Análisis del desempeño del Protocolo de Comunicación XMPP en una Red IOT

Mendoza Velásquez, Hernán Fabricio

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Trabajo de titulación, previo a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones

Ing. Sáenz Enderica, Fabián Gustavo

25 de enero del 2021

URKUND

Document Information

Analyzed document	Tesis_Fabricio_Mendoza_FINAL.docx (D86077468)
Submitted	11/20/2020 3:32:00 PM
Submitted by	
Submitter email	fgsaenz@espe.edu.ec
Similarity	1%
Analysis address	fgsaenz.espe@analysis.urkund.com

Sources included in the report

W	URL: https://docplayer.es/94553206-Universidad-andres-bello-facultad-de-ingenieria-escu ... Fetched: 11/20/2019 7:22:27 PM	1
SA	**Universidad de las Fuerzas Armadas ESPE / Tesis_Fabricio_Mendoza_FINAL.pdf** Document Tesis_Fabricio_Mendoza_FINAL.pdf (D86077558) Submitted by: fgsaenz@espe.edu.ec Receiver: fgsaenz.espe@analysis.urkund.com	2
W	URL: http://www.criptored.upm.es/crypt4you/temas/privacidad-proteccion/leccion3/leccion ... Fetched: 11/20/2020 10:06:00 PM	1

FABIAN GUSTAVO SAENZ ENDERICA

Firmado digitalmente por FABIAN GUSTAVO SAENZ ENDERICA
Fecha: 2020.12.17 23:37:55 -05'00'

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

CERTIFICACIÓN

Certifico que el trabajo de titulación, "Análisis del desempeño del Protocolo de Comunicación XMPP en una Red IOT" fue realizado por el señor Mendoza Velásquez, Hernán Fabricio el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 25 de enero del 2021

Firmado electrónicamente por:
FABIAN GUSTAVO
SAENZ ENDERICA

.....

Ing. Sáenz Enderica, Fabián Gustavo

C.C.: 0102343985

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

RESPONSABILIDAD DE AUTORÍA

Yo, Mendoza Velásquez, Hernán Fabricio, con cédula de ciudadanía n°1750190819, declaro que el contenido, ideas y criterios del trabajo de titulación: "Análisis del desempeño del Protocolo de Comunicación XMPP en una Red IOT" es de mi autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 25 de enero del 2021

Mendoza Velásquez, Hernán Fabricio

C.C.: 1750190819

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

AUTORIZACIÓN DE PUBLICACIÓN

Yo Mendoza Velásquez, Hernán Fabricio, con cédula de ciudadanía n°1750190819, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: "Análisis del desempeño del Protocolo de Comunicación XMPP en una Red IOT" en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Sangolquí, 25 de enero del 2021

Mendoza Velásquez, Hernán Fabricio

C.C.: 1750190819

Dedicatorias

Quiero dedicar este trabajo a Dios, por permitirme culminar esta etapa en mi vida, a pesar de limitaciones y adversidades que se presentaron en el diario vivir, ya que me dio la fortaleza para superar obstáculos y tomar buenas decisiones. También quiero dedicar este logro a mi madre, que día a día me acompañó en este arduo camino con sus consejos y cariño.

HERNÁN FABRICIO MENDOZA VELÁSQUEZ

Agradecimientos

A mi madre, por la confianza y apoyo incondicional que me brindó a lo largo de estos años en esta etapa, siempre supo guiarme de la mejor manera motivándome a no rendirme y seguir adelante con mis metas, sueños y anhelos, de una forma constante y perseverante; los buenos principios y valores que me inculcó me formaron como persona, aprecio todo el esfuerzo que realizó.

A mi hermana, que también estuvo conmigo a lo largo de estos años, indistintamente, dándome ánimos para continuar y no declinar, juntamente con mi madre, fueron los pilares fundamentales en los cuales me apoyé, para seguir adelante, hasta concluir mis estudios en mi formación.

A toda mi familia, que siempre han estado en los buenos y malos momentos, con palabras que me motivaban y consejos que fortalecían mi decisión de terminar con éxito, esta etapa de mis estudios.

A mi estimado tutor Ing. Fabián Sáenz, que confió en mí, para el desarrollo de este proyecto de titulación, con su valiosa guía y asesoramiento, los cuales dieron excelentes resultados.

A mi querida Universidad de las Fuerzas Armadas – ESPE y a todos los docentes que me acompañaron en mi formación académica, gracias por darme la oportunidad de haber crecido como persona y profesional.

HERNÁN FABRICIO MENDOZA VELÁSQUEZ

ÍNDICE

CONTENIDO

Reporte Urkund 2

Certificación 3

Responsabilidad de autoría 4

Autorización de publicación 5

Dedicatorias 6

Agradecimientos 7

Resumen 20

Capítulo I. Introducción 22

Antecedentes 22

Justificación 23

Objetivos 24

General 24

Específicos 24

Capítulo II. Marco Teórico 25

Internet de las Cosas 25

Objetivo de IoT 26

Arquitectura del IoT 27

Características 29

Redes del IoT 30
Seguridad.....36
Protocolos de mensajería instantánea.....38
Arquitectura 39
Protocolos más utilizados 41
Protocolo XMPP.....46
Computadoras de placa única.....54
Aplicación 54
Arquitectura ARM.....56
Raspberry Pi.....57
Capítulo III. Diseño 65
Diseño de la red IoT 65
Partes de la red IoT.....65
Funcionamiento.....66
Clasificación.....68
Diseño utilizado para el proyecto 69
Funcionamiento.....70
Lista de componentes.....71
Configuraciones 72
Configuración de la Raspberry Pi 3b+ 72

Configuración de XMPP 80

Sensor de temperatura.....81

Conexión del circuito completo 84

Código del algoritmo.....85

Código GPIO Luminarias 85

Código lectura de temperatura 88

Código XMPP Nodo IoT 1.....89

Código XMPP Nodo IoT 2.....92

Capítulo IV. Desarrollo y resultados 96

Preparación para las pruebas 96

Cuentas.....96

Configuración cliente Pidgin 97

Configuración cliente Xabber 99

Inicializar nodos.....99

Rutas.....100

Ruta de comunicación directa 101

Ruta de comunicación redireccionada.....101

Ruta de comunicación con múltiples clientes activos.....102

Pruebas experimentales 103

Tiempo de respuesta 103

Throughput y retransmisiones.....131

Round Trip Time 146

Capítulo V: Conclusiones y recomendaciones 158

Conclusiones.....158

Recomendaciones.....159

Referencias Bibliográficas 160

Índice de tablas

Tabla 1: *Niveles de inteligencia según las capacidades* 27

Tabla 2: *Dispositivos de la arquitectura de IoT* 28

Tabla 3: *Aspectos fundamentales de la seguridad en IoT* 36

Tabla 4: *Ventajas y desventajas de las computadoras de placa única*.....56

Tabla 5: *Características del Raspberry Pi* 58

Tabla 6: *Comparativa Raspberry Pi 3 B y B+* 63

Tabla 7: *Descripción de nodos* 70

Índice de figuras

Figura 1: *IOT Internet de las cosas*. 25

Figura 2: *Capas arquitectura IoT* 28

Figura 3: *Flujo de datos IoT* 32

Figura 4: *Tipos de redes*. 33

Figura 5: *Redes según su alcance*. 33

Figura 6: *Redes según su disposición (topología).* 34

Figura 7: *Arquitectura cliente-servidor*. 40

Figura 8: *MSN Messenger*.....42

Figura 9: *MSN Messenger*.....43

Figura 10: *Yahoo! Messenger* 44

Figura 11: *Formato de paquete de datos OTR* 45

Figura 12: *Skype interfaz* 46

Figura 13: *Ejemplo de stanza*.....47

Figura 14: *Arquitectura típica XMPP* 50

Figura 15: *Comunicación entre clientes XMPP*.....51

Figura 16: *Message Stanza*.....52

Figura 17: *Message Stanza*.....53

Figura 18: *Computadora de placa única (Tinker Board)* 54

Figura 19: *Aplicaciones de computadoras de placa única* 55

Figura 20: *Procesador ARM Arquitectura* 57

Figura 21: *Rapsberry Pi*.....58

Figura 22: *Sistema operativo Rapsbian* 62

Figura 23: *Descripción de pines* 64

Figura 24: *Topología red IoT del proyecto* 69

Figura 25: *Versiones Raspbian*.....72

Figura 26: *Ventana Rufus* 73

Figura 27: *Escritorio Raspbian* 74

Figura 28: *Comando configurar raspberry*.....76

Figura 29: *Habilitar VNP* 76

Figura 30: *VNP viewer ingreso de credenciales* 77

Figura 31: *Habilitar SSH* 78

Figura 32. *Configuración Putty* 79

Figura 33: *Visualización Raspberry a través de Putyy* 79

Figura 34: *Comando para instalar SleekXMPP en Windows* 80

Figura 35: *Comando para instalar SleekXMPP en Raspberry* 80

Figura 36. *Sensor de temperatura DS18B20* 81

Figura 37. *Conexión sensor con Raspberry Pi* 82

Figura 38. *Instalación librería del sensor térmico* 82

Figura 39. *Configuración Raspberry Pi* 83

Figura 40. *Configuración 1-Wire* 83

Figura 41. *Conexión Raspberry Pi a LED* 84

Figura 42: *Diseño y conexión del circuito nodo IoT* 85

Figura 43: *Inicialización para GPIO* 86

Figura 44: *Función primer led verde al recibir dato* 86

Figura 45: *Función segundo led verde al recibir dato* 86

Figura 46: *Función led al calcular temperatura* 87

Figura 47: *Función led al completar el cálculo de temperatura* 87

Figura 48: *Función activar led rgb* 87

Figura 49: *Función activar led rgb* 88

Figura 50: *Código lectura de temperatura* 89

Figura 51: *Código XMPP Nodo IoT 1 parte 1* 89

Figura 52: *Código XMPP Nodo IoT 1 parte 2* 90

Figura 53: *Código XMPP Nodo IoT 1 parte 3* 91

Figura 54: *Código XMPP Nodo IoT 1 parte 4* 92

Figura 55: *Código XMPP Nodo IoT 2 parte 1* 92

Figura 56: *Código XMPP Nodo IoT 2 parte 2* 93

Figura 57: *Código XMPP Nodo IoT 2 parte 3* 94

Figura 58: *Código XMPP Nodo IoT 2 parte 4* 95

Figura 59: *Página principal Ligtwitch* 96

Figura 60: *Formulario creación de cuenta.* 97

Figura 61: *Cuadro de inicio Pidgin* 98

Figura 62: *Añadir cuenta cliente Pidgin*. 98

Figura 63: *Añadir cuenta cliente Xabber*. 99

Figura 64: *Nodos* 100

Figura 65: *Ruta de comunicación directa* 101

Figura 66: *Ruta de comunicación redireccionada* 102

Figura 67: *Ruta de comunicación con dos o más clientes activos* 103

Figura 68: *Paquetes al enviar instrucción directa de apagado de la cocina al Nodo 1* 104

Figura 69: *Paquetes al enviar instrucción directa de apagado de la sala al Nodo 1* 104

Figura 70: *Paquetes al enviar instrucción directa de apagado del comedor al Nodo 1* 105

Figura 71: *Paquetes al enviar instrucción directa de apagado del baño al Nodo 2* 105

Figura 72: *Paquetes al enviar instrucción directa de apagado del cuarto A al Nodo 2* 106

Figura 73: *Paquetes al enviar instrucción directa de apagado del cuarto B al Nodo 2* 106

Figura 74: *Paquetes al enviar instrucción directa de encendido de la cocina al Nodo 1* 107

Figura 75: *Paquetes al enviar instrucción directa de encendido de la sala al Nodo 1* 107

Figura 76: *Paquetes al enviar instrucción directa de encendido del comedor al Nodo 1* 108

Figura 77: *Paquetes al enviar instrucción directa de encendido del baño al Nodo 2* 108

Figura 78: *Paquetes al enviar instrucción directa de encendido del cuarto A al Nodo 2* 109

Figura 79: *Paquetes al enviar instrucción directa de encendido del cuarto B al Nodo 2* 109

Figura 80: *Paquetes al enviar instrucción directa de toma de temperatura al Nodo 1* 110

Figura 81: *Paquetes al enviar instrucción directa de toma de temperatura al Nodo 2* 110

Figura 82: *Paquetes al enviar instrucción redireccionada de apagado cocina al N* 2.....111

Figura 83: *Paquetes al enviar instrucción redireccionada de apagado sala al Nodo 2* 111

Figura 84: *Paquetes al enviar instrucción redireccionada de apagado comedor al N2*.....112

Figura 85: *Paquetes al enviar instrucción redireccionada de apagado baño al Nodo 1*.....112

Figura 86: *Paquetes al enviar instrucción redireccionada de apagado c. A al Nodo 1* 112

Figura 87: *Paquetes al enviar instrucción redireccionada de apagado c. B al Nodo 1* 113

Figura 88: *Paquetes al enviar instrucción redireccionada de encendido cocina al N2* 114

Figura 89: *Paquetes al enviar instrucción redireccionada de encendido sala al Nodo 2* 114

Figura 90: *Paquetes al enviar instrucción redireccionada de encendido comedor al N2* 115

Figura 91: *Paquetes al enviar instrucción redireccionada de encendido baño al N1* 115

Figura 92: *Paquetes al enviar instrucción redireccionada de encendido c. A al Nodo 1*.....116

Figura 93: *Paquetes al enviar instrucción redireccionada de encendido c. B al Nodo 1*.....116

Figura 94: *Paquetes al enviar instrucción redireccionada de temp. del N1 al N2* 117

Figura 95: *Paquetes al enviar instrucción redireccionada temp. del N2 al N1* 117

Figura 96: *Paquetes al enviar instrucción directa de apagado de la cocina al Nodo 1* 118

Figura 97: *Paquetes al enviar instrucción directa de apagado de la sala al Nodo 1* 118

Figura 98: *Paquetes al enviar instrucción directa de apagado del comedor al Nodo 1* 119

Figura 99: *Paquetes al enviar instrucción directa de apagado del baño al Nodo 2* 119

Figura 100: *Paquetes al enviar instrucción directa de apagado del cuarto A al Nodo 2* 120

Figura 101: *Paquetes al enviar instrucción directa de apagado del cuarto B al Nodo 2* 120

Figura 102: *Paquetes al enviar instrucción directa de encendido de la cocina al N*1 121

Figura 103: *Paquetes al enviar instrucción directa de encendido de la sala al Nodo 1* 121

Figura 104: *Paquetes al enviar instrucción directa de encendido del comedor al N1* 122

Figura 105: *Paquetes al enviar instrucción directa de encendido del baño al Nodo 2* 122

Figura 106: *Paquetes al enviar instrucción directa de encendido del cuarto A al N2* 123

Figura 107: *Paquetes al enviar instrucción directa de encendido del cuarto B al N2* 123

Figura 108: *Paquetes al enviar instrucción directa de temperatura al Nodo 1* 124

Figura 109: *Paquetes al enviar instrucción directa de temperatura al Nodo 2* 124

Figura 110: *Paquetes al enviar instrucción redireccionada de apagado cocina al N2* 125

Figura 111: *Paquetes al enviar instrucción redireccionada de apagado sala al N2* 125

Figura 112: *Paquetes al enviar instrucción redireccionada de apagado comedor al N2*.....126

Figura 113: *Paquetes al enviar instrucción redireccionada de apagado baño al N1*.....126

Figura 114: *Paquetes al enviar instrucción redireccionada de apagado c. A al N1*.....127

Figura 115: *Paquetes al enviar instrucción redireccionada de apagado c. B al N1* 127

Figura 116: *Paquetes al enviar instrucción redireccionada de encendido cocina al N2* 128

Figura 117: *Paquetes al enviar instrucción redireccionada de encendido sala al N2* 128

Figura 118: *Paquetes al enviar instrucción redireccionada de encendido comedor al N2* 128

Figura 119: *Paquetes al enviar instrucción redireccionada de encendido baño al N1* 129

Figura 120: *Paquetes al enviar instrucción redireccionada de encendido c. A al N1*.....129

Figura 121: *Paquetes al enviar instrucción redireccionada de encendido c. al N1*.....130

Figura 122: *Paquetes al enviar instrucción redireccionada de temperatura al Nodo 1* 130

Figura 123: *Paquetes al enviar instrucción redireccionada de temperatura al Nodo 2* 131

Figura 124: *Throughput y retransmisiones, instrucciones directas de apagado al N1*.....132

Figura 125: *Throughput y retransmisiones, instrucciones directas de apagado al N2*.....133

Figura 126: *Throughput y retransmisiones, instrucciones directas de encendido al N1*.....134

Figura 127: *Throughput y retransmisiones, instrucciones directas de encendido al N2*.....134

Figura 128: *Throughput y retransmisiones, instrucciones directas de temperatura al N1*.....135

Figura 129: *Throughput y retransmisiones, instrucciones directas de temperatura al N2*.....135

Figura 130: *Throughput y retransmisiones, al enviar instrucciones redireccionadas de apagado al nodo 1* 136

Figura 131: *Throughput y retransmisiones al enviar instrucciones redireccionadas de apagado al nodo 2* 137

Figura 132: *Throughput y retransmisiones al enviar instrucciones redireccionadas de encendido al nodo 1* 137

Figura 133: *Throughput y retransmisiones al enviar instrucciones redireccionadas de encendido al nodo 2* 138

Figura 134: *Throughput y retransmisiones al enviar instrucciones redireccionadas de temperatura al nodo 1* 138

Figura 135: *Throughput y retransmisiones al enviar instrucciones redireccionadas de temperatura al nodo 2* 139

Figura 136: *Throughput y retransmisiones al enviar instrucciones directas de apagado al nodo 1* 139

Figura 137: *Throughput y retransmisiones al enviar instrucciones directas de apagado al nodo 2* 140

Figura 138: *Throughput y retransmisiones al enviar instrucciones directas de encendido al nodo 1* 140

Figura 139: *Throughput y retransmisiones al enviar instrucciones directas de encendido al nodo 2* 141

Figura 140: *Throughput y retransmisiones al enviar instrucciones directas de temperatura al nodo 1*....141

Figura 141: *Throughput y retransmisiones al enviar instrucciones directas de temperatura al nodo 2*....142

Figura 142: *Throughput y retransmisiones al enviar instrucciones redireccionadas de apagado al nodo 1* 143

Figura 143: *Throughput y retransmisiones al enviar instrucciones redireccionadas de apagado al nodo 2* 143

Figura 144: *Throughput y retransmisiones al enviar instrucciones redireccionadas de encendido al nodo 1* 144

Figura 145: *Throughput y retransmisiones al enviar instrucciones redireccionadas de encendido al nodo 2* 144

Figura 146: *Throughput y retransmisiones al enviar instrucciones redireccionadas de temperatura al nodo 1* 145

Figura 147: *Throughput y retransmisiones al enviar instrucciones redireccionadas de temperatura al nodo 2* 145

Figura 148: *RTT al enviar la instrucción directa de apagado al nodo 1* 146

Figura 149: *RTT al enviar la instrucción directa de apagado al nodo 2* 147

Figura 150: *RTT al enviar la instrucción directa de temperatura al nodo 1* 147

Figura 151: *RTT al enviar la instrucción directa de temperatura al nodo 2* 148

Figura 152: *RTT al enviar la instrucción redireccionada de apagado de N1 al N2* 149

Figura 153: *RTT al enviar la instrucción redireccionada de apagado de N2 al N1* 150

Figura 154: *RTT al enviar la instrucción redireccionada de temperatura al nodo 2* 150

Figura 155: *RTT al enviar la instrucción redireccionada de temperatura al nodo 1* 151

Figura 156: *RTT al enviar la instrucción directa de apagado al nodo 1* 152

Figura 157: *RTT al enviar la instrucción directa de apagado al nodo 2* 153

Figura 158: *RTT al enviar la instrucción directa de temperatura al nodo 1* 153

Figura 159: *RTT al enviar la instrucción directa de temperatura al nodo 2* 154

Figura 160: *RTT al enviar la instrucción redireccionada de apagado del N1 al N2* 155

Figura 161: *RTT al enviar la instrucción redireccionada de apagado del N2 al N1* 155

Figura 162: *RTT al enviar la instrucción redireccionada de temperatura del N1 al N2* 156

Figura 163: *RTT al enviar la instrucción redireccionada de temperatura del N2 al N1* 157

Resumen

Para la realización del presente proyecto de investigación, se utilizó el protocolo XMPP - *Extensible Messaging Presence Protocol* (Protocolo extensible de mensajería y presencia) para realizar la comunicación en tiempo real entre cuatro nodos IoT (Internet de las Cosas), con la finalidad de compartir datos y comandos. Este protocolo, debido a su adaptabilidad, permite ser implementado de manera muy práctica para diferentes dispositivos. Se pretendió diseñar una red de cuatro nodos que permitió desarrollar un algoritmo de funcionamiento utilizando un servidor XMPP gratuito. Esta red diseñada contó con actuadores para monitorear en tiempo real sus variables físicas. Para las pruebas, la red propuesta, se diseñó mediante una topología adecuada para monitoreo, que consiste por dos nodos IoT, dos clientes (un PC con Windows y un teléfono inteligente con Android) y como servidor para las comunicaciones XMPP público. Nodo 1. Fue una SBC (*Single Board Computer* o Pc de placa única) Raspberry Pi 3b+. Tuvo directamente conectado un sensor de temperatura (sensor 1) y tres actuadores (actuador 1, 2, 3). Nodo 2. Fue una SBC Raspberry Pi 3b+. Tuvo directamente conectado un sensor de temperatura (sensor 2) y tres actuadores (actuador 4, 5, 6). Cliente. Los dispositivos clientes fueron una PC con Windows y un teléfono inteligente con Android. Cualquiera de los dos clientes o los dos al mismo tiempo pudieron estar activos. Servidor XMPP. El servidor utilizado fue uno público y gratuito, existen muchos servidores de este tipo en la web, para este proyecto utilizó jabber.at. Se realizaron distintas pruebas de la red, para probar su eficiencia, retardo para entrega de mensajes, lo que permitió definir qué tan apropiada es esta red para proyectos IoT sencillos y complejos.

PALABRAS CLAVE:

- **XMPP**
- **IOT**
- **SBC**
- **RASPBERRY PI 3 B+**

Abstract

For the present research project, the XMPP - Extensible Messaging Presence Protocol will be used to perform real-time communication between four IoT (Internet of Things) nodes, in order to share data and commands. This protocol due to its adaptability allows it to be implemented in a very practical way for different devices. The aim is to design a 4-node network that allows the development of an operating algorithm using a free XMPP server. This designed network will have actuators and its physical variables can be monitored in real time. The devices to be used are two Raspberry Pi 3b+, a Windows PC and an Android smartphone. For the tests of the proposed network, designed using a suitable topology for monitoring, consisting of two IoT nodes, two clients (a Windows PC and an Android Smartphone) and a public XMPP server will be used as the server for communications. Node 1. It will be a Raspberry Pi 3b+ SBC (Single Board Computer or Pc). It will have a temperature sensor (Sensor 1) and three actuators (Actuator 1, 2, 3) directly connected. Node 2. It will be a SBC Raspberry Pi 4. It will have directly connected a temperature sensor (Sensor 2) and three actuators (Actuator 4, 5, 6). Client. The client devices will be a Windows PC and an Android Smartphone, any version of the operating systems that supports the applications to be used is adequate. Either or both clients can be active at the same time. XMPP server. The server used will be a public and free one, there are many servers of this type on the web, for this project jabber.at will be used. Different tests of the network will be carried out to test its efficiency, delay for delivery of messages, which will allow defining how appropriate this network is for simple and complex IoT projects.

KEYWORDS:

- **XMPP**
- **IOT**
- **SBC**
- **RASPBERRY PI 3 B+**

Capítulo I. Introducción

Antecedentes

El internet de las cosas o IoT (*Internet Of Things*) se define como la interconexión de aparatos electrónicos de uso diario a través de internet, son cosas que usualmente se tiene en el hogar, que antes no presentaban una conexión a una red tales como: automóviles, electrodomésticos, luminarias, sensores, etc. Para que todos estos objetos sean controlados de forma remota ya sea por personas u otros objetos. Cómo se puede observar es un cambio drástico en la calidad de vida, se pretende el fácil acceso de los datos y esto ofrece más oportunidad en ámbitos de educación, seguridad y transporte, entre otros campos. Este concepto de internet de las cosas fue propuesto en 1999 por Kevin Ashton, en Auto-ID Center, donde se trabajaban en investigaciones de campo de identificación por radiofrecuencia en red (RFID) y tecnologías de sensores (Elder, 2019).

