certain amount of data from one IoT device to another. Moreover, the results also show that the use of private network between devices that belong to the same network has also better performance that using the network's gateway's public IP address to loop the data back to the receiver. Moreover, the use of IEC870-101-5 protocol to detect any error that may occur during communication has less effect over the performance of the IoT devices when they are located in the network and communicate directly, according to the reduction in the time required to send data and acknowledgments back and forth between the communicating devices.

In future work, different encryption techniques are going to be implemented in order to evaluate the performance of the IoT devices when communicate encrypted data using the proposed method. Such approach adds another security layer to the communications among these devices, so that, attackers who may acquire the communicated data cannot interpret the actual information being transferred.

References

[1]. R. Want, B. N. Schilit, and S. Jenson, "Enabling the internet of things," *Computer*, vol. 48, pp. 28-35, 2015.
[2]. H. Zisimopoulos and H. Liang, "Network entity, a wireless communication unit and methods for access to a remote private IP network and supporting thereof," ed: Google Patents, 2018.
[3]. Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear, "Address allocation for private internets," 2070-1721, 1996.
[4]. M. Holdrege and P. Srisuresh, "Protocol complications with the IP network address translator," 2070-1721, 2000.
[5]. M. Hassan and R. Jain, *High performance TCP/IP networking*: Prentice Hall, 2003.
[6]. M. Casado, T. Koponen, R. Ramanathan, and S. Shenker, "Virtualizing the network forwarding plane," in *Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow*, 2010, p. 8.
[7]. Y. Sun and E. M. Belding-Royer, "Dynamic address configuration in mobile ad hoc networks," 2003.
[8]. H. Asmat and S. Ullah, "The Impact of Existing and Future Mobile Technologies on Pakistan: A Survey," *International Journal of Future Computer and Communication*, vol. 4, p. 254, 2015.
[9]. A. Sgora, D. D. Vergados, and P. Chatzimisios, "A survey on security and privacy issues in wireless mesh networks," *Security and Communication Networks*, vol. 9, pp. 1877-1889, 2016.
[10]. E. C. Munger, V. J. Sabio, R. D. Short III, V. D. Gligor, and D. C. Schmidt, "Agile network protocol for secure communications with assured system availability," ed: Google Patents, 2016.
[11]. B. Goode, "Voice over internet protocol (VoIP)," *Proceedings of the IEEE*, vol. 90, pp. 1495-1517, 2002.
[12]. J. B. R. Lawas, A. C. Vivero, and A. Sharma, "Network performance evaluation of VPN protocols (SSTP and IKEv2)," in *Wireless and Optical Communications Networks (WOCN), 2016 Thirteenth International Conference on*, 2016, pp. 1-5.
[13]. C. Y. Kin, "Virtual network mechanism to access well known port application programs running on a single host system," ed: Google Patents, 1997.
[14]. R. L. McGuire, "Low-latency hole punching," ed: Google Patents, 2015.

[15]. B. Ford, P. Srisuresh, and D. Kegel, "Peer-to-Peer Communication Across Network Address Translators," in *USENIX Annual Technical Conference, General Track*, 2005, pp. 179-192.
[16]. C. Stergiou, K. E. Psannis, B.-G. Kim, and B. Gupta, "Secure integration of IoT and cloud computing," *Future Generation Computer Systems*, vol. 78, pp. 964-975, 2018.

Dr. Sefer Kurnaz . " Establishing Point to Point Connections Among IoT Devices Over the Internet." IOSR Journal of Computer Engineering (IOSR-JCE) 20.5 (2018): 54-63.