La principal idea de IoT es la de comunicar cosas a través de internet y así convertirlas en inteligentes, esto quiere decir que se pueda controlar, medir y recolectar datos. Cómo se conoce, las aplicaciones en dispositivos móviles conectados a internet son bastantes, la mayoría de las empresas utiliza microprocesadores que puedan realizar estas operaciones de comunicación, pero esto conlleva que exista un bajo consumo de energía, las computadoras de placa única o SBC (*Single Board Computer*), las Rapsberry Pi 3b+ son muy utilizadas en este campo por su bajo costo y gran capacidad para manejo de datos.

Cuando se tiene los dispositivos inteligentes para formar la red, se necesita una comunicación que sea muy eficaz para la ejecución de los respectivos requerimientos de cada usuario, por ende, entran diversos protocolos de comunicación y mensajería, ya que estos son encargados del envío y recepción de los distintos paquetes de datos, el protocolo extensible de mensajería y presencia o XMPP (*Extensible Messaging and Presence*), es muy utilizado para mensajería instantánea (González, 2014).

En la actualidad las empresas se están dedicando al desarrollo de protocolos de comunicación de redes IoT con distintos parámetros necesarios como: retardo de mensajes, paquetes perdidos, *throughput*, etc. Estos protocolos son bastantes complejos para su implementación, la mayoría no son de libre acceso y requieren de licencias y servidores. Para empresas de gran escala no representan ningún inconveniente, pero sí para las empresas pequeñas.

Justificación

En el presente proyecto de investigación se realizará el análisis del protocolo de mensajería instantánea XMPP, desarrollado para un enfoque de transporte de comunicación en los sistemas IoT, el mismo que será implementado en dos raspberry Pi 3b+, a la vez que se determinará los parámetros necesarios para una correcta comunicación, tales como retardo de mensajes, *throughput*, *jitter*, paquetes perdidos, los que serán receptados desde una computadora mediante un sensor de temperatura y los diodos leds (nodos). Los nodos estarán enviando datos en tiempo real, desde una misma red de internet.

Todo este análisis tiene como finalidad el poder entender el desempeño de este protocolo de mensajería y su posterior implementación en diferentes dispositivos electrónicos para proyectos simples y complejos de un sistema IoT.

El análisis del protocolo de mensajería, verificará las ventajas predefinidas, por ser un código abierto y de fácil acceso para cualquier usuario basado en XML. Del análisis previo se conoce su utilización principal en servicios de mensajería instantánea. Además, se determinará que la utilización en sistemas de gran escala para distintos escenarios, si es seguro frente ataques electrónicos, descentralizado, extensible y escalable.

Objetivos

General

Realizar el análisis del desempeño del protocolo de comunicación XMPP en una Red IOT.

Específicos

- Comprobar que la implementación de un protocolo de mensajería instantánea XMPP, es viable para diferentes dispositivos electrónicos indistintamente del sistema operativo que maneje.
- Plantear un procedimiento de comunicación entre los nodos de la red IoT empleando el protocolo de mensajería XMPP que permita el acceso desde cualquier lugar.
- Analizar el desempeño de una red con varios nodos IoT que transmitan información al usuario en tiempo real a través de internet y los parámetros físicos obtenidos a partir de sensores.
- Analizar los distintos parámetros para el desempeño de la red IoT, como: paquetes perdidos, retardo de mensaje, throughput, jitter.

Capítulo II. Marco Teórico

Internet de las Cosas

El internet de las cosas surge de la realización de la gran cantidad de objetos que utilizan las personas en su vida cotidiana. Durante todo el día las personas interactúan con diferentes objetos con múltiples propósitos, de ahí se generó la idea de convertir dichos objetos en inteligentes, esto quiere decir que tengan la capacidad de conectarse a internet o una red local de manera que puedan recolectar y, posteriormente, enviar la data e información generada por dicha interacción (RedHat, s.f.).

La actualización al protocolo IPv4 denominado IPv6 es el que hace que esta idea se convierta en una realidad ya que permite otorgar direcciones únicas a cada dispositivo en una cantidad mucho mayor que su antecesor. En la figura que se muestra a continuación es posible observar en general lo que representa el internet de las cosas (Gracia, s.f.).

Figura 1

IOT Internet de las cosas.

Nota. Tomado de www.freepik.com

Objetivo de IoT

La cuarta revolución industrial, también llamada Industria 4.0, tiene como una base o pilar al Internet de las cosas ya que su objetivo es el de organizar la producción de tal manera que se digitalice y automatice toda actividad humana. Esto no representa el omitir la actuación de las personas sino el ayudar que las actividades de estas sean mucho más sencillas y rápidas de realizar, además de obtener tras ello información valiosa para cambiar o mejorar procesos (Cruz Vega, y otros, 2015).

Por ello el objetivo principal es de dotar a los objetos de capacidades particulares que los convierten en inteligentes (Quiñonez, 2019):

Identidad

Corresponde a un código o identificador qué hará único e irrepetible al objeto, así los demás dispositivos estarán en la posibilidad de diferenciarlo de otros y que un objeto pueda tomar el puesto de otro.

Direccionamiento

Esto se refiere a la anteriormente mencionada dirección IPv6, la misma que será la puerta para establecer la comunicación con el exterior.

Comunicación

Pueden conectarse a internet para tener acceso a la red mundial con la finalidad de compartir e intercambiar información con los distintos objetos y dispositivos.

Acción

Esta capacidad le da la posibilidad al objeto de realizar un cambio de manera física en su entorno ya sea esta visual, auditiva u otro.

Localización

El objeto reconoce su ubicación en la tierra mediante sensores, en este caso un GPS o alguna tecnología con la misma finalidad.

Al estudiar dichas capacidades es entonces posible establecer niveles a la inteligencia de un objeto dependiendo de cuál o cuáles capacidades tiene, estos niveles son puestos por cada empresa desarrolladora, si bien no existe un consenso en este tema, para los propósitos de este trabajo de investigación, se tomarán en cuenta los siguientes:

Tabla 1

Niveles de inteligencia según las capacidades

Nivel 1	**Identidad**	El objeto puede reconocerse y diferenciarse de los demás
Nivel 2	**Ubicación**	El objeto conoce su localización en la tierra
Nivel 3	**Estado**	El objeto puede comunicar al servidor u otro si se encuentra operacional, no operacional u otro
Nivel 4	**Contexto**	El objeto puede percibir y entender lo que sucede en su entorno
Nivel 5	**Criterio**	El objeto actúa según el entendimiento de su contexto

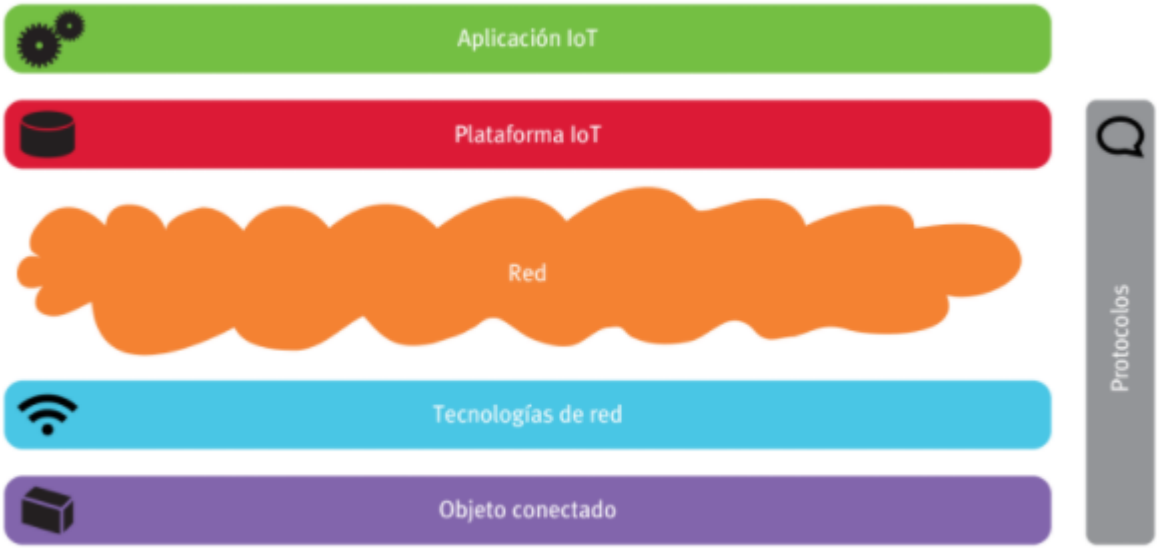
Nota. Tomado de *(Domodesk, s.f.).*

Arquitectura del IoT

La arquitectura del Internet de las cosas se basa en cinco capas o áreas fundamentales que están bien definidas ya que son las que permiten la conectividad de la inmensa cantidad de objetos que se pretende comunicar entre sí, estas capas son: la capa aplicación, la plataforma IoT, la red, la tecnología de red y los objetos conectados.

Figura 2

Capas arquitectura IoT



Nota. Tomado de *(Cruz Vega, y otros, 2015)*.

Según la figura es posible separar las capas en dos grupos distintos: uno los dispositivos, que corresponden al objeto conectado y dos la red, que son las capas de tecnología de red, red y plataforma de red. De esta manera resulta apropiado primero explicar cuáles son los tipos de dispositivos que actualmente se encuentran admitidos en esta arquitectura IoT.

Tabla 2

Dispositivos de la arquitectura de IoT

Tipo de dispositivo	Descripción
Pequeños	Son aquellos que no disponen de un sistema operativo.
Intermedios	Tienen sistemas operativos de 32 bits especialmente diseñados para una tarea específica, pueden ser en Linux o Windows Iot.
Grandes o plataforma	Tienen sistemas operativos de 32 o 64 bits, estos sistemas suelen ser Android, Linux, entre otros. Como el mayor ejemplo de estos dispositivos se tienen a los celulares inteligentes o al Rapsberry Pi.

Nota. Tomado de *(Cárdenas, 2020)*.

Cabe recalcar que dentro de los dispositivos se encuentran los denominados sensores o también llamados como la capa de percepción, estos tienen el objetivo de recuperar datos de los fenómenos que ocurren en su ambiente, como por ejemplo la temperatura, la altura, humedad, localización, etc. Ahora el segundo grupo comprendido por las capas relacionadas con la red se describen de la siguiente manera (Cruz Vega, y otros, 2015):

- **Tecnologías de red:** Son aquellas encargadas de la transmisión de la información recolectada por los dispositivos hacia los sistemas de procesamiento en los que se realizarán los distintos análisis.
- **Plataforma IoT:** En esta capa se procesa la información recolectada, para tomar decisiones según amerite y luego guardar la información en una base de datos.

Además de estos detalles en la aplicación del Internet de las cosas se tienen varios tipos de redes de acuerdo al orden de conexión de los distintos dispositivos, esto se ampliará más adelante.

Características

Para que una arquitectura IoT funcione de manera correcta debe cumplir con ciertas características que se enlistan a continuación (VIU, 2018):

- Los dispositivos clientes y servidores deben tener una comunicación segura entre ellos.
- Los dispositivos maliciosos o que fueran hurtados deben poder desconectarse de manera remota para preservar la seguridad.
- Se debe tener presente en todo momento qué dispositivos se encuentran conectados a la red.
- Los dispositivos deben poder actualizar su sistema.
- Deben poder tomar acción de acuerdo a un correcto análisis de los datos obtenidos.
- Debe poder actualizar la información de seguridad como contraseñas, pin, etc.

- Se debe poder admitir usuarios, dispositivos y tecnologías nuevas sin tener que realizar cambios grandes en la arquitectura.
- Control sobre las opciones del hardware.
- Si el dispositivo se pierde o es hurtado debe poder localizarse.
- La disponibilidad de los dispositivos debe ser permanente.
- La configuración de la red e infraestructura debe poder configurarse de manera remota a través de internet.

Redes del IoT

Para describir el comportamiento de la red dentro del internet de las cosas se debe primero establecer los denominados niveles del sistema IoT, estos se refieren a los puntos por los que pasa la comunicación, a continuación, se describen (Sandoval, 2020):

Dispositivos

Los dispositivos son los aparatos que recolectan la información y realizan la acción, son denominados nodos. Estos no cuentan con una conexión directa a internet, para ello deben comunicarse con el siguiente nivel llamado *Gateway*.

Gateway

El Gateway cumple la función de comunicar los nodos al internet para que la data recolectada llegue al nivel de sistema de datos. Estos pueden ser receptores de transferencias inalámbricas como *bluetooth* o w*ireless*.

Sistema de datos

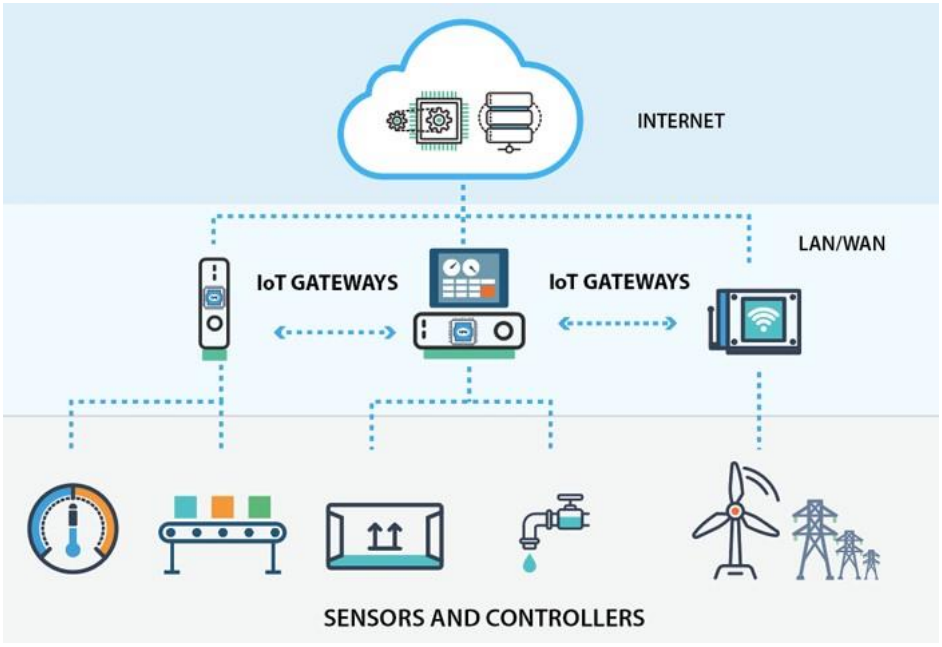
Este sistema corresponde a los servidores que se encargan de analizar la información obtenida por los nodos y entregada por el *Gateway.* Los servidores realizan el análisis mediante algoritmos diseñados exclusivamente para esa tarea.

Ahora bien, comprendidos los niveles del IoT se puede entonces hablar acerca de los flujos de datos que pueden existir entre estos niveles. Esto se explica en la siguiente lista (Nallapaneni, 2018):

- **Dispositivo a dispositivo:** Es sencillamente la transmisión de mensajes de un nodo otro sin la necesidad de un intermediario o *Gateway.*
- **Dispositivo a *Gateway*:** Esta conexión ocurre cuando el nodo tiene que enviar la información al sistema de datos, por ello se comunica con el *gateway* que realizará el correspondiente direccionamiento.
- **Gateway a Sistema de datos:** Es la conexión con la que finalmente el servidor o nube recibe la información recolectada por los nodos con la finalidad de procesarla.
- **Sistema de datos a Sistema de datos:** Esta comunicación puede darse en los casos en que uno de los servidores no puede realizar un servicio necesario, por ello lo envía a otro con la capacidad de hacerlo.

Figura 3

Flujo de datos IoT



Nota. Tomado de *(Rojas, 2020)*.

Tipos de redes

Las redes pueden clasificarse de diferente manera de acuerdo a dos conceptos básicos que son el alcance y la topología, el primero hace referencia al tamaño de la superficie o área que abarca la conectividad de la red, el segundo se trata de la disposición en cómo se conectan los dispositivos o nodos, gateways y servidores. A continuación, se explicará de forma breve los diferentes tipos de redes de acuerdo a su alcance (Juliá, s.f.):

Figura 4

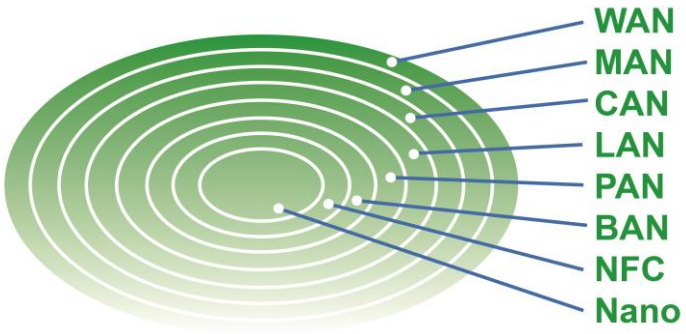
Tipos de redes.

Tipo	Descripción
Nano	Utilizada especialmente en el campo de la nanotecnología se utiliza para comunicar dispositivos que se encuentran muy cerca.
NFC	Por sus siglas en inglés Comunicación de campo cercano. Corresponde a la comunicación entre dispositivos separados entre sí tan solo unos centímetros.
BAN	Por sus siglas en inglés Red de Área Corporal, como su nombre lo indica es la red que abarca el cuerpo del usuario, como por ejemplo entre su teléfono inteligente y su marcapasos, reloj inteligente o gafas de realidad aumentada.
PAN	Por sus siglas en inglés Red de Área Personal. Comunica dispositivos que se encuentran a pocos metros de distancia, por ejemplo, la conectividad de los nodos dentro de una habitación.
LAN	Por sus siglas en inglés Red de Área Local, Puede comunicar dispositivos a varios metros de distancia como una casa, departamento o edificio de una empresa.
CAN	Por sus siglas en inglés Red de Área Corporativa. Comunica dispositivos dentro del alcance de los predios de una empresa o corporación, por ejemplo, el campus de una universidad.
MAN	Por sus siglas en inglés Red de Área Metropolitana, esta puede abarcar varios kilómetros y corresponde a la conectividad de una ciudad entera.
WAN	Por sus siglas en inglés Red de Área Amplia. Es una red a gran escala que comunica varios países entre sí, la que se conoce comúnmente como Internet.

Nota. Tomado de *(Juliá, s.f.).*

Figura 5

Redes según su alcance.

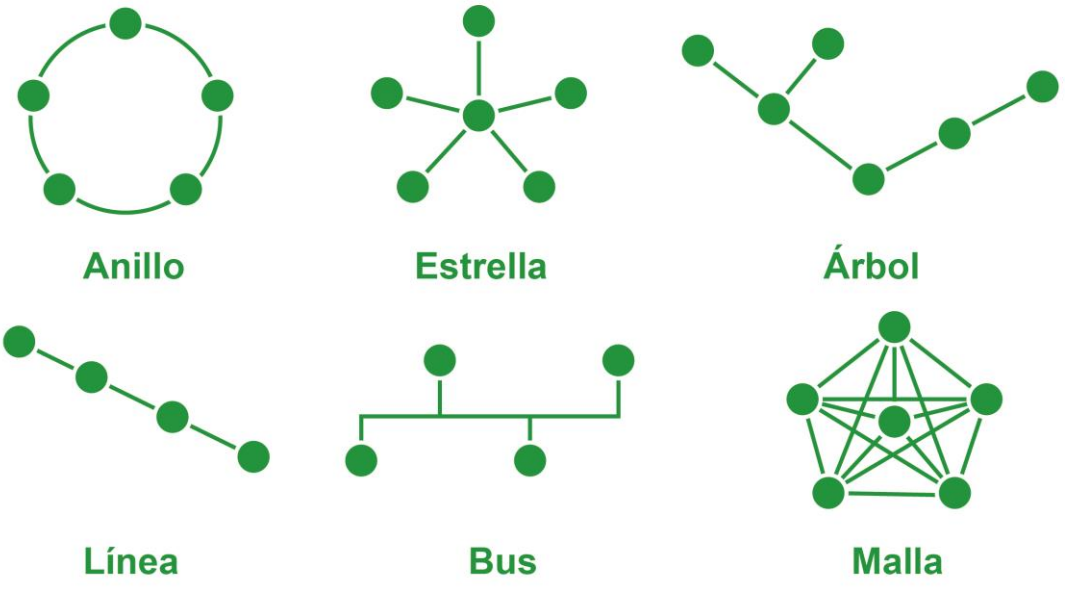


La clasificación según la topología es la siguiente (Lederkremer, 2019):

- **Anillo:** Los dispositivos se encuentran conectados de los extremos formando un círculo entre todos.
- **Estrella:** Los dispositivos se conectan a un dispositivo central, usualmente un *gateway*.
- **Árbol:** Múltiples redes de tipo estrella.
- **Línea:** Cada dispositivo se conecta únicamente al siguiente sin que el último se conecte con el primero.
- **Bus:** Cada dispositivo se conecta a una línea central o troncal.
- **Malla:** Todos los dispositivos se encuentran comunicados directamente con todos.

Figura 6

Redes según su disposición (topología).



Protocolos

Los protocolos son reglas que necesariamente tienen que cumplir los sistemas que quieren comunicarse y transferir información entre sí. El internet utiliza el protocolo TCP/IP, esto es verdad para cualquier dispositivo, aparato o sistema que quiera ser parte del internet, desde los teléfonos celulares hasta los computadores de sobremesa independientemente del sistema operativo con el que funcionen, por ello las cosas que se pretenden hacer inteligentes deben regirse a este estándar.

Ahora dentro del TCP/IP existen otros protocolos que son utilizados para diferentes fines evitando así confusiones o pérdida de información, entre ellos se tiene al FTP, HTTP, POP3, DNS, XMPP, entre otros. A continuación, se detallarán un poco más a fondo los más importantes (Lederkremer, 2019).

- FTP: Protocolo de transferencia de datos utilizado para enviar y recibir archivos.
- SMTP: Protocolo simple de transferencia de correo, se dedica a controlar la recepción y envío de correos electrónicos.
- TCP: Protocolo de control de transporte. Maneja los paquetes de datos transferidos desde un dispositivo emisor a un receptor.
- UDP: Protocolo de datagrama de usuario, funciona como transporte a la par del protocolo TCP.
- HTTP: Permite realizar peticiones de datos y recursos en archivos de tipo HTML generalmente.
- POP3: Protocolo estándar para la recepción de correos electrónicos, trabaja con los puertos 110 y 995.
- XMPP: Protocolo extensible de mensajería y comunicación de presencia, utilizado para mensajería instantánea, de voz y video.

Seguridad

Desde la llegada de la comunicación en red y, sobretodo, de internet, la seguridad ha tomado un papel fundamental en el correcto uso de estas tecnologías, esto es porque esta comunicación a concedido a personas con malas intenciones acceder remotamente a la información de los usuarios con objetivos tan diversos pero criminales como el robo, extorsión, suplantación de identidad entre otros.

Como se puede comprender el internet de las cosas no se escapa de esto, y es por ello que se deben aplicar formas o soluciones de seguridad para que únicamente el usuario tenga acceso y control de sus dispositivos, permitiéndole tener privacidad y confianza en que su información no será utilizada en su contra. Para poder entender cómo se aplica este concepto en el IoT se debe entender que existen tres aspectos fundamentales disponibilidad, confidencialidad e integridad para cumplir con estos aspectos se crearon tres capas de abstracción que son la de capa de percepción, capa de red y capa intermedia o de aplicación (Microsoft, s.f.).

Tabla 3

Aspectos fundamentales de la seguridad en IoT

Aspecto	Descripción
Integridad	Se refiere que la información no sea alterada, robada o llegue a un destino diferente al que fue enviada, esto puede ocurrir por fenómenos naturales al cortarse líneas de comunicación o por intervención humana. Se han creado distintas formas de asegurar la integridad como redundancia cíclica, sumas de verificación entre otros.
Disponibilidad	Especifica la capacidad del usuario de tener acceso a la información tanto en buena condición cómo en la condición más deplorable, siempre cumpliendo con el anterior aspecto, integridad.
Confidencialidad	Este aspecto se encarga de asegurar que solo el usuario al que fue enviada la información pueda verla en su totalidad, previniendo que terceras personas puedan acceder a ella en cualquier capa de la comunicación en que se encuentre el envío del paquete de datos, una de las prácticas más comunes para llevar esto a cabo es la encriptación.

Nota. Tomado de *(Cárdenas, 2020)*.

Las capas de abstracción se definen de la siguiente manera (Thales, s.f.):

- **Capa de percepción:** Se la conoce dentro de la arquitectura del internet de las cosas como la capa del nivel inferior, la base. Su trabajo es controlar el primer contacto del dispositivo con la información entrante. Por ello debe cumplir con la función de autenticar los dispositivos (nodos) de acuerdo a las credenciales únicas de cada uno de ellos mientras mantiene la privacidad con la aplicación de programas de encriptación que protegen el contenido del paquete de datos, así como de información sensible del dispositivo (identidad y localización). Además, valora riesgos de mantener en funcionamiento un nodo cuya seguridad haya sido vulnerada, teniendo la capacidad de detener su funcionamiento.
- **Capa de red:** En esta capa que se encarga de la conexión punto a punto se necesita un cifrado que bloquee el acceso a dispositivos no autorizados en lo referente a la autenticación. Se encarga de que el enrutamiento de los datos sea exacto y sin fallas, de tal manera que la información llegue a su destinatario de una forma u otra, usualmente mediante redundancia. Además, se encarga de que los datos mantengan su privacidad.
- **Capa intermedia o de aplicación:** En esta capa se deben poner más esfuerzos en la autentificación, ya que los ataques se realizan en primera instancia a través de esta, es por ello que las credenciales son procesadas a través de la nube. Se establecen firewalls y cifrado para mantener la seguridad de los datos, se evalúan riesgos para prevenir problemas como intrusos, ataques, entre otros.

Ahora a manera de resumen se enlistan las amenazas que puede afectar una infraestructura IoT (Microsoft, s.f.):

- Suplantación de identidad.
- Revelación de Información.

- Alteración.
- Denegación de servicio.
- Elevación de privilegios.

Los ataques de ciberseguridad de Iot pueden amenazar (Microsoft, s.f.):

- Los procesos.
- La comunicación.
- El almacenamiento.

Para que se pueda brindar la mayor seguridad posible en los nodos es necesario realizar continuos chequeos y mantenimiento a ellos mismos, de manera que la información que recolecten o actividad que realicen sea íntegro, también se pueden implementar alarmas cuando el nodo identifique un cambio en su hardware, ya que sería una posible intrusión. Para solventar el tema de disponibilidad los nodos cuentan con fuentes de energía que duran un tiempo considerable, ya que su hardware es compacto y no tiende a gastar mucha energía.

Algunos de los sectores en los que ya se encuentran aplicada esta tecnología del Internet de las Cosas son: la red eléctrica con control remoto de medidores, la agricultura con sistemas automatizados de riego y monitoreo del clima, la automatización de procesos en la industria, medicina, ciudades inteligentes, hogares, entre otros (Fracttal, s.f.).

Protocolos de mensajería instantánea

La mensajería instantánea nace del requerimiento de la humanidad de sostener conversaciones mediante las nuevas tecnologías de la comunicación de manera inmediata y sencilla, problema que no podía solventar de manera efectiva los correos electrónicos. El *chat* fue la solución, un software que se

fundamenta en envío y recepción de texto en tiempo real, siempre y cuando los dispositivos intervinientes se encuentren activos. Además, de acuerdo al protocolo que se utilice para este fin, se puede necesitar de un servidor que controle el proceso. En la actualidad es posible hasta realizar llamadas con video utilizando estos mismos protocolos, por ello resulta relevante resaltar algunas de las funciones que ofrecen (Molina, 2014):

- **Aviso de presencia:** Notifica si un usuario está conectado en ese momento o no.
- **Buzón:** Guarda los mensajes que llegan a usuarios que no se encuentren conectados en el momento del envío del mensaje.
- **Transferencia de archivos:** Envío y recepción de archivos entre usuarios.
- **Comunicación por voz:** Realizar llamadas para comunicarse mediante micrófono y altavoz.
- **Comunicación con video:** Permite añadir la compartición de imagen por cámara durante llamadas.

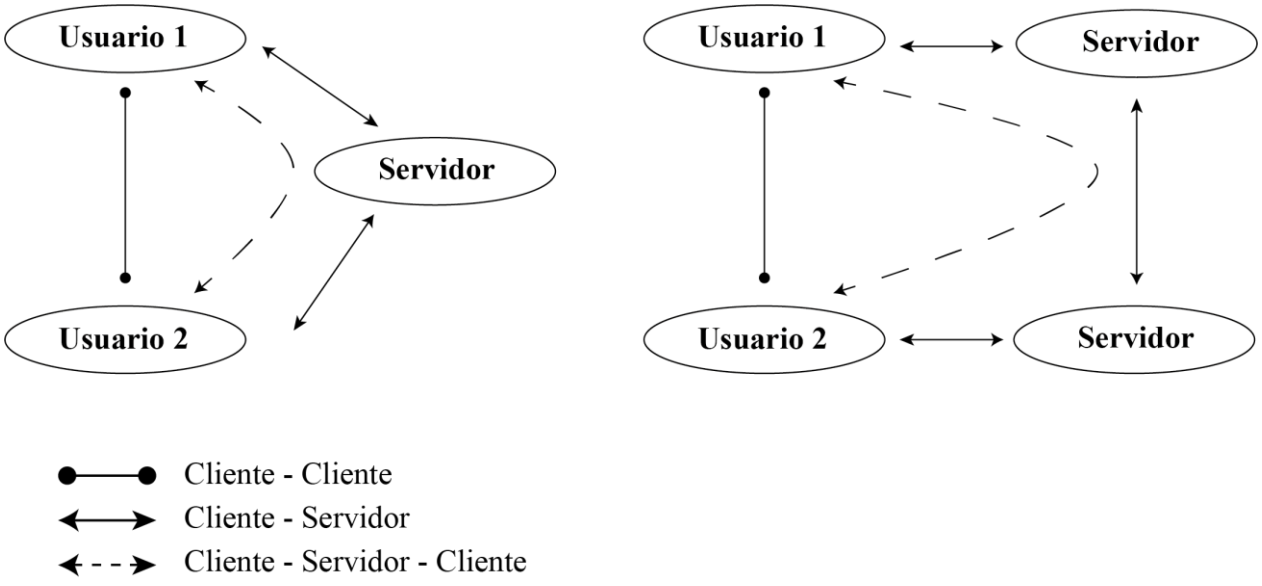
Arquitectura

Para entender la arquitectura utilizada en los protocolos de mensajería instantánea es necesario conocer su historia, en un principio todos los dispositivos que quería conectarse entre sí podían no ser compatibles por utilizar distintos parámetros, de ahí nació la necesidad de crear estándares para que todos se rijan a ellos y pueda conseguirse una comunicación transparente entre los diferentes dispositivos, esto ocurrió en el año 2000 cuando la *Internet Engineering Task Force* (IETF) publico dos *Request for Comments* (RFC) sobre el tema. Este estándar se lo conocería como Instant Messagon and Precence Protocol Working Group (IMPPWG), gracias a esto en 2002 se creó el primer borrador del protocolo IMPP que luego pasó a ser el conocido XMPP, protocolo en el que se basa este trabajo investigativo y se conocerá a fondo más adelante.

La arquitectura genérica y más común utilizada es la de cliente – servidor, en la que los dispositivos se conectan a un servidor que se encarga de gestionar la comunicación y direccionar el mensaje a su destinatario como se puede ver en la siguiente figura.

Figura 7

Arquitectura cliente-servidor.



Como se puede obtener de la figura, el usuario uno consigue una sesión en el servidor, mismo que le notifica cuáles de sus contactos se encuentran disponibles, luego este usuario realiza una petición al servidor para empezar la comunicación, el servidor recibe el mensaje y luego lo reenvía al usuario dos. También puede concederles una comunicación directa como por ejemplo en la transferencia de un archivo, en la que el archivo nunca llega al servidor, sino que pasa directo de un usuario a otro (Delgado, 2013).

Protocolos más utilizados

Para poder comprender las ventajas y desventajas del protocolo XMPP es necesario conocer su competencia, por ello, a continuación, se explicará brevemente los protocolos de mensajería instantánea más utilizados que son: MSNP, OSCAR, YMSG y Skype.

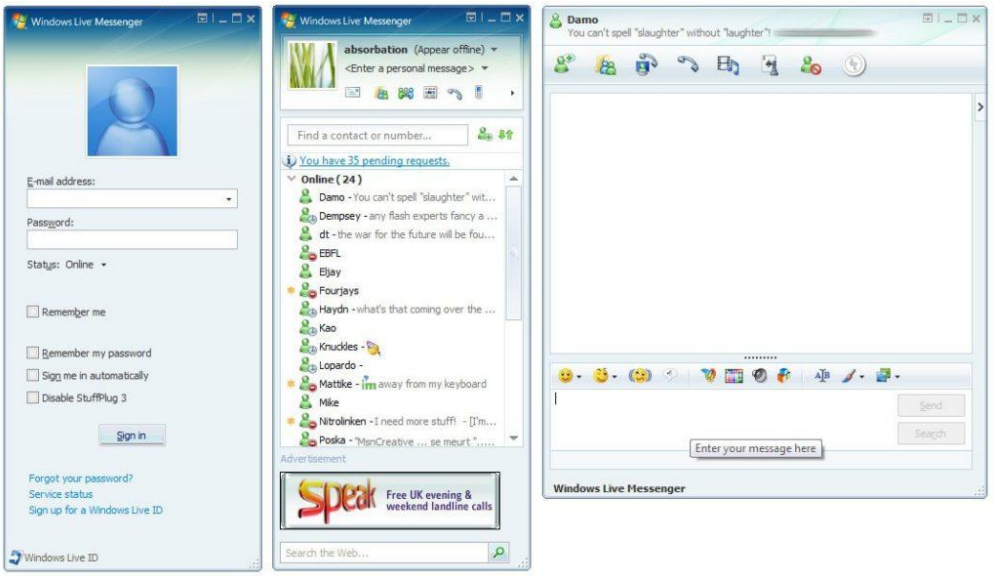
Protocolo Mobile Status Notification Protocol (MSNP)

Creado por Microsoft en 1999, utilizado por su famoso cliente Windows Live Messenger tuvo en su día gran acogida gracias a venir instalado junto al sistema operativo Windows, su última versión fue la MSNP24. Su código es de tipo propietario, los servidores utilizados en este protocolo son servidor de despacho (DS), servidor de notificación (NS) y servidor de tablero (SS). El proceso de comunicación inicia con la conexión al servidor de despacho (Messenger.hotmail.com puerto 1863) que reenvía al servidor de notificación en el que inicia el siguiente proceso (Mintz, 2003):

- Mensaje SOAP (*Simple Object Access Protocol)* con credenciales.
- Respuesta del servidor con fichero RST.srf
- Al autenticar se entrega lista de contactos con su estado.

Figura 8

MSN Messenger



Nota. Tomado de https://miracomohacerlo.com/wp-content/uploads/2019/05/windows-live-messenger-810-3-e1559201680309.jpg

Protocolo OSCAR

Ahora el protocolo Open System for Communication in Real Time (OSCAR) es un protocolo de tipo propietario de la compañía AOL que era utilizado por los clientes AIM e ICQ. Con similitud con el MSNP este protocolo utiliza varios servidores para realizar acciones específicas, estos servidores son el servidor de acceso (AS, login.oscar.aol.com) y el servidor principal(BOSS), el proceso sigue los siguientes pasos:

- Acceso al servidor AS.
- Entrega de cookie
- Ingreso al servidor BOSS
- Corroboración de credenciales
- Entrega de lista de contactos y servicios

Figura 9

MSN Messenger

Nota. Tomado *de* https://messenger.es/wp-content/uploads/2010/04/icqcl.jpg

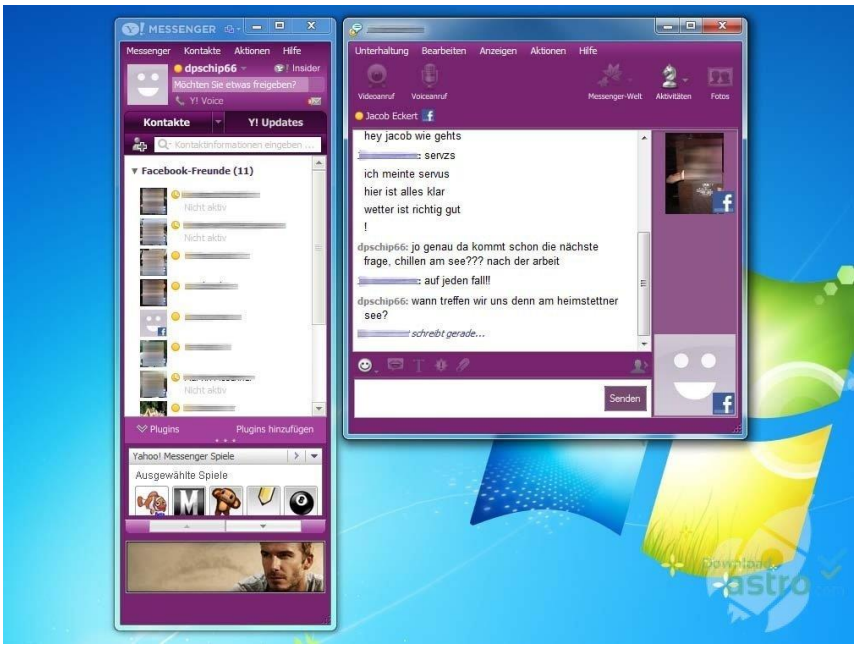
Protocolo YMSG

Publicado en 1999, el protocolo YMSG pertenece a Yahoo! Y utilizaba el cliente Yahoo! Messenger que era muy similar al Windows Live Messenger pero se distinguía con su capacidad de adaptarse con la web. Su arquitectura es también de tipo cliente-servidor pero con la diferencia de que el servidor al que se conecta el cliente es al azar y no definido, esto se logra al tener un dominio del siguiente tipo csNN.msg.dcn.yahoo.com en donde NN es un número cualquiera (Tellis, 2010).

En este protocolo no se utiliza cifrado de la comunicación es por ello que se excluyen clientes que utilizan cifrado SSL.

Figura 10

Yahoo! Messenger



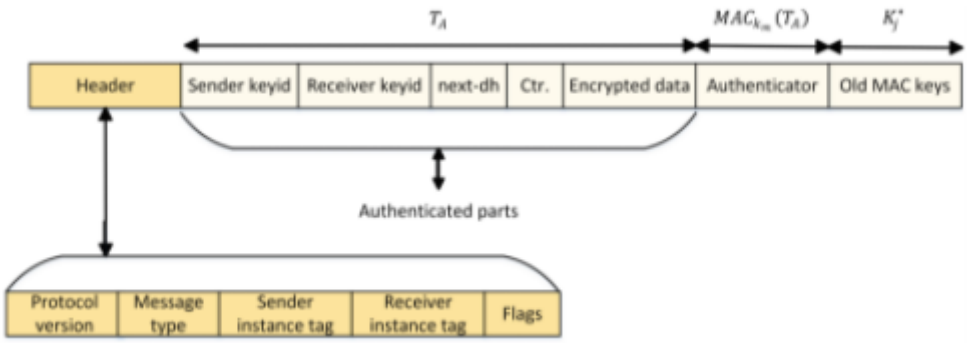
Nota. Tomado de https://i0.wp.com/webadictos.com/media/2016/06/yahoomessenger-old.jpg?w=1006&ssl=1

Protocolo OTR

OTR (off-the-record) que en español significa fuera de registro se trata de un protocolo de seguridad que propone crear sesiones privadas en la comunicación de mensajería instantánea, promueve confidencialidad de punto a punto mediante encriptación y autenticación de la transmisión de mensajes. Tiene una característica de denegación que hace que ni el emisor pueda probar quién fue el autor del mensaje. Fue creado por Borisov en 2004, utiliza una abstracción de alto nivel como AES (128-bit), SHA-256 y SHA-1, el uso de llaves de autentificación se hace mediante DH Share (ByungHoon & Kim, 2017).

Figura 11

Formato de paquete de datos OTR



Nota. Tomado de *(ByungHoon & Kim, 2017)*

Protocolo Skype

Este protocolo permite que sus usuarios realicen llamadas, videoconferencia y, por supuesto, comunicación por texto. Su arquitectura es la P2P, aunque para la autentificación utiliza arquitectura cliente – servidor. Este protocolo utiliza dos tipos de nodos, hosts y supernodos, los primeros corresponden a los clientes y los segundos a los dispositivos que realizan la gestión de envío de mensajes. En cuanto a la seguridad, Skype utiliza cifrado, pero al usar P2P no pueden ocultar las ip de los actores de la conversación y quedan vulnerables.

Figura 12

Skype interfaz

Nota. Tomado de *shorturl.at/abrul*

Protocolo XMPP

XMPP es como todos los otros protocolos, define un formato para transferir datos entre dos o más entidades que se comunican entre sí, siendo estas un cliente y un servidor para este caso. Existen múltiples servidores XMPP en internet que son accesibles para todas las personas formando una red interconectada de sistemas. Este protocolo utiliza XML para transferir la data por ello puede tomar ventaja de la gran cantidad de conocimiento y soporte de la comunidad acerca del Lenguaje de Marcado Extensible (Moffitt, 2010).

El uso del XML le permite extenderse fácilmente, añadiendo características que son retrocompatibles con versiones previas son dificultad, es por ello que más de 200 extensiones del protocolo existen en la actualidad y están registrados con los estándares del XMPP, esto les da a los desarrolladores una cantidad prácticamente ilimitada de herramientas y soluciones (Moffitt, 2010).

Los *stanza* son un grupo de líneas de código agrupadas en un script o archivo de configuración, es importante destacar este concepto ya que a partir de este momento se utilizará en gran medida para comprender más a fondo el protocolo XMPP, en este ámbito se pueden numerar tres tipos de *stanzas: message stanza, presence stanza* y *IQ stanza* (Li, s.f.).

Estos conforman el núcleo del protocolo ya que contienen la información sobre la disponibilidad de las otras entidades de la red, mensajes personales, o estructuras de comunicación que serán procesadas por computadora. A continuación, se muestra un ejemplo de esto (Moffitt, 2010):

Figura 13

Ejemplo de stanza

```
<message to='elizabeth@longbourn.lit'
         from='darcy@pemberley.lit/dance'
         type='chat'>
  <body>What think you of books?</body>
</message>
```

Nota. Tomado de *(Moffitt, 2010)*

Historia

La tecnología Jabber, luego renombrada como XMPP, fue inventada por Jeremie Miller en el año de 1998 al cansarse de utilizar hasta cuatro diferentes clientes para mensajería instantánea cerrada. En 1999 lanzó un servidor de código abierto llamado jabberd, esto llamó la atención de diferentes desarrolladores qué, al ver esta gran iniciativa, decidieron ayudar creando clientes para Linux, Machintosh y Windows que se podían utilizar mediante librerías en Perl o Java (Saint-Andre, Smith, & Tronçon, 2009).

En 2001 se creó la Jabber Software Fundation de manera que continúe el soporte para este protocolo, no fue sino hasta 2007 que se cambió el nombre al ahora conocido XMPP Standards Foundation creando

un gran número de extensiones al núcleo del protocolo. Luego de varios años de desarrollo y esfuerzo finalmente fue incluido el protocolo en el IETF (Internet Engineering Task Force), fuerza que ya ha estandarizado la mayoría de los núcleos de internet como los protocolos TCP/IP, HTTP, SMTP, POP, IMAP y SSL/TLS. Tras de ello en 2004 esta entidad publicó la estandarización del protocolo XMPP con los RFC 9320 y RFC 3921 (Saint-Andre, Smith, & Tronçon, 2009).

Características

Las características del XMPP se pueden separar en dos grandes temas, esto permite un entendimiento más acertado del protocolo, es por ello que se especificaran cada uno de ellos a continuación:

Servicios

- *Encriptación de canal*: Crea una encriptación de la conexión entre dos servidores o entre cliente y servidor.
- *Autenticación*: Asegura que los dispositivos que intentes comunicarse en la red sean autenticados por el servidor.
- *Presencia*: Se encarga de notificar a un dispositivo si otro se encuentra activo o no, si está activo si ya se encuentra en otra conversación, etc.
- *Lista de contactos*: Se trata de un conjunto seleccionado de contactos que el usuario considera amigos, conocidos o cercanos con los que tiene la intención de iniciar en algún momento una comunicación.
- *Mensajería punto a punto*: Permite a dos dispositivos establecer una comunicación e intercambio de mensajes, estos pueden ser dos usuarios con sus clientes, o dos nodos, o dos servidores, o un servidor y un nodo, etc.

- *Mensajería de muchos*: Crea un cuarto en el que ingresen más de dos usuarios y puedan, libremente hablar entre todos.
- *Notificaciones*: Este servicio establece que el protocolo genera advertencias dirigidos a los distintos dispositivos de acuerdo a un evento que lo inicie.
- *Descubrimiento de servicio*: permite a un dispositivo conocer si otro dispositivo está haciendo uso de un servicio.
- *Capacidad de publicidad*: proporciona una notación abreviada para los datos de descubrimiento de servicios para que pueda almacenar en caché fácilmente las funciones que son compatibles con otras entidades en la red.
- *Formularios estructurados de datos*: Logra que distintas entidades compartan información ordenada en formularios.
- *Administración de flujo de trabajo*: manejo del flujo de trabajo permite que una entidad interactúe con otra dependiendo de un evento.
- *Sesiones de medios de igual a igual*: Administra y permite crear sesiones con un dispositivo para utilizarse en sesiones de transferencia de archivos, chat de voz, chat de texto, entre otros (Saint-Andre, Smith, & Tronçon, 2009).

Aplicaciones

- *Mensajería instantánea*: En esta aplicación se utilizan tres servicios fundamentales, presencia, lista de contactos y mensajería uno a uno.
- *Chat grupal*: Se crea un grupo de usuarios que tienen una finalidad en común para esa comunicación, por ejemplo, un grupo de Whatsapp del trabajo.

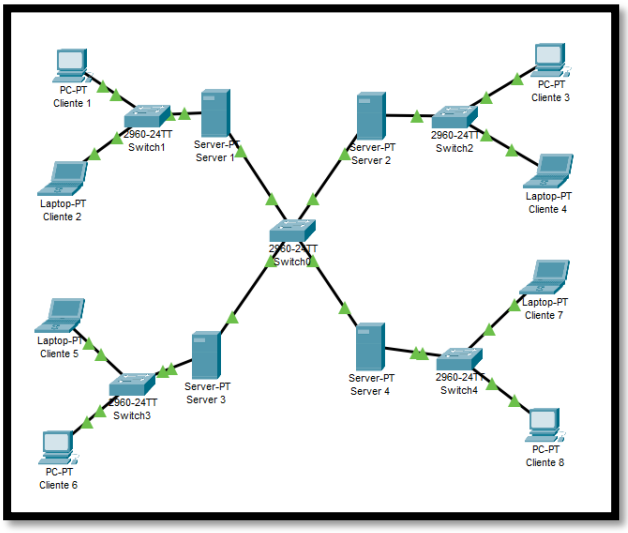
- *Videojuegos*: Se utiliza tanto la mensajería uno a uno como los chats grupales, es cada vez más una necesidad inherente para poder jugar en línea.
- *Sistemas de control*: Aplicaciones cuyo dominio incluyen manejo de red, telemetría científica y control robótico.
- Geolocalización: Permite generar aplicaciones basados en localización como por ejemplo rastreo vehicular (Saint-Andre, Smith, & Tronçon, 2009).

Arquitectura

XMPP utiliza una arquitectura del tipo cliente-servidor descentralizada, esto quiere decir que cada usuario está en la capacidad de crear su propio servidor y cliente, de esta manera también un equipo puede dedicarse por completo a realizar la mejor experiencia de interfaz con el usuario con el cliente, y otro equipo diferente dedicarse a optimizar hasta el mínimo detalle la parte del servidor. Esto da paso a la configuración que se puede ver en la figura 14, en ella se encuentran cuatro servidores con dos clientes cada uno (Saint-Andre, Smith, & Tronçon, 2009).

Figura 14

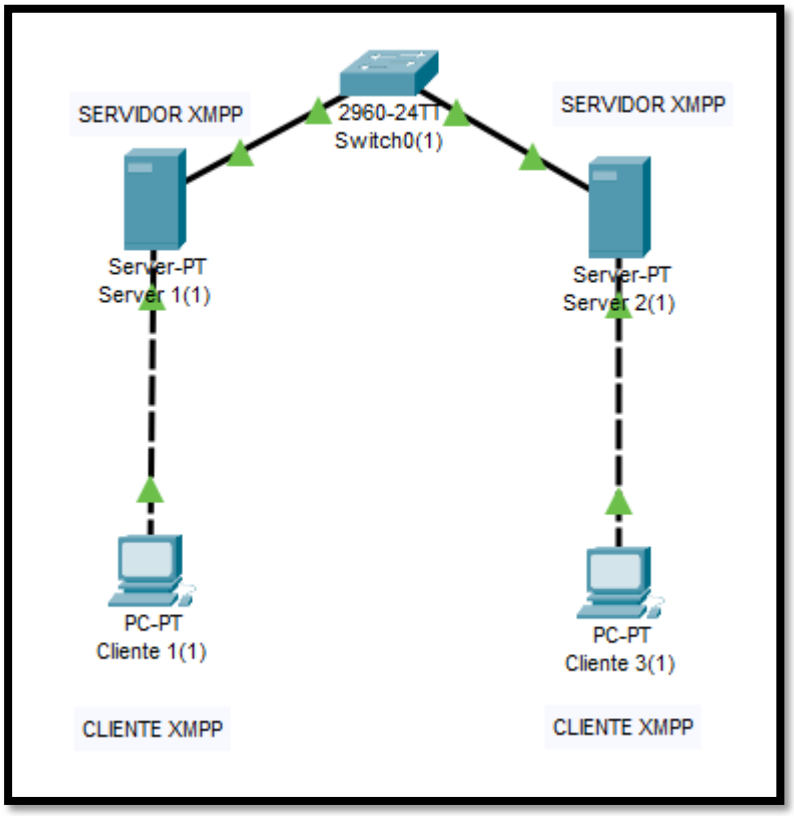
Arquitectura típica XMPP



Esto representa la arquitectura típica de los protocolos XMPP, ahora bien, en la siguiente figura se puede observar cómo es la comunicación entre dos usuarios teniendo distintos servidores cada uno.

Figura 15

Comunicación entre clientes XMPP.



En el momento que el cliente uno quiere hacer contacto con el cliente dos, el cliente uno se conecta con el servidor uno, este dirige el tráfico al servidor dos quién reenvía la información al cliente dos, esto se realiza mediante un identificador denominado JID, este tiene una composición similar a la de un correo electrónico como clinete1@server, donde clinete1 es el usuario y server es el dominio (Saint-Andre, Smith, & Tronçon, 2009).

Stanzas

Como se mencionó anteriormente, el protocolo XMPP utiliza stanzas para realizar la conversación entre cliente y servidor, estos contienen mensajes XML y pueden ser de tres tipos generalmente *stanzas: message stanza, presence stanza* y *IQ stanza,* a continuación, se explicará brevemente cada uno de ellos:

Stanza de mensaje

Como su nombre lo indica son utilizados para enviar mensajes de un cliente a otro, estos stanzas son dispara y olvida, es decir, que una vez enviado no recibe de vuelta una notificación de recibido, es por eso que no son confiables totalmente, si se desea añadir confianza se debe añadir capas de notificación en el protocolo de aplicación (Moffitt, 2010).

Figura 16

Message Stanza

```
<message from='bingley@netherfield.lit/drawing_room'
         to='darcy@pemberley.lit'
         type='chat'>
  <body>Come, Darcy, I must have you dance.</body>
  <thread>4fd61b376fbc4950b9433f031a5595ab</thread>
</message>

<message from='bennets@chat.meryton.lit/mrs.bennet'
         to='mr.bennet@longbourn.lit/study'
         type='groupchat'>
  <body>We have had a most delightful evening, a most excellent ball.</body>
</message>
```

Nota. Tomado de *(Moffitt, 2010)*.

En la figura 16 se puede observar un ejemplo de un *message stanza* en el que se puede destacar que se especifica el destinatario, el emisor y el tipo comunicación, bajo la etiqueta <body> se encuentra el cuerpo del mensaje que fue enviado.

Stanza de presencia

Su finalidad es la de indicar si una entidad está o no disponible, si no está disponible el porqué, es decir, no está conectado o está ocupado o lejos del dispositivo, etc.

Figura 17

Message Stanza

```
<presence/>

<presence type='unavailable'/>

<presence>
  <show>away</show>
  <status>at the ball</status>
</presence>

<presence>
  <status>touring the countryside</status>
  <priority>10</priority>
</presence>

<presence>
  <priority>10</priority>
</presence>
```

Nota. Tomado de *(Moffitt, 2010)*.

De la figura 17 es posible comprender que la etiqueta <show> indica el estado, la etiqueta <status> indica la razón del estado puesto por el usuario mediante un *string*, también se le puede asignar una prioridad con la etiqueta <priority> (Moffitt, 2010).

Stanza IQ

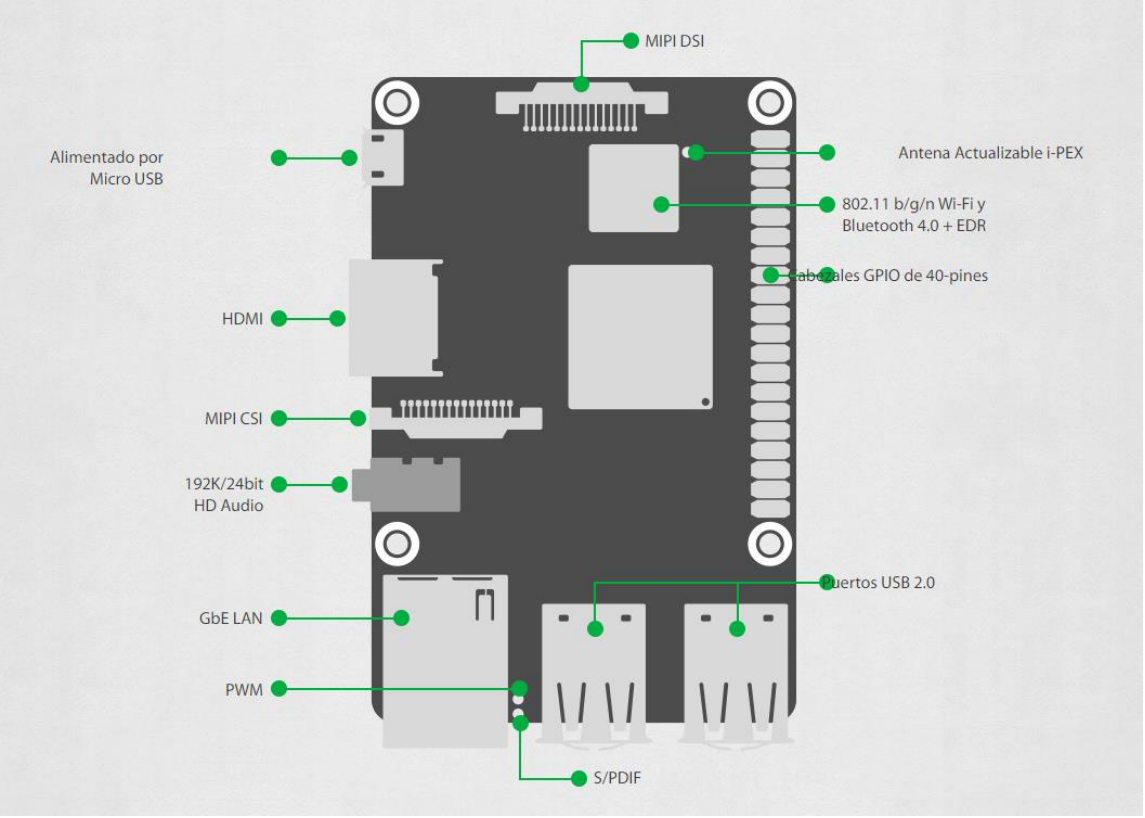
Se trata de un mensaje pero que sí espera una notificación de recepción del mismo, existen cuatro subtipos de este stanza que son get, set, result y error. Cada uno de estos con la finalidad que indican sus nombres respectivamente: solicita notificación de recepción, envía la notificación de recepción, resultado del get, y si la entidad a la que se le solicitó no responde (Moffitt, 2010).

Computadoras de placa única

De siglas SBC en inglés, Single Board Computer, es un ordenador completo construido en una sola tarjeta, dentro de la estructura de la placa se encuentra un microprocesador, memoria RAM y ROM, además de sus entradas y salidas correspondientes (Stone, 2019).

Figura 18

Computadora de placa única (Tinker Board)



Nota. Tomado de *(ASUS, s.f.)*

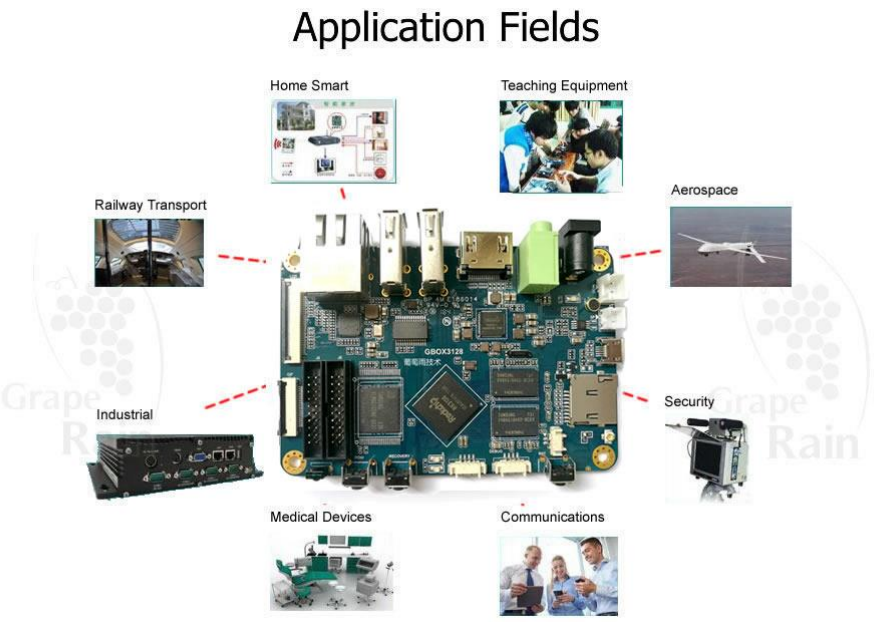
Aplicación

Estas computadoras tienen utilidad como controlador en sistemas que no requieren demasiada potencia del procesador, esto se debe a que los componentes son pequeños, para que tengan la capacidad de entrar en la misma placa, por ello su velocidad y el almacenamiento son reducidos, por lo que no es

común utilizarlas como computadoras personales ya que su rendimiento no es el adecuado. Su importancia está en el mercado IoT, estos ordenadores permiten tener información de forma automática por los sensores integrados (Zhang, 2010).

Figura 19

*Aplicaciones de c*omputadoras de placa única



Nota. Tomado de *https://www.graperain.com/ARM-Embedded-RK3128-Single-Board-Computer/*

Las computadoras de placas únicas brindan el servicio en tanto sistemas de control o automatización, además tienen ventajas frente a los ordenadores personales o de escritorio, también tienen algunas desventajas, a continuación: se presentan las principales (Zhang, 2010).

Tabla 4

Ventajas y desventajas de las computadoras de placa única

Ventajas	Desventajas
Tamaño: es reducido perfecto para sistemas integrados en espacios limitados	Reparación: es difícil debido a que los componentes no son modulares y si fallan el reparar es complicado, frecuentemente reemplazan toda la placa.
Costo: es bajo por tanto accesible	*Escalabilidad:* no existe, es prácticamente imposible dado que los componentes están soldados en la misma placa.
Eficiencia: es alta por tanto gasta menos energía y produce menos calor	
Aislamiento: es fil la protección de la placa debido al tamaño.	

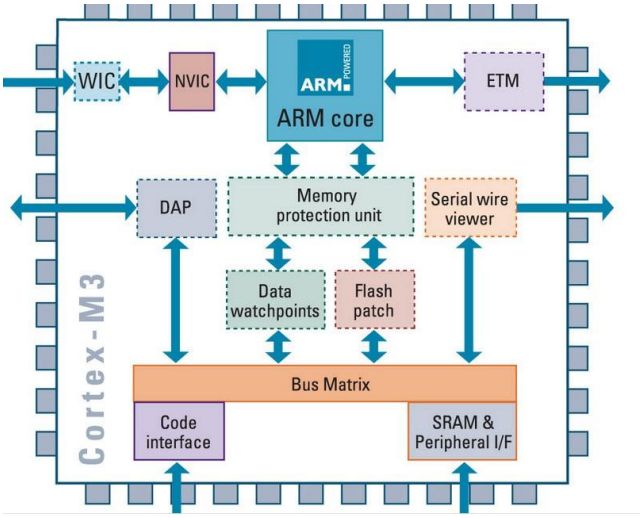
Nota. Tomado de *(Zhang, 2010)*

Arquitectura ARM

Por sus siglas en inglés ARM (Advanced RISC Machines) significa Máquinas Avanzadas RISC (Ordenador con Conjunto Reducido de Instrucciones), se utiliza con frecuencia en microprocesadores con instrucciones de tamaño fijo en un número reducido de formatos que permite operar a mayor velocidad realizando varias instrucciones por segundo, debido a que dichas instrucciones son simples solo necesitan de uno a dos ciclos de reloj, esto permite un gran rendimiento con una sola fracción de energía del procesador. (Complex instruction set computing-CISC) (Hachman, 2002).

Figura 20

Procesador ARM Arquitectura



Nota. Tomado de *shorturl.at/chP01*

Este excepcional rendimiento de los procesadores ARM hace que su mercado y nichos de mercado estén relacionados con pequeños aparatos electrónicos, el principal es el sistema de dispositivos IoT, seguido de las tabletas y teléfonos inteligentes. Este mercado es atractivo debido a que sus instrucciones requieren menos transistores, circuitos y por lo tanto un consumo bajo de energía.

Raspberry Pi

Es una SBC, es decir, una computadora de placa única de tamaño reducido y bajo costo con el fin de que las personas tengan acceso a la informática, un propósito entorno a la educación. Vio la luz en Reino Unido donde fue desarrollada por la fundación Raspberry Pi y lanzada a la venta en el 2012, su atractivo se debió en gran parte a su espectacular tamaño comparable al de una tarjeta de crédito, qué, aun así, era capaz de realizar algunas instrucciones. Debido a la modernidad y al avance de la misma tecnología, se han

lanzado varias versiones de la misma placa, con actualizaciones y componentes adecuados al auge digital, sin embargo, todas las versiones comparten las siguientes características (Stone, 2019):

Figura 21

Rapsberry Pi

Nota. Tomado de *shorturl.at/nWZ29*

Tabla 5

Características del Raspberry Pi

Criterio	Característica
Chipset Broadcom	BCM2835, que contiene un procesador central ARM1176JZF-S a 700 MHz.
Procesador Gráfico	GPU VideoCore IV
Memoria RAM	Módulo de 512 MB de memoria
Puerto de red	RJ45 integrado Ian9512-jzx de SMSC que permite velocidades de 10/100 MBPS
Buses de datos	Dos USB 2.0
Salidas	Analógica de audio estéreo por Jack de 3.5 mm. Digital de video y audio HDMI Analógica de video RCA
Pines	Entrada y salida de propósito general
Conector	Alimentación microUSB
Lector	Tarjeta SD
Conectividad	WIFI Bluetooth

Nota. Tomado de *(Stone, 2019)*

Historia

El proyecto vio la luz y fue desarrollado en la Universidad de Cambridge en el año 2006, sin embargo, se introdujo al mercado en el 2012 con el objetivo de acercar a los niños al aprendizaje de las ciencias computacionales, por lo que para que empresas como Google han realizado donaciones para que llegue a varias instituciones educativas y propicie el aprendizaje en electrónica y programación. En sus inicios utilizaban el microcontrolador Atmel Atmega644 que tiene la descarga del diseño disponible a nivel público (Petrikowski, 2014).

Para el 2009 se crea la fundación Raspberry Pi como una asociación sin fines de lucro con el fin de llevar el aprendizaje de la informática en los niños con ordenadores baratos. El primer prototipo tuvo base en ARM que fue incorporado en un paquete de tamaño reducido con un puerto USB y HDMI (Petrikowski, 2014).

En el 2011 desarrollaron 50 placas de Modelo Alpha en agosto, cuatro meses después realizaron 25 placas con modelo Beta. En este mismo mes se subastaron 10 placas en la plataforma de comercialización eBay. Posteriormente salió al mercado un lote de 10 000 placas fabricadas en el continente asiático para abaratar el costo de producción en el 2012 y meses después del lanzamiento llegaron a vender 500 000 unidades (Petrikowski, 2014).

Componentes

La Raspberry Pi tiene en su placa los mismos componentes que una computadora de escritorio como puertos, pines, salidas, conectores, procesador, etc (Raspberri Pi, s.f.).

- **CPU:** Esta unidad de procesamiento utiliza un ARM de sexta generación con 700 Mhz de base y una frecuencia de 1Ghz.

- **GPU:** el procesamiento gráfico es básico, sin embargo, soporta la demanda de video ya que es un Core VideoCore IV Multimedia con núcleo 3D y soporte para librerías OpenGL ES2.0 y OpenVG. Para la salida del video tiene un conector RCA, HDMI y una interfaz DSI (permite conexión de pantallas parecidas a las de celulares). Se tiene una mejor calidad de imagen con el HDMI con resolución 1920*1080 pixeles comúnmente llamado Full HD.
- **AUDIO:** Para el audio cuenta con un Jack de 3,5 y HDMI.
- **RAM:** Memoria de 512 MB SDRAM de módulo único, frecuencia de 400MHz normal y turbo 600MHz.
- **Almacenamiento:** necesita una tarjeta microSD de mínimo 8GB donde se instalará el sistema operativo y almacenará programas-archivos. Cabe recalcar que no cuenta con almacenamiento interno.
- **Entradas y salidas de propósito general:** la placa cuenta con puertos USB para teclado. mouse y memoria USB el modelo B tiene dos puertos y el A solo uno. La característica de la Raspberry son los puertos GPIO que son digitales y tiene interfaces de comunicación que tienen la característica de poder ser activados según el usuario.
- **Conectividad:** Tiene un conector RJ-45 conectado a un integrador LAN modelo 9512-JZX que genera 10/100 Mbps de velocidad, permite una conexión directa a una PC o Router. El DHCP se activa por defecto, se puede asignar una dirección ip estática al configurar la red. También está la conectividad integrado al módulo WI-FI como conexión inalámbrica, si no tiene esta característica se puede utilizar un adaptador externo.
- **Alimentación:** no existe la opción de encendido o reinicio en ninguna versión actual, por lo que para apagar de debe desconectar la fuente, sin embargo, no se recomienda, solo si es un caso

extremo porque se puede dañar. La alimentación es mediante una entrada microUSB de 5v y mínimo 700mA.

Sistema operativo y arquitectura

Como se mencionó anteriormente y al ser una computadora de placa única se basa en la arquitectura ARM gracias a que permite tener un consumo de energía bajo, se utiliza también en teléfonos inteligentes, no es muy usual en computadoras de escritorio. Este modelo no tiene la misma arquitectura que una PC, por lo que no tienen los mismos sistemas operativos, sin embargo, esto no se convierte en una desventaja ya que los programas de una PC tienen versiones que son adaptables al modelo Raspberry (Carles, 2020).

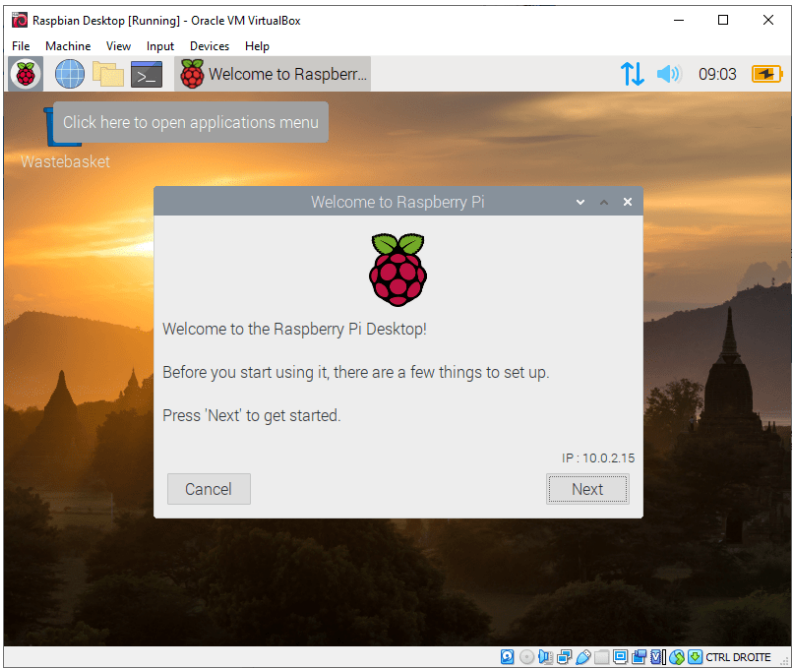
Para instalar un sistema operativo es necesaria una tarjeta microSD, tiene los siguientes pasos:

- Obtener una imagen del sistema operativo
- Copiar la ISO en la tarjeta microSD
- Insertar la tarjeta en la Raspberry Pi y prender el dispositivo.
- Seguir los pasos de instalación según el manual del SO.

El sistema operativo más común utilizado es Raspbian que tiene su base en Linux conocida como Debian pero optimizada para que sirva en la arquitectura ARM, tiene la opción de ser modificada por su código abierto, la versión puede ser descargable desde la página oficial. Gracias a su naturaleza de código abierto y popularidad existe gran cantidad de información y desarrollo de software que le da una versatilidad importante.

Figura 22

Sistema operativo Rapsbian



Nota. Tomado de *shorturl.at/stTX1*

Lenguaje de programación en Raspberry

El lenguaje de programación más utilizado en las Raspberry pi es Phyton, ya que es un lenguaje de alto nivel cuya sintaxis es fácil de utilizar. Este lenguaje soporta una programación orientada a objetos, imperativa, funcional, entre otros. Esto lo convierte en multiparadigma y esencial para aplicar en una computadora de placa única (Stone, 2019).

Modelo más actual Raspberry Pi 3 B+

Cuando se habló de los raspberry se mencionó que existían varias versiones de esta, que han ido mejorando con el tiempo gracias al aporte en desarrollo importante que se ha realizado. El modelo más actual es el Raspberry Pi 3 B+, y resulta relevante comprenderlo ya que será el utilizado en este trabajo de investigación. El modelo B+ es una versión mejorada de la B a un precio casi idéntico, a continuación, se muestra una pequeña comparación de ambas (Raspberri Pi, s.f.):

Tabla 6

Comparativa Raspberry Pi 3 B y B+

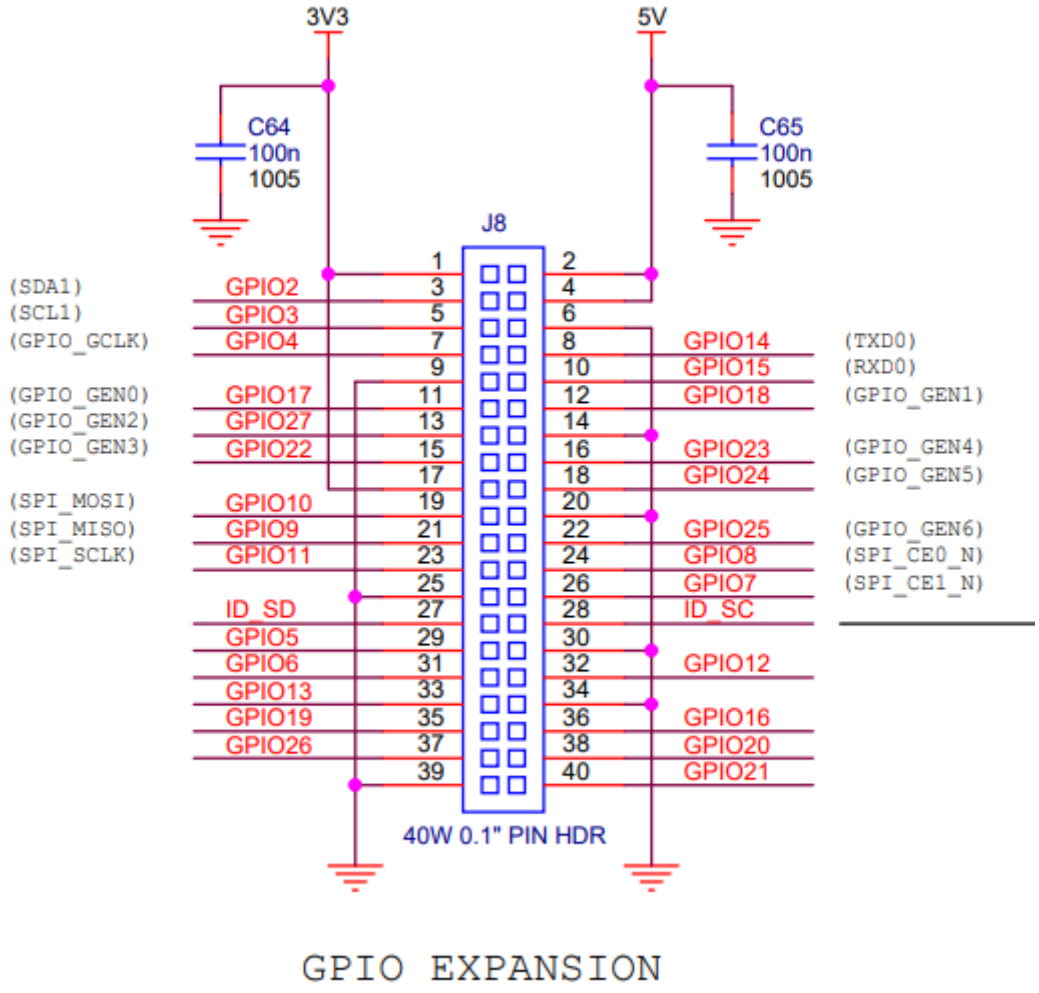
ESPECIFICACIONES	**RASPBERRY PI 3 MODEL B+ (2018)**	**RASPBERRY PI 3 MODEL B (2016)**
PROCESADOR	Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz	Broadcom BCM2837, Cortex-A53 (ARMv8) 64-bit SoC @ 1.2GHz
RAM	1GB RAM	1GB RAM
CONECTIVIDAD	WiFI 802.11.b/g/n/ac de doble banda 2.4GHz y 5GHz Bluetooth 4.2 Puerto Ethernet de hasta 300Mbps	WiFi 802.11 b/g/n (2.4GHz) Bluetooth 4.1 Puerto Ethernet de hasta 100Mbps
PUERTOS	HDMI completo, 4 USB 2.0, Micro SD, CSI camera, DSI display	HDMI completo, 4 USB 2.0, Micro SD, CSI camera, DSI display

Nota. Tomado de *(Yúbal, 2018).*

Es relevante comprender su disposición GPIO ya que se utilizarán en el desarrollo de la investigación.

Figura 23

Descripción de pines



Nota. Tomado de *(Raspberri Pi, s.f.)*

Capítulo III. Diseño

Diseño de la red IoT

En esta sección del trabajo de investigación se hablará sobre el diseño propuesto de la red de Internet de las cosas y el desarrollo de su aplicación, es decir, se explicarán sus partes, funciones, aplicaciones, usos, etc.

Partes de la red IoT

Esta red se diseñó con cinco puntos o denominados también nodos, estos tienen diferentes características que les permiten realizar diferentes acciones, mismas que son necesarias para la consecución del objetivo. Estos nodos se conectan a internet de manera alámbrica o inalámbrica según requiera el caso. A continuación, se detallan más profundamente cada una de estas partes o elementos.

Cliente

Los clientes son aquellos dispositivos (celulares, computadores, entre otros) que visualizan los datos obtenidos de los recolectores o nodos IoT después de ser procesados por el servidor. Pueden alterar las configuraciones de estos nodos de manera remota. Deben tener instalado un cliente XMPP que será el software que se comunique con el servidor y los nodos.

Nodo IoT

Estos nodos tienen la capacidad de recolectar información y transferirla, además pueden solicitar los datos obtenidos de otros nodos o servidores y modificar las configuraciones tal como puede hacerlo el cliente. Tienen instalado el cliente XMPP para realizar la comunicación, también cuentan con sensores para la recolección de datos.

Nodo Simple IoT

Estos nodos a diferencia de los anteriores solo tienen la capacidad de obtener datos mediante los sensores que llevan instalados, luego pre-procesan esta información y la envían a través de XMPP. Pueden responder solicitudes de acuerdo a la programación que se les realizó, tienen también la capacidad de cambiar la configuración de los sensores que llevan instalados.

Servidor XMPP

Su función principal es la de permitir el inicio de sesión de los dispositivos de acuerdo a sus credenciales, para ello tendrá instalado el programa tipo servidor de XMPP. Tiene la base de datos con la información y función de cada nodo que compone la red. Almacena y gestiona la información obtenida por los recolectores.

Servidor de análisis

Tiene la función de procesar la información de la base de datos de manera que pueda enviar alertas por si algo está fuera de los rangos permitidos para su funcionamiento. Para comunicarse tendrá instalado un cliente XMPP y también un gestor de base de datos para almacenar toda la información procesada, aquí se agregará la programación requerida para el análisis de los datos. Puede solicitar la información a los nodos a través de la comunicación XMPP.

Funcionamiento

El funcionamiento de esta red IoT se diseñó para funcionar desde un caso simple con pocos nodos, hasta algo complejo con cientos de nodos y decenas de servidores. Todo manteniendo la relación en capacidad de cada servidor para manejar cierto número de nodos, así como también controlando la capacidad de almacenamiento de información de los servidores de análisis. De esta manera, para que dicha

red funcione sin inconveniente independiente de su complejidad se propone el siguiente algoritmo de funcionamiento:

Si el cliente solicita a un nodo una información que le compete a un nodo diferente, el primer nodo tiene la capacidad de reenviar la petición al nodo que sí maneja dicha información. De no existir ningún nodo que maneje aquella petición se notifica al cliente ello.

Clasificación

La red IoT puede clasificarse de acuerdo a su grado de complejidad, es decir, los diferentes elementos y características que la componen, a continuación, se detalla las tres categorías que son simple, media y alta.

Complejidad simple

Se trata de la red con menor complejidad ya que solo comprende de 1 a 10 nodos IoT (simples o normales) y en misma cantidad los clientes, estos se conectan a un solo servidor XMPP. Por su extensión no es necesario utilizar servicios pagos de XMPP ya que el número de cuentas que se deben crear es bajo. Es mayormente utilizado para proyectos menores tal como es el caso de este trabajo de investigación, es por ello que esta será la complejidad del proyecto a desarrollarse.

Complejidad media

Se considera de complejidad media cuando en la red se encuentra ya un servidor de análisis dentro de su topología, por ello los elementos que comprende son de 10 a 100 nodos IoT y en igual cantidad los clientes, un servidor de comunicación XMPP y un servidor de análisis. Diseñada para proyectos de mediano alcance en los que es posible utilizar aún servicios gratuitos de servidor XMPP pero es recomendable tener uno propio.

Complejidad alta

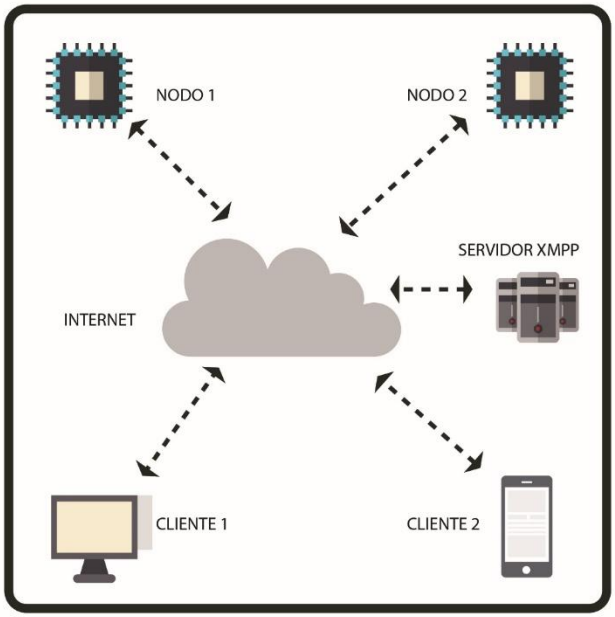
En esta instancia es absolutamente necesario contar con servidor de análisis y XMPP propios que manejen la comunicación y el almacenamiento de la información. Puede llegar a tener más de 100 nodos IoT y más de 100 clientes, dependiendo de ello será el número de servidores XMPP. Específico para proyectos de gran alcance.

Diseño utilizado para el proyecto

Como se mencionó anteriormente se utilizará para este proyecto una red IoT de complejidad baja por su practicidad, por ello la topología seleccionada contará con dos nodos IoT, dos Clientes y un Servidor XMPP gratuito. Los nodos IoT serán computadores de placa única Raspberry Pi y los clientes será uno un computador con Windows 10 y otro un celular inteligente con sistema operativo Android.

Figura 24

Topología red IoT del proyecto



En la figura 24 se muestra la topología que se utilizará en el trabajo de investigación, ahora corresponde detallar cada uno de las partes o nodos que conforman la red. Como se puede notar cada nodo puede estar ubicado en un lugar totalmente diferente de otro, tan solo necesita estar conectado alámbrica o inalámbricamente a internet.

Tabla 7

Descripción de nodos

Nodo	Descripción
Nodo 1	SBC Raspberry Pi 3b+. Con sensor de temperatura y tres actuadores. Sensor 1, actuador 1, 2 y 3.
Nodo 2	SBC Raspberry Pi 3b+. Con sensor de temperatura y tres actuadores. Sensor 2, actuador 4, 5 y 6.
Cliente 1	Computador con Windows 10.
Cliente 2	Teléfono inteligente con sistema operativo Android
Servidor XMPP	Servidor gratuito y público lightwitch.org

Funcionamiento

Los pasos que el sistema seguirá para funcionar adecuadamente son los siguientes:

1. Los nodos IoT obtienen constantemente los datos mediante su sensor de temperatura mientras sus actuadores cambian su estado según se necesite en ese momento.
2. Cada nodo identifica su sensor y actuadores de acuerdo al número designado, de igual manera reconoce los del otro nodo. Tienen guardados los JID de XMPP propio y de cada nodo de la red.

3. Cuando un dispositivo se activa inicia sesión en el servidor XMPP de acuerdo a sus credenciales y su estatus en la red será activo disponible.
4. Los clientes solo tendrán almacenados los JID de XMPP de los nodos IoT, es decir, no conocerán los números de cada sensor o actuador.
5. Los clientes pueden en cualquier instante solicitar información recolectada por los sensores de los nodos o cambiar el estado de los actuadores.
6. Si la solicitud se la realizó al nodo correcto, es decir, al nodo que controla el sensor o actuador solicitado por el cliente, el nodo responde directamente la solicitud a través del servidor XMPP.
7. En caso de que la solicitud se la realizó al nodo incorrecto, este enviará la petición al nodo cuyo sensor o actuador sea el requerido.
8. El cliente consigue la información o cambio requerido, pero desconoce qué nodo posee dicho sensor o actuador.

Lista de componentes

Los componentes que conforman los nodos de la red IoT son los siguientes:

- Nodo IoT
    - Dos protoboards
    - Dos Raspberry Pi 3b+
    - Dos sensores DS18B20 (de temperatura)
    - Cuatro resistencias un par de 4.7 kOhm y otro de 330 Ohm
    - Dos leds RGB ánodos comunes y seis de color verde
- Clientes
    - Pc con Windows 10
    - Teléfono inteligente con Android

Configuraciones

Antes de realizar la programación de los algoritmos de funcionamiento del sistema se debe configurar los diferentes dispositivos para ello, esto incluye instalación de sistema operativo y conexión de la circuitería de los actuadores y sensores de temperatura.

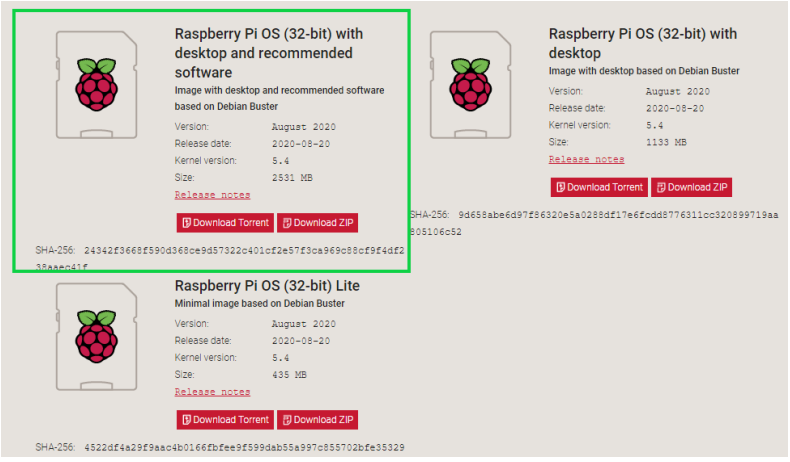
Configuración de la Raspberry Pi 3b+

La configuración de la computadora Raspberry empieza con la instalación del sistema operativo. El SO seleccionado es el Raspbian, que es una versión de Debian hecho especialmente para la Raspberry, es por esta razón que fue el elegido para el proyecto, ya que cuenta con mucho trabajo de optimización en él, además que es posible instalar el cliente XMPP.

1. Lo primero que se debe hacer es descargar el SO desde la página oficial: https://www.raspberrypi.org/downloads/raspbian, existen tres versiones disponibles como se puede evidenciar en la figura 25, se debe seleccionar “Raspbian Stretch with desktop and recomended software” y empezará la descarga en .zip.

Figura 25

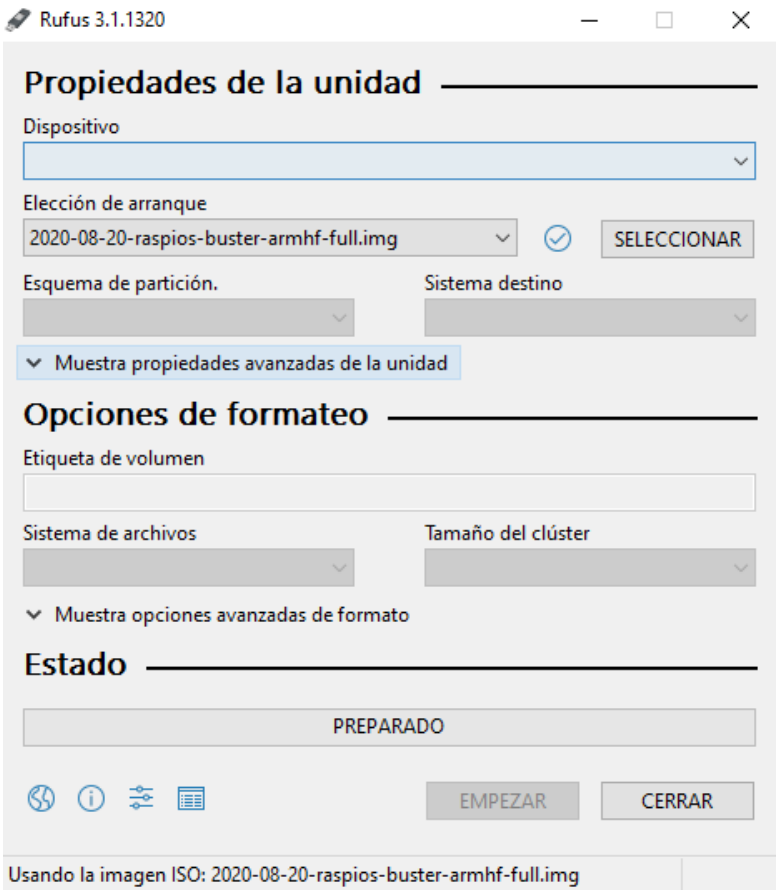
Versiones Raspbian



2. Como segundo paso se debe preparar la tarjeta micro SD en la que estará instalado el SO, la tarjeta debe cumplir con requerimientos de calidad para que el sistema operativo pueda funcionar correctamente, por ello se utilizó para este proyecto una tarjeta SD de gama media. Para poder instalar el sistema operativo se debe utilizar un software que hace booteable a la tarjeta SD, este es Rufus y es gratuito su uso, se lo debe descargar del siguiente enlace: https://rufus.akeo.ie/. En la figura 26 se puede ver la ventana que se abre cuando se ejecuta el programa, en dispositivo se selecciona la tarjeta SD, luego en elección de arranque se elige la imagen del SO Raspbian y finalmente se da en el botón empezar.

Figura 26

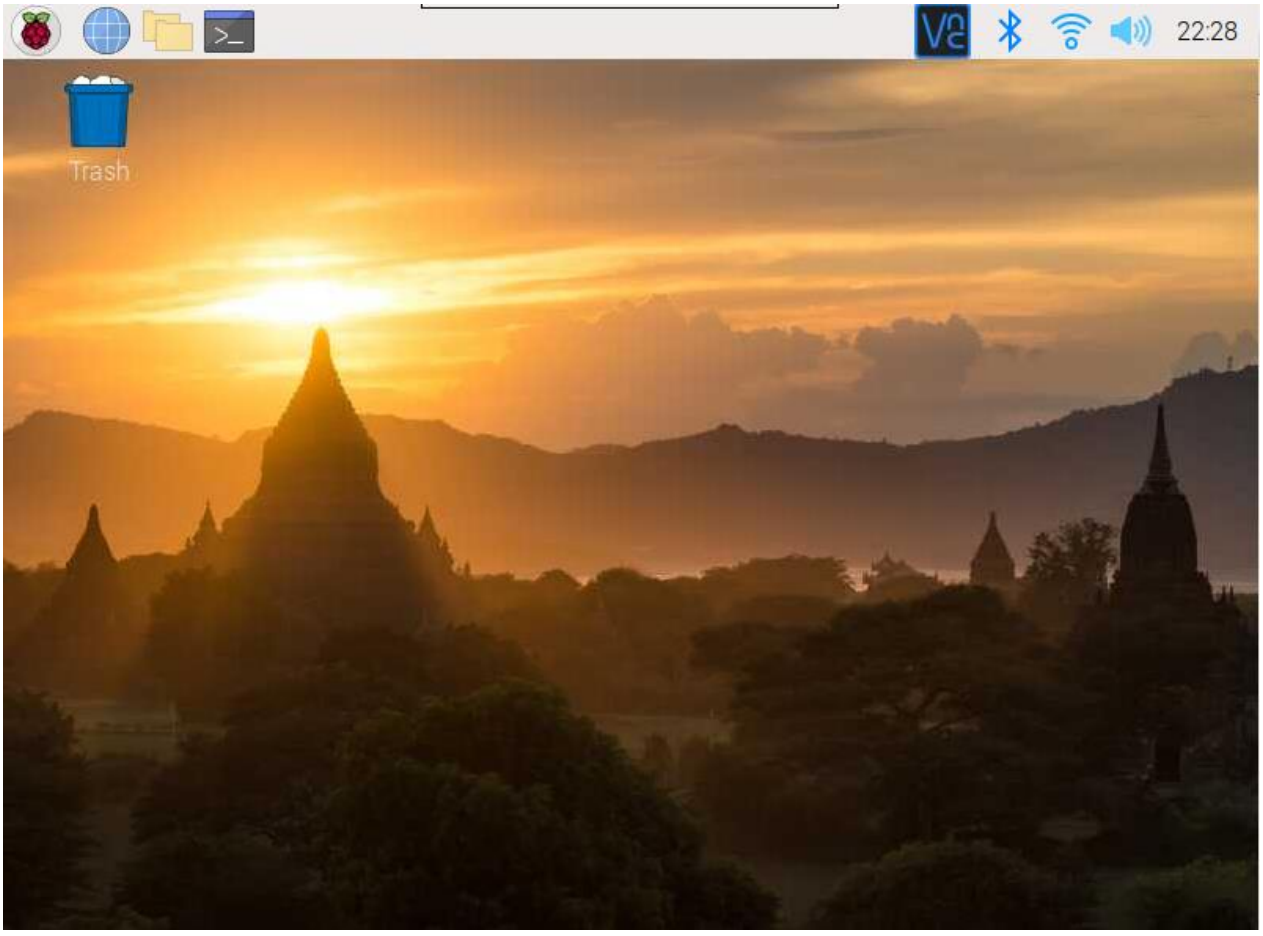
Ventana Rufus



3. Con la tarjeta SD preparadas continúa la instalación del sistema operativo, para ello se coloca la tarjeta en el Raspberry, además, se conectan monitor, teclado y mouse. Se enciende el dispositivo y este pide las credenciales que por defecto son usuario “pi” y contraseña “raspberry”. Tras autenticarse correctamente se inicia la instalación, durante este proceso se solicita configuraciones como la hora, idioma, etc. Finalmente, el sistema se configura y queda con el ambiente de escritorio que se puede observar en la figura 27. Es importante ahora conectar el dispositivo a internet.

Figura 27

Escritorio Raspbian



4. Una vez instalado y funcional el sistema operativo se deben configurar el acceso remoto al dispositivo, de manera que no sea necesario utilizar monitor, teclado y mouse conectados

directamente al Raspberry para utilizarlo. Aunque existen varios programas para lograr este cometido, se han seleccionado dos para aplicarlos en el presente proyecto, estos son VNC (Virtual Network Computing) y SSH (Secure Shell).

- **VNP:** Esta es una aplicación que permite trabajar remotamente en el escritorio de un dispositivo que tenga instalado VNP Server a través de VNP Viewer. Se eligió esta opción ya que viene instalada por defecto en el SO Raspbian, únicamente debe ser activado desde la terminal y los pasos de estos se indican a continuación:

  Se ingresa a la terminal para escribir los comandos, es necesario actualizar mediante el comando de la figura 28 y luego actualizar el VNP con el comando de la figura 29.

Figura 28

Comando instalar VNP server

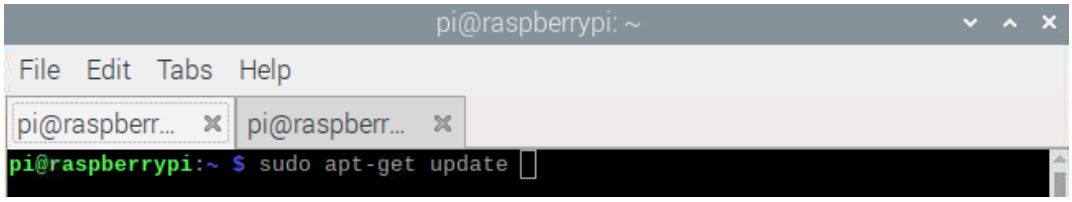
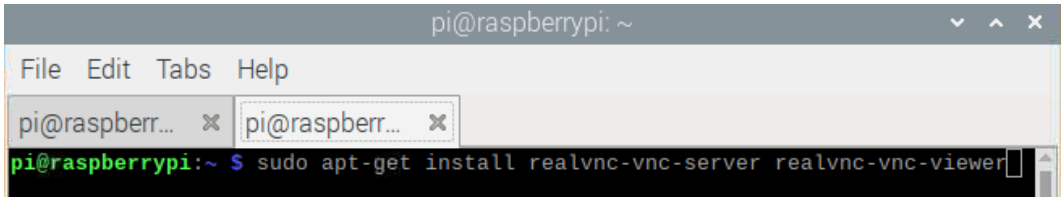


Figura 29

Comando actualizar raspberry



Una vez hecho esto se necesita activarlo, para ello se ingresa el comando de la figura 30 que abre la configuración, dentro de esta se habilita el VNP como se muestra en la figura 31.

Figura 28

Comando configurar raspberry

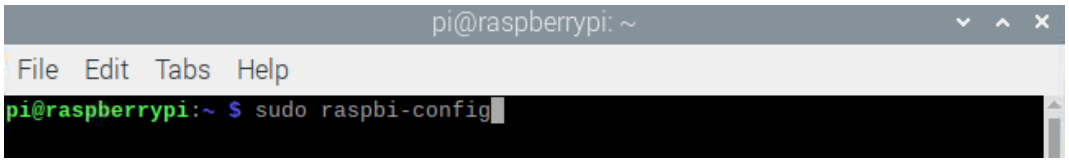
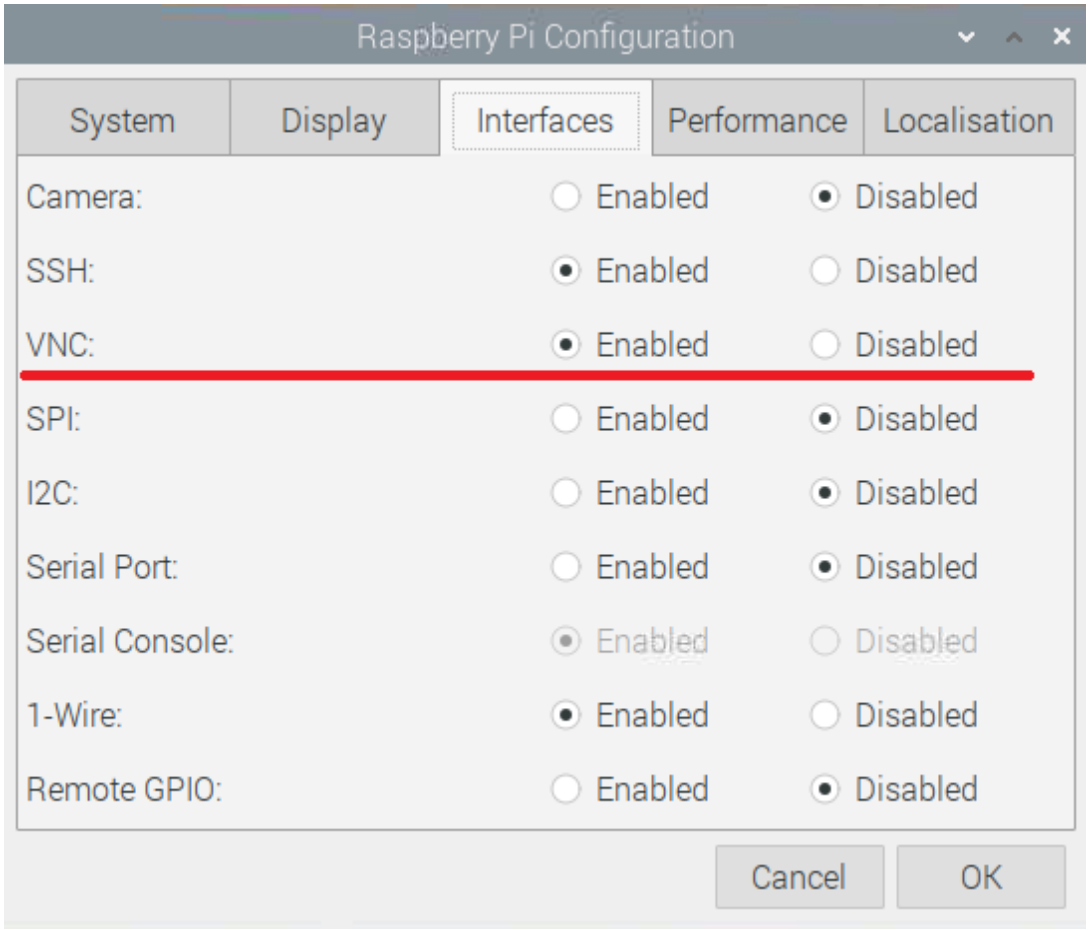


Figura 29

Habilitar VNP

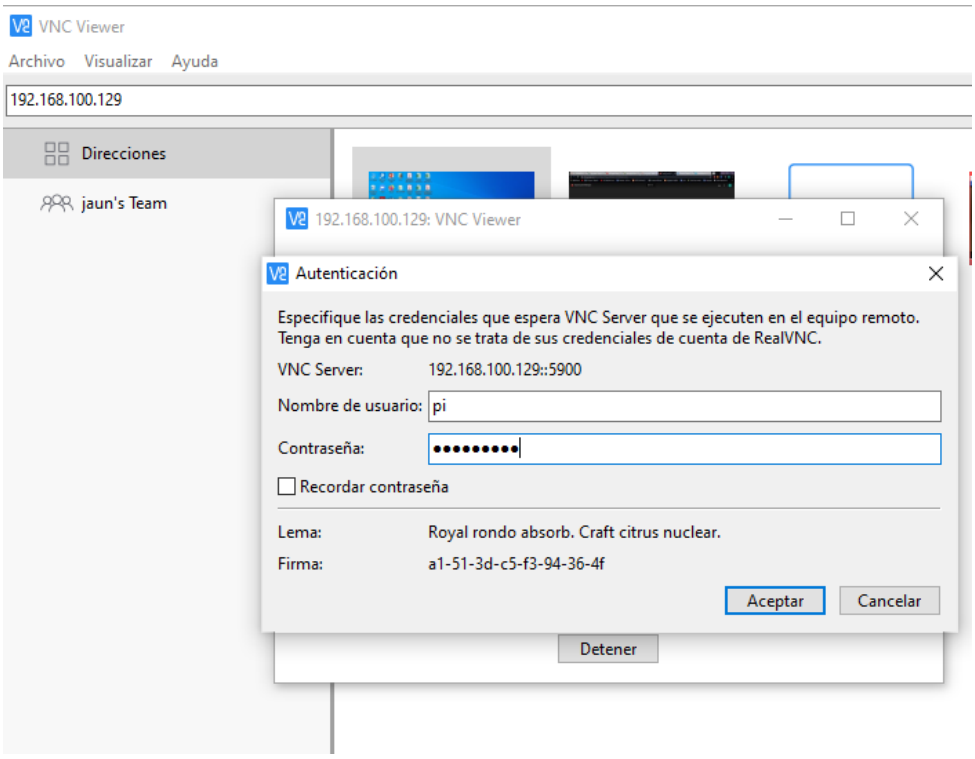


Para instalar el NVP viewer en el computador se necesita visitar el siguiente enlace: https://www.realvnc.com/es/connect/download/viewer/ descargar la aplicación e instalarla según los pasos indicados. Ahora para realizar la conexión entre ambas, raspberry

y computador, es necesario obtener la dirección ip de la raspberry, si no se le ha asignado una estática se puedo conseguir mediante el comando - *#sudo ifconfig.* Cerciorándose de que la raspberry se encuentre prendida se abre el VNP viewer en la computadora, el programa solicitará la dirección ip, al ingresarla correctamente necesitará que se ingrese el usuario y contraseña de la raspberry como se ve en la figura 32. Si las credenciales son correctas se realiza de manera exitosa la conexión y en una ventana se muestra el escritorio de la raspberry en el computador.

Figura 30

VNP viewer ingreso de credenciales

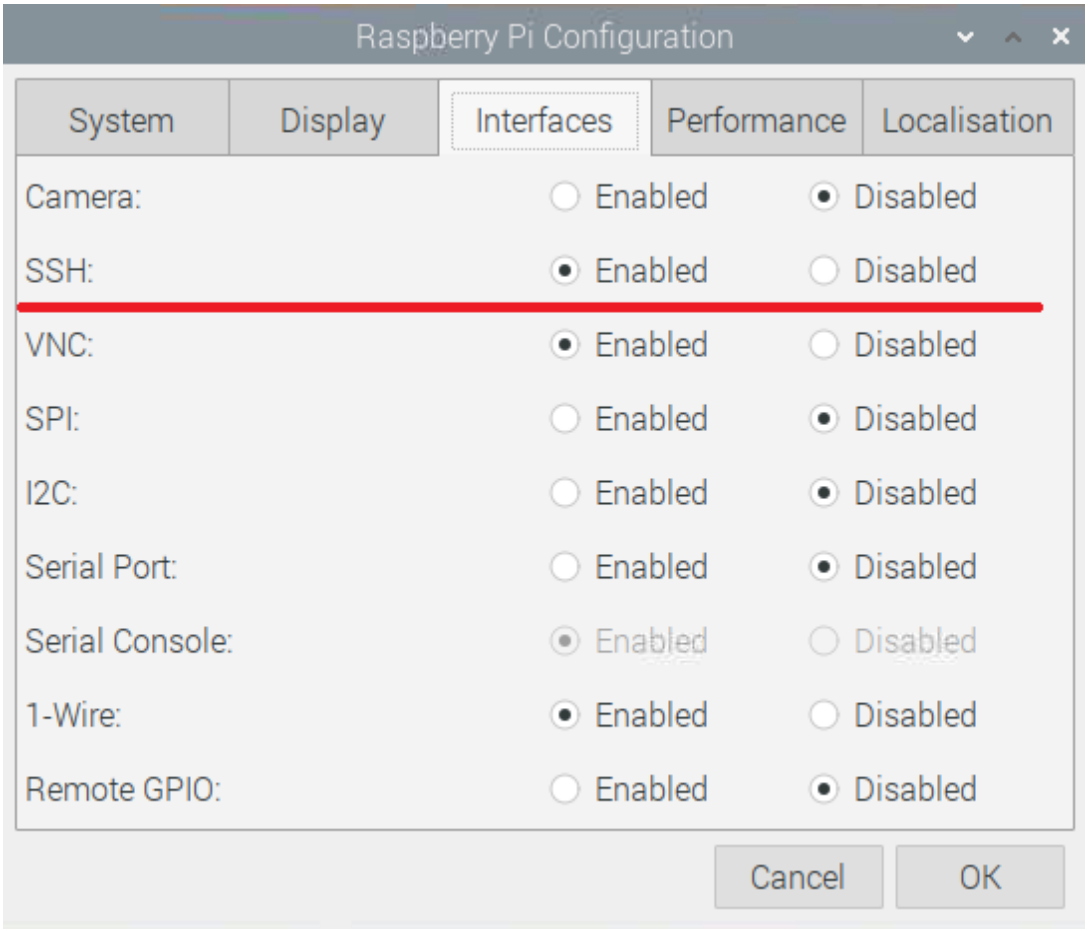


- **SSH:** De igual manera que el VNP esta aplicación permite controlar de forma remota la rapberry, asi como también viene precargado en el sistema operativo, pero se diferencia en que este solo puede acceder a la terminal de comandos, es decir, no tiene visualización

del escritorio. Como sucedió con VNP se debe activar el servicio a través del comando de la figura 29 y la confirmación de la figura 33.

Figura 31

Habilitar SSH



La aplicación de tipo cliente que se utiliza en Windows para hacer uso del SSH se llama Putty y es un software libre que se consigue en el siguiente enlace: https://www.putty.org/. Para hacer uso del programa es necesario ingresar la dirección ip de la raspberry, se puede observar en la figura 34. Al dar clic en el botón “Open” se genera la conexión exitosa como se puede verificar en la figura 35.

Figura 32

Configuración Putty

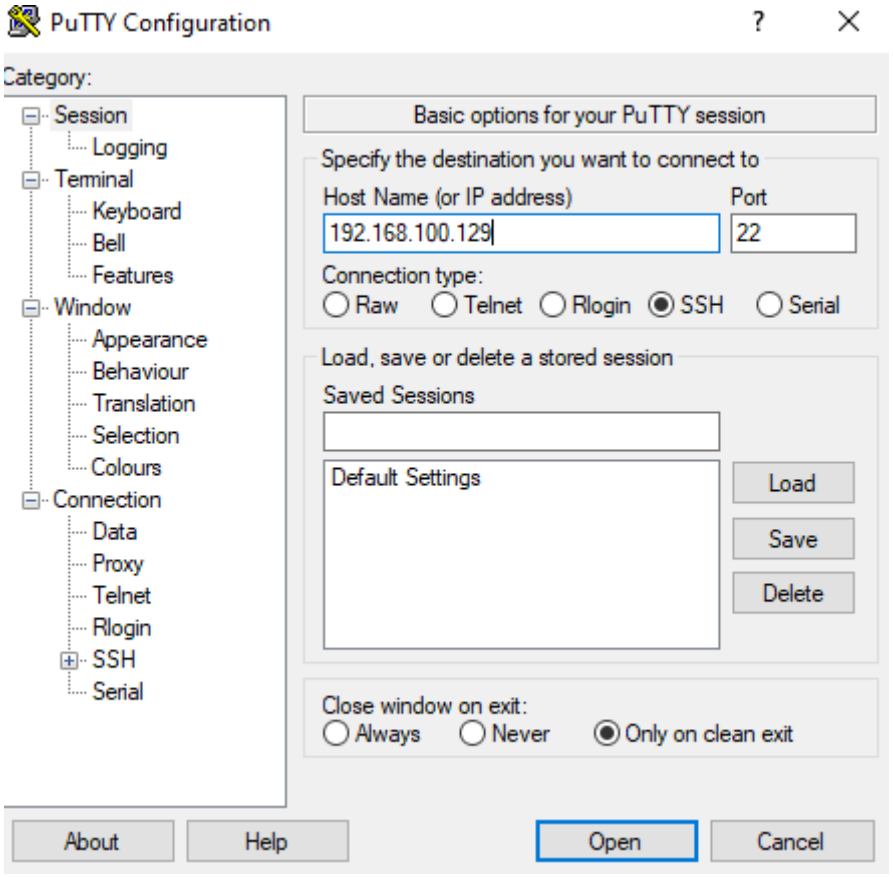
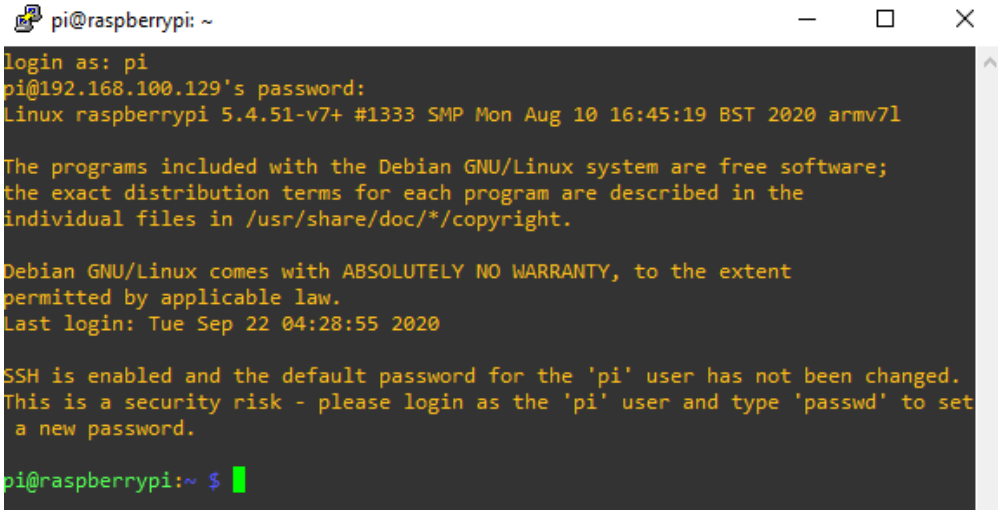


Figura 33

Visualización Raspberry a través de Putyy



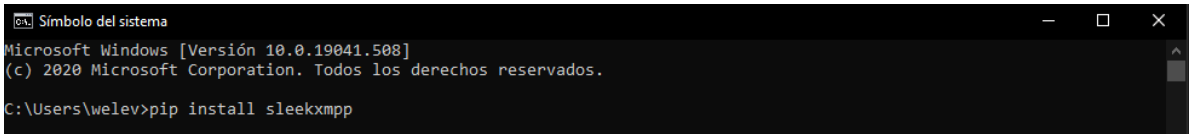
Configuración de XMPP

El servicio de comunicación XMPP debe instalarse tanto en los nodos Raspberry como en los clientes Windows y Android, esto se lo realiza a través de librerías creadas por desarrolladores de manera independiente y, para este proyecto, se seleccionó la librería “SleekXMPP”, la razón es que está programda en Phyton. A continuación, se detallará el proceso para cada uno de estos:

1. Para instalar la librería SleekXMPP en el cliente Windows se debe previamente instalar la versión de Phyton actualizada ingresando al símbolo del sistema e ingresando el comando de la figura 36.

Figura 34

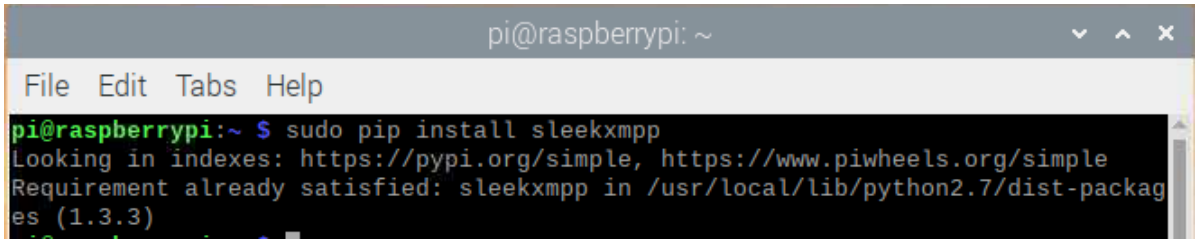
Comando para instalar SleekXMPP en Windows



2. La instalación de la librería en el Raspberry se lleva a cabo mediante la terminal con el comando de la figura 37.

Figura 35

Comando para instalar SleekXMPP en Raspberry



3. Una vez instaladas las librerías es momento de instalar el cliente XMPP, que en este caso es el cliente Pidgin. Para descargarlo basta visitar el siguiente enlace: https://www.pidgin.im/. Abrir el archivo y seguir los pasos de la instalación.

4. Para crear las cuentas del servidor XMPP público se visita la página https://lightwitch.org, aquí se crea una cuenta por cada nodo con un correo electrónico diferente cada uno. Para el presente proyecto serán solo tres cuentas, uno para cada nodo IoT y otro para el cliente que podrá acceder desde el PC o desde el teléfono inteligente.

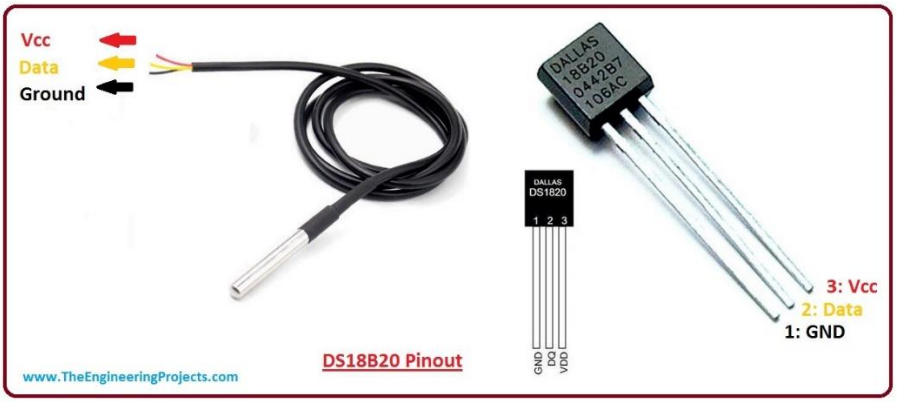
Sensor de temperatura

El sensor de temperatura DS18B20 fue seleccionado para este proyecto gracias a que existe programación en Python para su uso, este sensor es de tipo sonda y sus características y aspectos generales son los siguientes (Del Valle, s.f.).

- Interfaz de comunicación de un hilo
- Funcionamiento autónomo
- Requiere de 3.3V a 5.5V
- Puede medir temperaturas de -55° a 125°
- Tiene una precisión de +- 0.5° en temperaturas comprendidas de -10° a 85°
- Resolución programable de 9 a 12 bits

Figura 36

Sensor de temperatura DS18B20



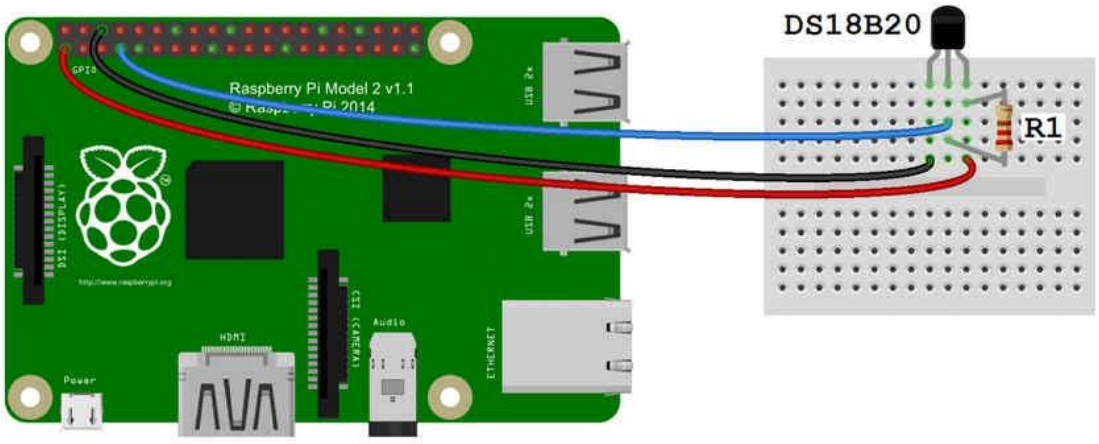
Nota. Tomado de *www.theengineeringprojects.com*

Conexión y configuración con la Raspberry Pi

Para conectar el sensor la Raspberry debe estar sin energía, es decir, apagada para evitar daños. El sensor cuenta con tres diferentes cables de diferentes colores, rojo para el voltaje, amarillo para la transferencia de datos y negro es tierra, la conexión se hace con una resistencia de acuerdo a la figura 39 que está, a continuación:

Figura 37

Conexión sensor con Raspberry Pi



Nota. Tomado de *https://i.imgur.com/mNNsUzt.jpg*

El siguiente paso consta de encender la máquina para realizar la instalación de la librería que controla el sensor, w1thermosensor, esto se realiza con el comando descrito en la figura 40.

Figura 38

Instalación librería del sensor térmico

```
pi@raspberrypi:~ $ sudo pip3 install w1thermsensor
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: w1thermsensor in /usr/local/lib/python3.7/dist-pa
ckages (1.3.0)
Requirement already satisfied: click in /usr/lib/python3/dist-packages (from w1t
hermsensor) (7.0)
```

Realizada la instalación se debe activar el servicio de comunicación de un hilo 1-wire, para ello se abre la configuración de la Raspberry Pi como se indica en las figuras 41 y 42.

Figura 39

Configuración Raspberry Pi

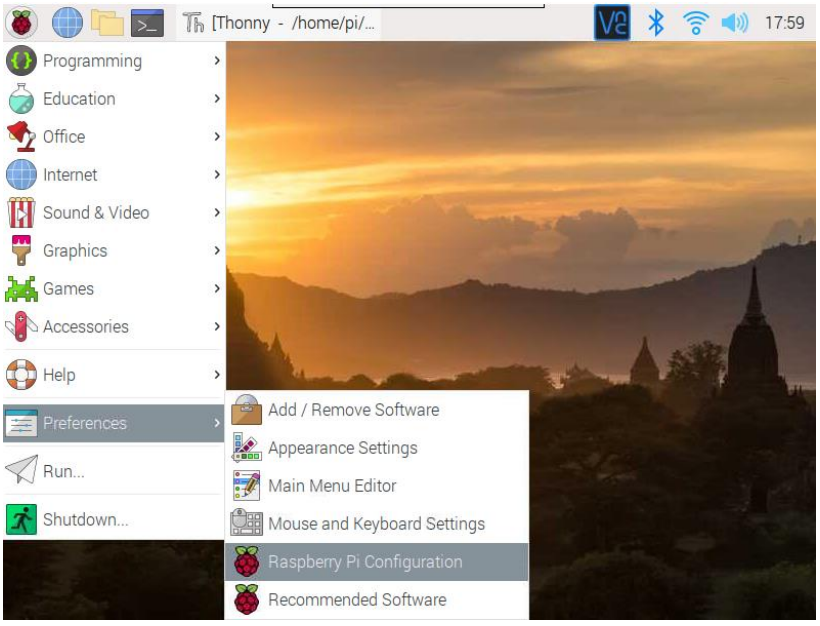
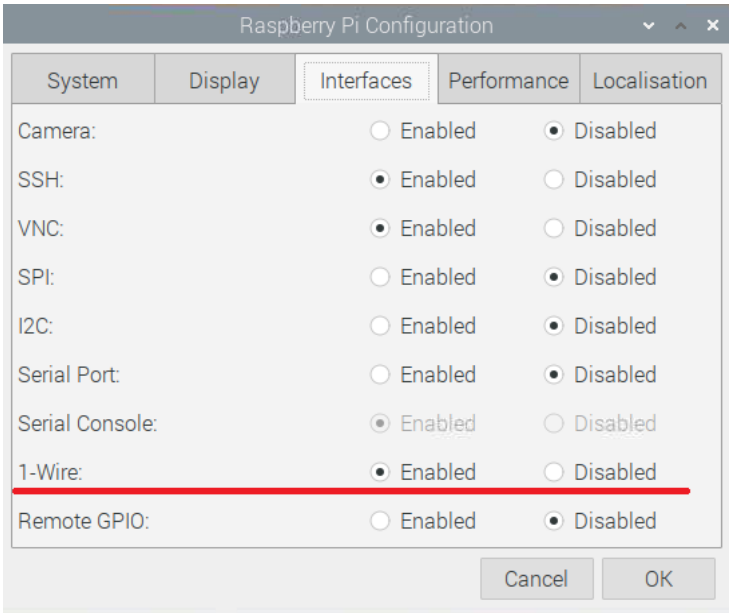


Figura 40

Configuración 1-Wire

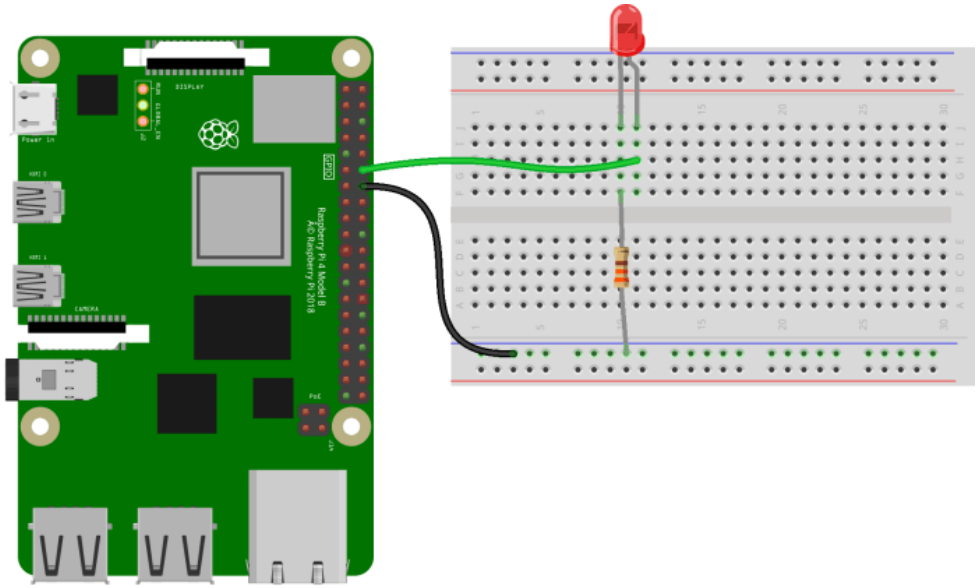


Diodo Led

Los LEDs son uno de los elementos fundamentales de los dispositivos elecrónicos, ya que son utilizados para informar estado de los procesos que se realizan, en el caso de las Raspberry estos se conectan utilizando las GPIO que son las entradas y salidas multipropósito, en la siguiente figura se muestra cómo debe ir la conexión, cabe recalcar que para todo el procedimiento de conexión el aparato debe estar apagado.

Figura 41

Conexión Raspberry Pi a LED



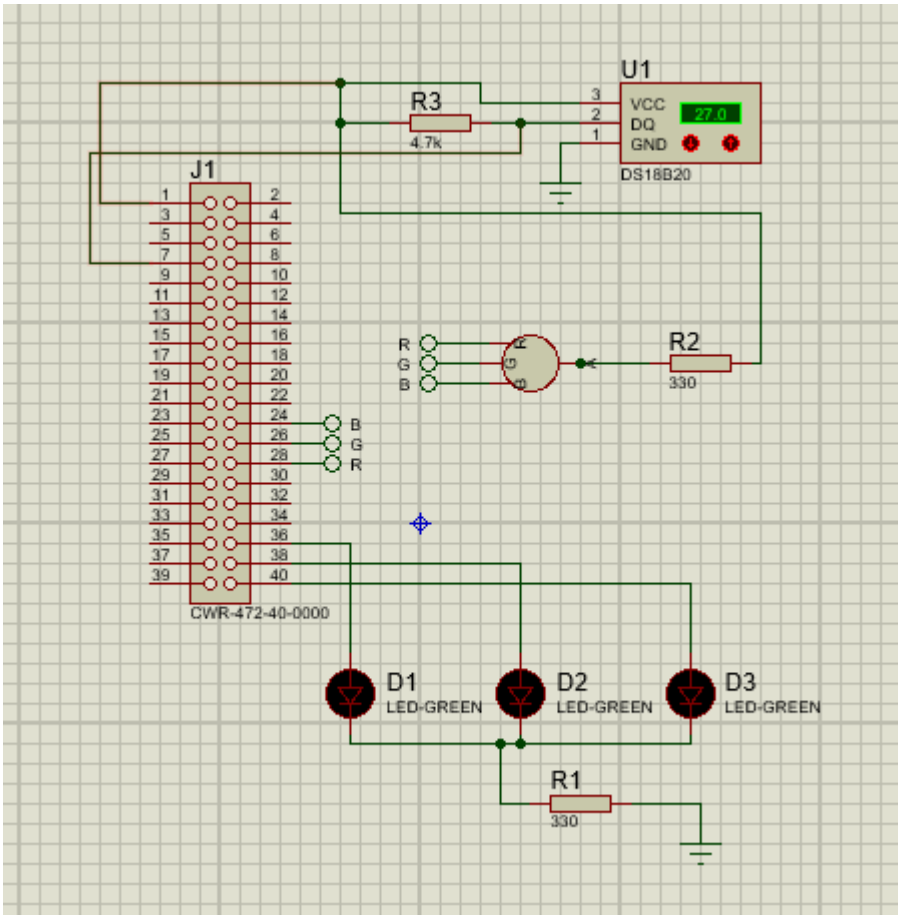
Nota. Tomado de *https://i2.wp.com/blog.330ohms.com/wpontent/uploads/2020/06/RPi_led_bb.png?fit=696%2C447&ssl=1*

Conexión del circuito completo

Finalmente se muestra a continuación cómo es el diseño final y las conexiones del circuito de los nodos Iot Raspberry Pi con sus sensores y actuadores.

Figura 42

Diseño y conexión del circuito nodo IoT



Código del algoritmo

Terminada la fase de ensamble y configuración se inicia la programación de los dos nodos, esto se refiere a los pasos que sigue el sistema para obtener información y realizar las peticiones del cliente. El lenguaje utilizado es Python.

Código GPIO Luminarias

A continuación, se muestra parte del código utilizado para controlar las distintas luminarias de los nodos IoT.

Figura 43

Inicialización para GPIO

```
#INICIALIZACION PARA GPIO
import RPi.GPIO as GPIO
import time
led0,led1,led2 =  21,20,16
r,g,b = 12,7,8
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(led0,GPIO.OUT)
GPIO.setup(led1,GPIO.OUT)
GPIO.setup(led2,GPIO.OUT)
GPIO.setup(r,GPIO.OUT)
GPIO.setup(g,GPIO.OUT)
GPIO.setup(b,GPIO.OUT)
```

La siguiente función desactiva y activa el primer led verde para indicar que se recibe un dato.

Figura 44

Función primer led verde al recibir dato

```
def mensaje_rx():
    GPIO.output(led0,GPIO.LOW)
    time.sleep(0.2)
    GPIO.output(led0,GPIO.HIGH)
```

La siguiente función desactiva y activa el segundo led verde para indicar que se transmite un dato.

Figura 45

Función segundo led verde al recibir dato

```
def mensaje_tx():
    GPIO.output(led1,GPIO.LOW)
    time.sleep(0.2)
    GPIO.output(led1,GPIO.HIGH)
```

La función de la figura 48 desactiva el tercer led verde para indicar que se inicia a calcular el promedio de temperaturas, la de la figura 49 activa el tercer led verde para indicar que esta listo el promedio de temperaturas.

Figura 46

Función led al calcular temperatura

```
28 def mensaje_tmp_ini():
29     GPIO.output(led2,GPIO.LOW)
30     time.sleep(0.2)
```

Figura 47

Función led al completar el cálculo de temperatura

```
33 def mensaje_tmp_lis():
34     GPIO.output(led2,GPIO.HIGH)
35     time.sleep(0.2)
```

La función de la figura 50 permite activar uno de los leds rgb de acuerdo al valor que toma la variable a, y la función de la figura 51 permite desactivar uno de los leds rgb de acuerdo al valor que toma la variable.

Figura 48

Función activar led rgb

```
56 def led_ison(a):
57     if a==1:
58         if GPIO.input(r):
59             return True
60         else:
61             return False
62     if a==2:
63         if GPIO.input(g):
64             return True
65         else:
66             return False
67     if a==3:
68         if GPIO.input(b):
69             return True
70         else:
71             return False
```

Figura 49

Función activar led rgb

```
def led_isoff(a):
    if a==1:
        if GPIO.input(r):
            return False
        else:
            return True
    if a==2:
        if GPIO.input(g):
            return False
        else:
            return True
    if a==3:
        if GPIO.input(b):
            return False
        else:
            return True
```

Código lectura de temperatura

A continuación, se indica el código para tomar la temperatura, se utiliza la instancia sensor de la clase W1ThermSensor para leer datos del sensor de temperatura, además se utiliza la función temperatura que calcula un promedio de cinco valores de temperatura obtenidos del método “get_temperature”.

Figura 50

Código lectura de temperatura

```
import time
from w1thermsensor import W1ThermSensor
sensor=W1ThermSensor()
def temperatura():
    promedio = 0
    for i in range(5):
        promedio += sensor.get_temperature()
        time.sleep(0.1)
    return(promedio/5)
```

Código XMPP Nodo IoT 1

Figura 51

Código XMPP Nodo IoT 1 parte 1

```
import logging
from sleekxmpp import ClientXMPP
from sleekxmpp.exceptions import IqError, IqTimeout
from gpio import *
from sensors import temperatura

class EchoBot(ClientXMPP):
    def __init__(self, jid, password):
        ClientXMPP.__init__(self, jid, password)
        self.add_event_handler("session_start", self.session_start)
        self.add_event_handler("message", self.message)
    def session_start(self, event):
        self.send_presence()
        self.get_roster()
    def message(self, msg):
        suma = 0
        cliente_win = 'fa_ber_win@lightwitch.org'
        nodo2 = 'fa_ber_no2@lightwitch.org'
```

Figura 52

Código XMPP Nodo IoT 1 parte 2

```
    if msg['type'] in ('chat', 'normal'):
        mensaje_rx()
        mrx = msg['body'].lower()
        msg.reply("").send()
        if "y ahora" in mrx:
            self.send_message(mto=cliente_win,
            mbody=mrx,
            mtype='chat')
            mensaje_tx()
        elif "baño" in mrx or "otro cuarto" in mrx or "mi cuarto" in mrx:
            self.send_message(mto=nodo2,
            mbody=mrx,
            mtype='chat')
            mensaje_tx()
        elif "hola" in mrx:
            msg['body'] = ("Hola,estoy activo !")
            msg.send()
            mensaje_tx()
        elif "adios" in mrx:
            msg['body'] = ("adios,continuo activo !")
            msg.send()
            mensaje_tx()
        elif ("cual" in mrx or "que" in mrx) and "temperatura" in mrx:
            msg['body'] = 'ahora le informo la temperatura'
            msg.send()
            mensaje_tx()
            mensaje_tmp_ini()
            msg['body'] = f"la temperatura es {temperatura()} °C"
            mensaje_tmp_lis()
            msg.send()
            mensaje_tx()
        elif "enciende" in mrx or "encender" in mrx:
            if "cocina" in mrx:
                if led_isoff(1):
```

Figura 53

Código XMPP Nodo IoT 1 parte 3

```
53                      led_on(1)
54                      msg['body'] = f"ahora encendi la luz de la cocina"
55                  else:
56                      msg['body'] = f"ahora esa luz ya esta encendida"
57                  msg.send()
58                  mensaje_tx()
59              elif "sala" in mrx:
60                  if led_isoff(2):
61                      led_on(2)
62                      msg['body'] = f"ahora encendi la luz de la sala"
63                  else:
64                      msg['body'] = f"ahora esa luz ya esta encendida"
65                  msg.send()
66                  mensaje_tx()
67              elif "comedor" in mrx:
68                  if led_isoff(3):
69                      led_on(3)
70                      msg['body'] = f"ahora encendi la luz del comedor"
71                  else:
72                      msg['body'] = f"ahora esa luz ya esta encendida"
73                  msg.send()
74                  mensaje_tx()
75          elif "apaga" in mrx or "apagar" in mrx:
76              if "cocina" in mrx:
77                  if led_ison(1):
78                      led_off(1)
79                      msg['body'] = f"ahora apague la luz de la cocina"
80                  else:
81                      msg['body'] = f"ahora esa luz ya esta apagada"
82                  msg.send()
83                  mensaje_tx()
84              elif "sala" in mrx:
85                  if led_ison(2):
86                      led_off(2)
87                      msg['body'] = f"ahora apague la luz de la sala"
```

Figura 54

Código XMPP Nodo IoT 1 parte 4

```
                else:
                    msg['body'] = f"ahora esa luz ya esta apaga"
                msg.send()
                mensaje_tx()
            elif "comedor" in mrx:
                if led_ison(3):
                    led_off(3)
                    msg['body'] = f"ahora apague la luz del comedor"
                else:
                    msg['body'] = f"ahora esa luz ya esta apagada"
                msg.send()
                mensaje_tx()
if __name__ == '__main__':
    logging.basicConfig(level=logging.DEBUG,
    format='%(levelname)-8s %(message)s')
    #Se ingresan los datos de la cuenta XMPP para el nodo1
    xmpp = EchoBot('fa_ber_no1@lightwitch.org', 'Nodoraspberryno_f1')
    #Inicia el proceso de conexión con el servidor
    xmpp.connect()
    xmpp.process(block = True)
```

Código XMPP Nodo IoT 2

Figura 55

Código XMPP Nodo IoT 2 parte 1

```
import logging
from sleekxmpp import ClientXMPP
from sleekxmpp.exceptions import IqError, IqTimeout
from gpio import *
from sensors import temperatura
class EchoBot(ClientXMPP):
    def __init__(self, jid, password):
        ClientXMPP.__init__(self, jid, password)
        self.add_event_handler("session_start", self.session_start)
        self.add_event_handler("message", self.message)
    def session_start(self, event):
        self.send_presence()
        self.get_roster()
```

Figura 56

Código XMPP Nodo IoT 2 parte 2

```
    def message(self, msg):
        suma = 0
        cliente_win = 'fa_ber_win@lightwitch.org'
        nodo1 = 'fa_ber_no1@lightwitch.org'
        if msg['type'] in ('chat', 'normal'):
            mensaje_rx()
            mrx = msg['body'].lower()
            msg.reply("").send()
            if "y ahora" in mrx:
                self.send_message(mto=cliente_win,
                mbody=mrx,
                mtype='chat')
                mensaje_tx()
            elif "baño" in mrx or "otro cuarto" in mrx or "mi cuarto" in mrx:
                self.send_message(mto=nodo2,
                mbody=mrx,
                mtype='chat')
                mensaje_tx()
            elif "hola" in mrx:
                msg['body'] = ("Hola,estoy activo !")
                msg.send()
                mensaje_tx()
            elif "adios" in mrx:
                msg['body'] = ("adios,continuo activo !")
                msg.send()
                mensaje_tx()
            elif ("cual" in mrx or "que" in mrx) and "temperatura" in mrx:
                msg['body'] = 'ahora le informo la temperatura'
                msg.send()
                mensaje_tx()
                mensaje_tmp_ini()
                msg['body'] = f"la temperatura es {temperatura()} °C"
                mensaje_tmp_lis()
                msg.send()
                mensaje_tx()
```

Figura 57

Código XMPP Nodo IoT 2 parte 3

```
49          elif "enciende" in mrx or "encender" in mrx:
50              if "cocina" in mrx:
51                  if led_isoff(1):
52                      led_on(1)
53                      msg['body'] = f"ahora encendi la luz de la cocina"
54                  else:
55                      msg['body'] = f"ahora esa luz ya esta encendida"
56                  msg.send()
57                  mensaje_tx()
58              elif "sala" in mrx:
59                  if led_isoff(2):
60                      led_on(2)
61                      msg['body'] = f"ahora encendi la luz de la sala"
62                  else:
63                      msg['body'] = f"ahora esa luz ya esta encendida"
64                  msg.send()
65                  mensaje_tx()
66              elif "comedor" in mrx:
67                  if led_isoff(3):
68                      led_on(3)
69                      msg['body'] = f"ahora encendi la luz del comedor"
70                  else:
71                      msg['body'] = f"ahora esa luz ya esta encendida"
72                  msg.send()
73                  mensaje_tx()
74          elif "apaga" in mrx or "apagar" in mrx:
75              if "cocina" in mrx:
76                  if led_ison(1):
77                      led_off(1)
78                      msg['body'] = f"ahora apague la luz de la cocina"
79                  else:
80                      msg['body'] = f"ahora esa luz ya esta apagada"
81                  msg.send()
82                  mensaje_tx()
83              elif "sala" in mrx:
```

Figura 58

Código XMPP Nodo IoT 2 parte 4

```
                    if led_ison(2):
                        led_off(2)
                        msg['body'] = f"ahora apague la luz de la sala"
                    else:
                        msg['body'] = f"ahora esa luz ya esta apaga"
                    msg.send()
                    mensaje_tx()
                elif "comedor" in mrx:
                    if led_ison(3):
                        led_off(3)
                        msg['body'] = f"ahora apague la luz del comedor"
                    else:
                        msg['body'] = f"ahora esa luz ya esta apagada"
                    msg.send()
                    mensaje_tx()
if __name__ == '__main__':
    logging.basicConfig(level=logging.DEBUG,
    format='%(levelname)-8s %(message)s')
    #Se ingresan los datos de la cuenta XMPP para el nodo2
    xmpp = EchoBot('fa_ber_no2@lightwitch.org', 'Nodoraspberryno_f1')
    #Inicia el proceso de conexión con el servidor
    xmpp.connect()
    xmpp.process(block = True)
```

Capítulo IV. Desarrollo y resultados

Preparación para las pruebas

Para iniciar las pruebas experimentales es necesario en primera instancia realizar la creación de las cuentas en el servidor público XMPP y su configuración en las aplicaciones cliente, luego de esto preparar físicamente los nodos y sus circuitos.

Cuentas

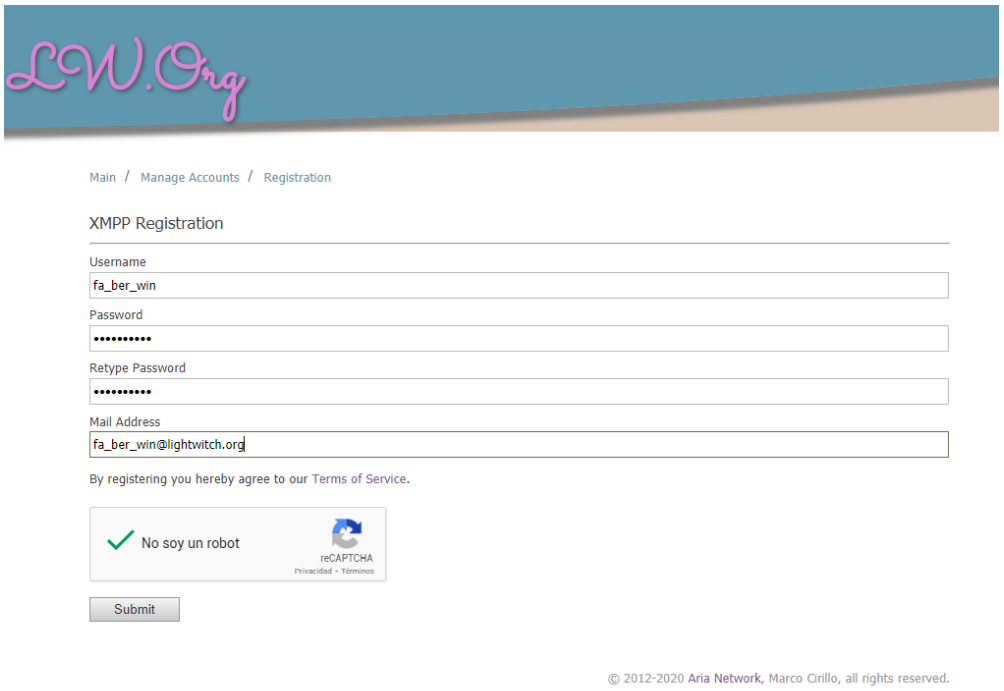
Para crear las cuentas se ingresa al sitio www.lightwitch.org (Figura 61), se selecciona "*manage account*" y sale el formulario que se muestra en la figura 62, se llenan los datos para crear las tres cuentas necesarias, una para la pc/Android y otras dos, una por cada nodo.

Figura 59

Página principal Ligtwitch

Figura 60

Formulario creación de cuenta.



Configuración cliente Pidgin

La configuración del cliente pidgin se hace luego de descargar e instalar el cliente en el dispositivo con Windows, al iniciar la aplicación sale un cuadro que indica que para su funcionamiento se deben ingresar la o las cuentas, para ello se da clic en el botón de “Añadir” (Figura 63). Luego aparecerá un formulario en el que se debe ingresar la cuenta, el dominio, usuario y contraseña para que se conecte al servidor (Figura 64); una vez allí se añaden los contactos con los que se comunicará, siendo estos los dos nodos Raspberry.

Figura 61

Cuadro de inicio Pidgin

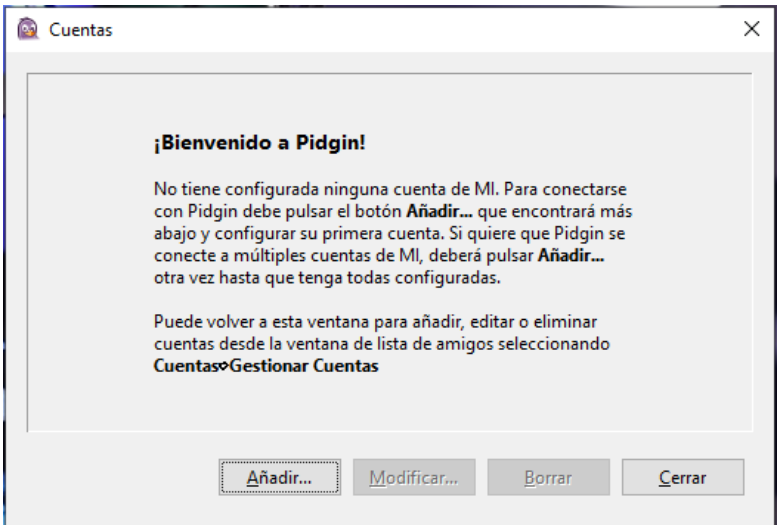
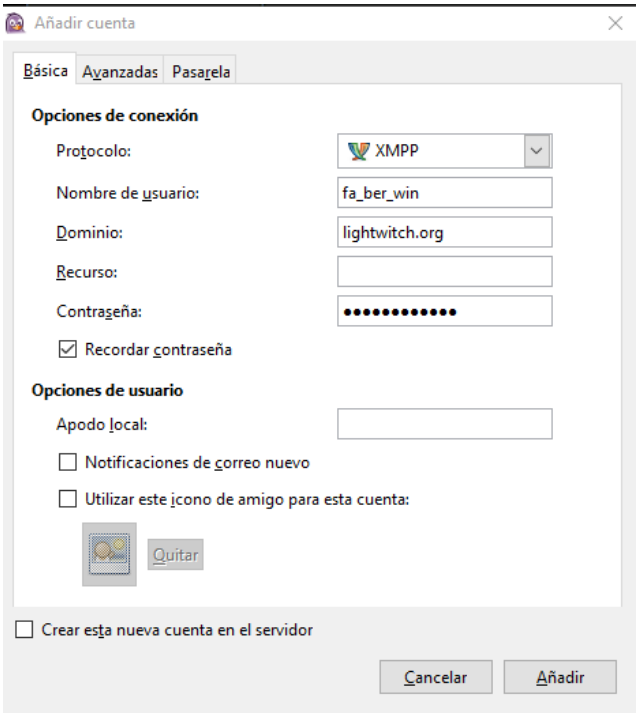


Figura 62

Añadir cuenta cliente Pidgin.

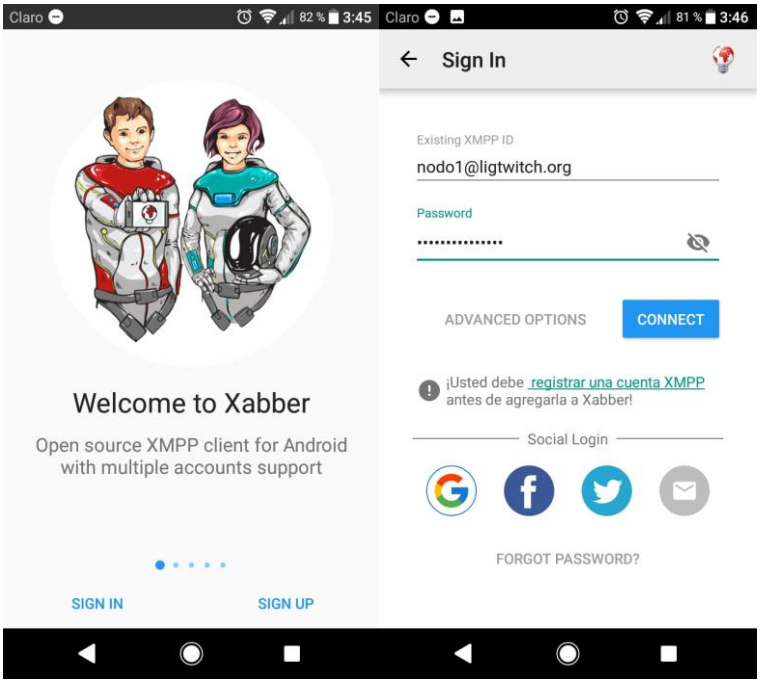


Configuración cliente Xabber

El cliente Xabber es el que controla la comunicación XMPP en el dispositivo Android, por ello, para obtenerlo se ingresó al Google Play Store y se lo descargó de manera gratuita, una vez instalado, al abrirlo, la pantalla (Figura 65) indica si se desea registrarse o iniciar con una cuenta existente, como se está utilizando ya un servidor XMPP se selecciona iniciar sesión, y se ingresan las credenciales (Figura 65). Una vez dentro se añaden los contactos siendo estos los dos nodos.

Figura 63

Añadir cuenta cliente Xabber.

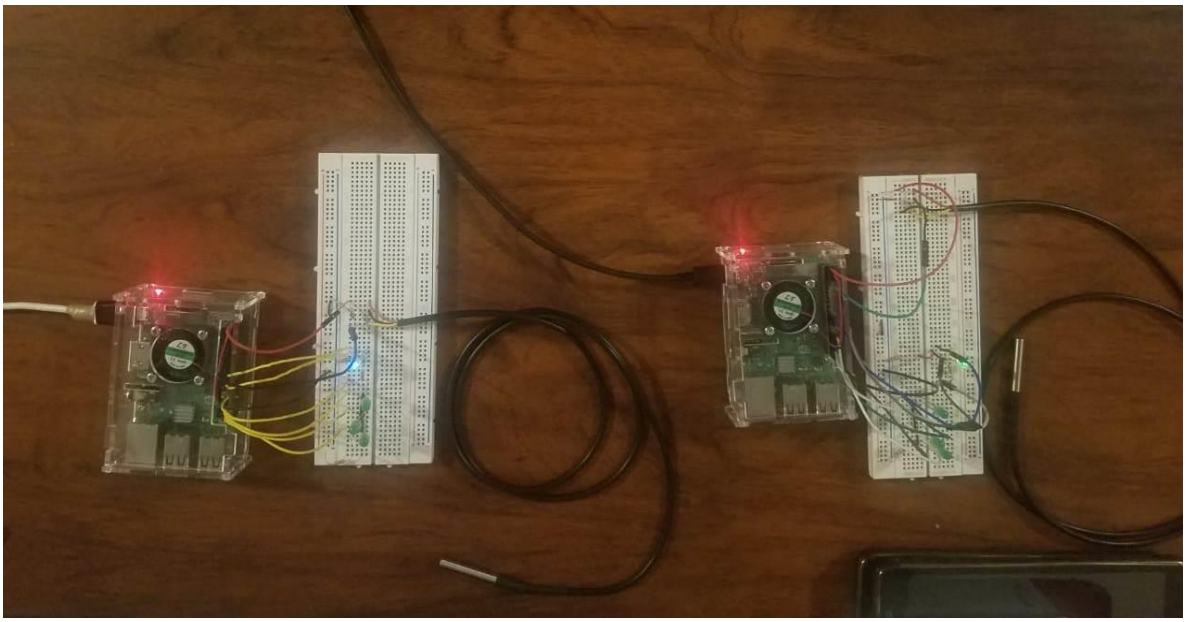


Inicializar nodos

En primera instancia se arma la circuitería del dispositivo tal como se indicó en la sección del diseño y se encienden ambas terminales como se puede observar en la siguiente figura:

Figura 64

Nodos



Para iniciar los nodos ya instaladas las librerías necesarias se debe, al momento de encenderlas, se presiona la tecla F5. El dispositivo iniciará los procesos de conexión con el servidor XMPP, si todo fue exitoso se puede verificar en el cliente de pc o Android si el contacto de ese nodo aparece como conectado.

Rutas

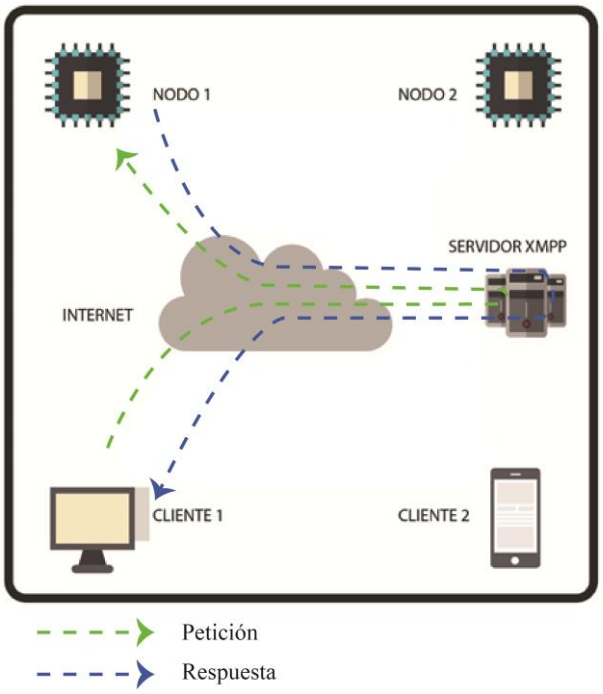
Para poder analizar los resultados de manera transparente resulta de gran importancia exponer primero las diferentes rutas que puede tener la comunicación entre los dispositivos, ya que esto afecta al tiempo de respuesta indudablemente. La comunicación XMPP utiliza una arquitectura cliente servidor, es por eso que existen tres tipos de rutas que pueden existir en este proyecto: comunicación directa, comunicación redireccionada y comunicación con dos o más clientes activos al mismo tiempo que comparten una misma cuenta.

Ruta de comunicación directa

En esta ruta la comunicación se dice directa porque solo interviene un nodo, un cliente y el servidor. Al momento de realizar el cliente una petición esta viaja a través de internet hacia el servidor XMPP, de ahí viaja al nodo que el cliente especificó, además este nodo tiene la capacidad de responder directamente ya que cuenta con la información requerida, la respuesta viaja al servidor XMPP que luego manda la respuesta al cliente, todo a través de internet en cada paso (Figura 67).

Figura 65

Ruta de comunicación directa



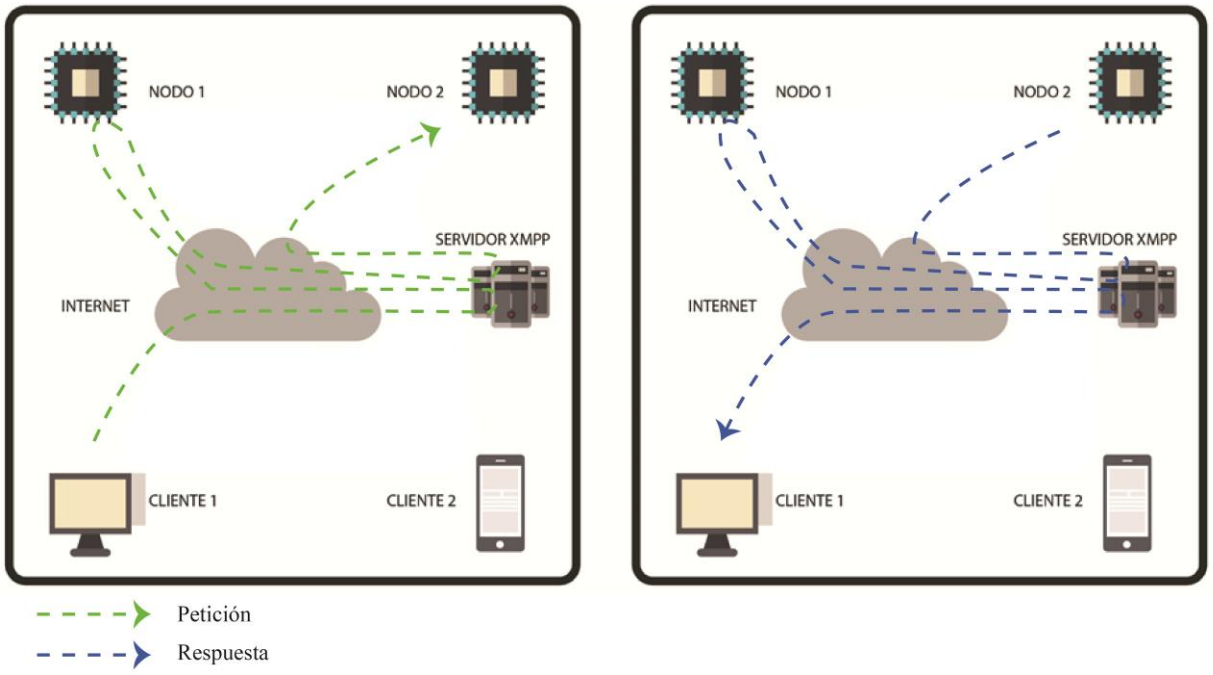
Ruta de comunicación redireccionada

Esta ruta de comunicación sucede cuando el cliente realiza la petición de una acción o información a un nodo que no tiene dicha información o control sobre la acción, es allí donde el nodo contacta al nodo que si disponga de lo solicitado a través del servidor XMPP. La respuesta tiene la característica de que

regresa de acuerdo a la ruta que tomó la petición, es decir, envía la respuesta al servidor, este al primer nodo contactado, luego el nodo al servidor y finalmente al cliente como se puede observar en la figura.

Figura 66

Ruta de comunicación redireccionada

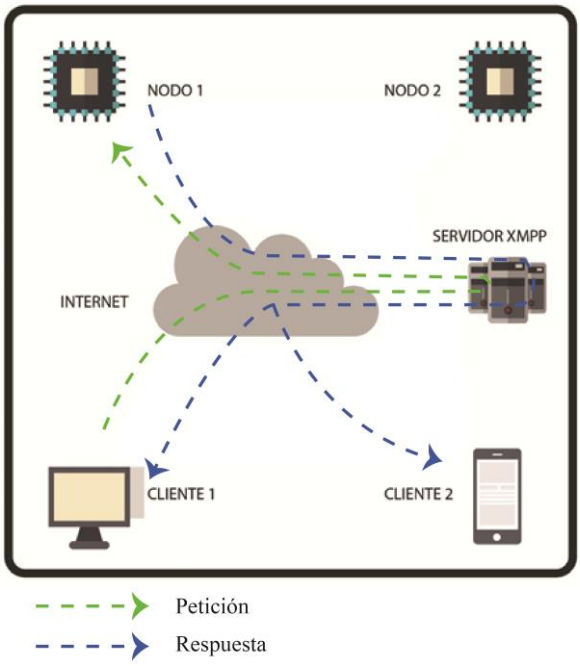


Ruta de comunicación con múltiples clientes activos

Esta comunicación puede darse tanto como en una ruta directa como redireccionada, la diferencia es que la respuesta es enviada a los dos clientes que comparten la misma cuenta, en el caso del ejemplo tanto el cliente PC como el cliente Android.

Figura 67

Ruta de comunicación con dos o más clientes activos



Pruebas experimentales

Para analizar la información de las pruebas se utilizó el software “Wireshark” ya que este obtiene los paquetes de la comunicación XMPP y entrega la información relevante para su estudio. Cabe recalcar que todos los mensajes enviados mediante dicho protocolo están cifrados y por ello no se pueden ver con transparencia en el Wireshark, por lo que se utilizará como guía el protocolo TLSv1.2 para identificar la conversación.

Tiempo de respuesta

Tras obtener los resultados en el Wireshark se toma el tiempo del inicio de la conversación y se lo resta del tiempo del fin de la misma, para conocer cuál es el último mensaje se notó que toda la conversación en la mayoría de casos toma un intercambio de diez paquetes en total. Los nodos se distribuyen así:

- Nodo 1: Cocina, sala y comedor
- Nodo 2: Cuarto A, Cuarto B y baño

Ruta directa cliente Pidgin (Windows)

Los tiempos obtenidos al enviar la instrucción directa de apagado a los nodos 1 y 2 desde las diferentes ubicaciones fueron los siguientes:

Figura 68

Paquetes al enviar instrucción directa de apagado de la cocina al Nodo 1

$$Tiempo\ respuesta = 24.660 - 23.972 = 0.688\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
53	23.972497	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TLSv1.2	249	Application Data
54	24.125354	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TCP	74	5222 → 56781 [ACK]
55	24.125458	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TLSv1.2	129	Application Data
56	24.231544	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TLSv1.2	136	Application Data
57	24.234838	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TCP	74	5222 → 56781 [ACK]
58	24.234925	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TCP	74	56781 → 5222 [ACK]
69	24.469619	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TLSv1.2	277	Application Data
71	24.510780	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TCP	74	56781 → 5222 [ACK]
80	24.613732	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TLSv1.2	331	Application Data
91	24.660143	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TCP	74	56781 → 5222 [ACK]

Figura 69

Paquetes al enviar instrucción directa de apagado de la sala al Nodo 1

$$Tiempo\ respuesta = 34.801 - 34.114 = 0.687\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
119	34.114138	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TLSv1.2	246	Application Data
120	34.220318	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TCP	74	5222 → 56781 [ACK]
121	34.220432	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TLSv1.2	129	Application Data
122	34.323346	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TLSv1.2	136	Application Data
123	34.330247	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TCP	74	5222 → 56781 [ACK]
124	34.330389	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TCP	74	56781 → 5222 [ACK]
125	34.609670	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TLSv1.2	277	Application Data
126	34.650590	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TCP	74	56781 → 5222 [ACK]
127	34.753336	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TLSv1.2	329	Application Data
128	34.800519	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TCP	74	56781 → 5222 [ACK]

Figura 70

Paquetes al enviar instrucción directa de apagado del comedor al Nodo 1

$$Tiempo\ respuesta = 39.512 - 38.828 = 0.684\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
143	38.828073	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TLSv1.2	249	Application Data
144	38.937791	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TCP	74	5222 → 56781 [ACK]
145	38.937846	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TLSv1.2	129	Application Data
146	39.042667	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TLSv1.2	136	Application Data
147	39.048114	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TCP	74	5222 → 56781 [ACK]
148	39.048210	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TCP	74	56781 → 5222 [ACK]
149	39.319870	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TLSv1.2	277	Application Data
151	39.362634	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TCP	74	56781 → 5222 [ACK]
152	39.465091	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TLSv1.2	330	Application Data
153	39.512799	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TCP	74	56781 → 5222 [ACK]

De las figuras 70, 71 y 72 se puede evidenciar la comunicación del cliente con el nodo 1 consiguiendo un tiempo de respuesta promedio de 0.686s, es decir, la comunicación tarda poco más de medio segundo en completarse, qué para la percepción humana es prácticamente instantáneo. Cabe recalcar que la diferencia en tiempo de respuesta de acuerdo al sitio en que se ubicaba el nodo es despreciable, por ello la ubicación no representa un gran factor a tomar en cuenta siempre y cuando exista una buena conexión a internet.

Ahora, la comunicación del cliente con el nodo 2 resultó así:

Figura 71

Paquetes al enviar instrucción directa de apagado del baño al Nodo 2

$$Tiempo\ respuesta = 3.324 - 2.718 = 0.606\ s$$

50	2.718195	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TLSv1.2	248	Application Data
51	2.830298	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TCP	74	5222 → 56781 [ACK]
52	2.830426	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TLSv1.2	129	Application Data
53	2.933807	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TLSv1.2	136	Application Data
54	2.940181	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TCP	74	5222 → 56781 [ACK]
55	2.940231	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TCP	74	56781 → 5222 [ACK]
56	3.131087	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TLSv1.2	277	Application Data
57	3.173886	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TCP	74	56781 → 5222 [ACK]
60	3.276178	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TLSv1.2	328	Application Data
61	3.324187	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TCP	74	56781 → 5222 [ACK]

Figura 72

Paquetes al enviar instrucción directa de apagado del cuarto A al Nodo 2

$$Tiempo\ respuesta = 10.303 - 9.698 = 0.605\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
103	9.698771	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TLSv1.2	251	Application Data
104	9.807184	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TCP	74	5222 → 56781 [ACK]
105	9.807355	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TLSv1.2	129	Application Data
106	9.910158	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TLSv1.2	136	Application Data
107	9.917031	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TCP	74	5222 → 56781 [ACK]
108	9.917129	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TCP	74	56781 → 5222 [ACK]
109	10.111445	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TLSv1.2	277	Application Data
110	10.153648	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TCP	74	56781 → 5222 [ACK]
111	10.256079	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TLSv1.2	331	Application Data
112	10.303857	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TCP	74	56781 → 5222 [ACK]

Figura 73

Paquetes al enviar instrucción directa de apagado del cuarto B al Nodo 2

$$Tiempo\ respuesta = 15.883 - 15.128 = 0.755\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
134	15.128385	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TLSv1.2	253	Application Data
136	15.284559	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TCP	74	5222 → 56781 [ACK]
137	15.284663	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TLSv1.2	129	Application Data
138	15.387901	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TLSv1.2	136	Application Data
139	15.394631	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TCP	74	5222 → 56781 [ACK]
140	15.394719	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TCP	74	56781 → 5222 [ACK]
142	15.614481	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TLSv1.2	277	Application Data
143	15.660405	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TCP	74	56781 → 5222 [ACK]
144	15.768750	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TLSv1.2	360	Application Data
145	15.776384	2800:bf0:23:1b9f:5a2:fff5:b…	2600:3c03::f03c:92ff:fe24:b7ee	TLSv1.2	137	Application Data
147	15.883986	2600:3c03::f03c:92ff:fe24:b…	2800:bf0:23:1b9f:5a2:fff5:b539:eb25	TCP	74	5222 → 56781 [ACK]

De las figuras 73, 74 y 75 se puede evidenciar la comunicación del cliente con el nodo 2 consiguiendo un tiempo de respuesta promedio de 0.655s, es decir, la comunicación tarda poco más de medio segundo en completarse similar a lo ocurrido con el nodo 1. Cabe recalcar que la diferencia en tiempo de respuesta de acuerdo al sitio varía en el cuarto B con una diferencia de 0.150s aproximadamente, este valor representa una cantidad igual despreciable para la percepción humana y puede explicarse por el tráfico de red aumentado que pudo suceder en ese instante, pudo ser que otro dispositivo ajeno a la red IoT pero que se encuentra dentro de la misma red de internet del lugar estuviere usando bastante ancho de banda.

Los tiempos obtenidos al enviar la instrucción directa de encendido a los nodos 1 y 2 desde las diferentes ubicaciones fueron los siguientes:

Figura 74

Paquetes al enviar instrucción directa de encendido de la cocina al Nodo 1

$$Tiempo\ respuesta = 6.115 - 5.473 = 0.642\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
23	5.473093	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TLSv…	213	Application Data
25	5.626225	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TCP	74	5222 → 56781 [ACK]
26	5.626334	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TLSv…	129	Application Data
27	5.774001	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv…	136	Application Data
28	5.774001	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TCP	74	5222 → 56781 [ACK]
29	5.774163	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]
31	5.916195	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv…	277	Application Data
32	5.964858	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]
33	6.068643	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv…	332	Application Data
34	6.115038	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]

Figura 75

Paquetes al enviar instrucción directa de encendido de la sala al Nodo 1

$$Tiempo\ respuesta = 34.053 - 33.378 = 0.675\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
249	33.378938	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TLSv…	248	Application Data
250	33.494372	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TCP	74	5222 → 56781 [ACK]
251	33.494500	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TLSv…	129	Application Data
252	33.598066	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv…	136	Application Data
253	33.604706	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TCP	74	5222 → 56781 [ACK]
254	33.604803	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]
256	33.795299	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv…	277	Application Data
257	33.837106	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]
259	33.939418	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv…	356	Application Data
260	33.946714	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TLSv…	136	Application Data
261	34.053830	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TCP	74	5222 → 56781 [ACK]

Figura 76

Paquetes al enviar instrucción directa de encendido del comedor al Nodo 1

$$Tiempo\ respuesta = 44.438 - 43.822 = 0.616\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
288	43.822143	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TLSv…	251	Application Data
291	43.929953	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TCP	74	5222 → 56781 [ACK]
292	43.930073	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TLSv…	129	Application Data
293	44.032425	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv…	136	Application Data
294	44.039057	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TCP	74	5222 → 56781 [ACK]
295	44.039143	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]
296	44.239743	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv…	277	Application Data
297	44.288640	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]
300	44.393213	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv…	331	Application Data
301	44.438441	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]

De las figuras 76, 77 y 78 se puede obtener un tiempo de respuesta promedio de 0.644s, quiere decir que la comunicación tarda poco más de medio segundo en completarse, de igual manera que cuando la instrucción era la de apagarse, esto quiere decir que en cuanto a las dos instrucciones en el nodo 1 no hay diferencia en tiempo de respuesta entre ellas.

Ahora, la comunicación del cliente con el nodo 2 resultó así:

Figura 77

Paquetes al enviar instrucción directa de encendido del baño al Nodo 2

$$Tiempo\ respuesta = 6.827 - 6.158 = 0.669\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
25	6.158322	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TLSv1.2	213	Application Data
26	6.311539	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TCP	74	5222 → 56781 [ACK]
27	6.311653	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TLSv1.2	129	Application Data
28	6.414413	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv1.2	136	Application Data
29	6.421195	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TCP	74	5222 → 56781 [ACK]
30	6.421264	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]
31	6.572874	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv1.2	303	Application Data
32	6.573695	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TLSv1.2	137	Application Data
33	6.783011	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv1.2	329	Application Data
34	6.827109	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]

Figura 78

Paquetes al enviar instrucción directa de encendido del cuarto A al Nodo 2

$$Tiempo\ respuesta = 15.278 - 14.653 = 0.625\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
100	14.653413	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TLSv1.2	253	Application Data
113	14.757565	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TCP	74	5222 → 56781 [ACK]
114	14.757801	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TLSv1.2	129	Application Data
119	14.867639	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv1.2	136	Application Data
120	14.870792	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TCP	74	5222 → 56781 [ACK]
121	14.870849	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]
129	15.081472	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv1.2	277	Application Data
131	15.128399	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]
134	15.231854	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv1.2	332	Application Data
135	15.278516	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]

Figura 79

Paquetes al enviar instrucción directa de encendido del cuarto B al Nodo 2

$$Tiempo\ respuesta = 19.972 - 19.357 = 0.615\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
202	19.357131	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TLSv1.2	255	Application Data
204	19.505560	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TCP	74	5222 → 56781 [ACK]
205	19.505664	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TLSv1.2	129	Application Data
206	19.608953	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv1.2	136	Application Data
207	19.615673	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TCP	74	5222 → 56781 [ACK]
208	19.615760	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]
209	19.773300	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv1.2	277	Application Data
210	19.822447	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]
212	19.929081	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv1.2	334	Application Data
213	19.972179	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]

El promedio de tiempo de respuesta obtenido de las figuras 79 al 81 es 0.636s, sigue siendo inferior a un segundo y no tiene una diferencia notable en los tiempos de acuerdo al sitio. El comportamiento es prácticamente igual que con el nodo 1.

Ahora, en cuanto a comunicación de ruta directa, resta analizar las peticiones a ambos nodos sobre la temperatura captada por sus sensores. Los datos obtenidos son los siguientes:

Figura 80

Paquetes al enviar instrucción directa de toma de temperatura al Nodo 1

$$Tiempo\ respuesta = 7.717 - 1.884 = 5.833\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
12	1.884944	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TLSv1.2	268	Application Data
16	2.046339	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TCP	74	5222 → 56781 [ACK]
17	2.046743	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TLSv1.2	129	Application Data
18	2.149815	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv1.2	136	Application Data
19	2.194563	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]
20	2.297881	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv1.2	277	Application Data
21	2.344806	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]
24	2.454599	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv1.2	330	Application Data
25	2.504446	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]
44	7.676850	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv1.2	334	Application Data
45	7.717631	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]

Figura 81

Paquetes al enviar instrucción directa de toma de temperatura al Nodo 2

$$Tiempo\ respuesta = 7.505 - 1.657 = 5.848\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
3	1.657687	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TLSv1.2	271	Application Data
4	1.808969	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TCP	74	5222 → 56781 [ACK]
5	1.809096	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TLSv1.2	129	Application Data
6	1.940258	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv1.2	136	Application Data
7	1.940258	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TCP	74	5222 → 56781 [ACK]
8	1.940412	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]
10	2.071273	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv1.2	277	Application Data
11	2.114761	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]
12	2.217290	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv1.2	356	Application Data
13	2.224525	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TLSv1.2	137	Application Data
14	2.328003	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TCP	74	5222 → 56781 [ACK]
48	7.454731	2600:3c03::f03c:92f…	2800:bf0:23:1b9f:5a…	TLSv1.2	333	Application Data
49	7.505541	2800:bf0:23:1b9f:5a…	2600:3c03::f03c:92f…	TCP	74	56781 → 5222 [ACK]

Cómo se puede ver en las figuras 82 y 83 el tiempo de respuesta de ambos nodos es prácticamente el mismo, 5.8s. Se nota que ahora si la respuesta toma mucho más tiempo en llegar al cliente desde que se envió la solicitud, esto se explica al comprender el proceso, cuando el nodo obtiene la petición debe realizar un proceso de toma de temperatura, este consiste en tomar tres mediciones, una después de otra y obtiene un promedio entre ellas, esto lo realiza para que la respuesta sea más fiable, este último dato es el que se

envía de regreso al cliente. Es normal que estas tomas de temperatura tomen poco más de un segundo entre ellas, por lo que un tiempo de 5.8s es normal y óptimo.

Ruta redireccionada cliente Pidgin (Windows)

Primero se probó la comunicación redireccionada al solicitar al nodo 2 el apagado de los actuadores del nodo 1, los datos obtenidos fueron los siguientes:

Figura 82

Paquetes al enviar instrucción redireccionada de apagado de la cocina al Nodo 2

$$Tiempo\ respuesta = 3.226 - 1.916 = 1.310\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
8	1.916086	192.168.100.22	172.104.16.89	TLSv1.2	228	Application Data
10	2.056932	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
11	2.056992	192.168.100.22	172.104.16.89	TLSv1.2	109	Application Data
68	2.150885	172.104.16.89	192.168.100.22	TLSv1.2	116	Application Data
70	2.157284	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
72	2.157370	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
122	2.313069	172.104.16.89	192.168.100.22	TLSv1.2	257	Application Data
124	2.356833	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
327	3.182028	172.104.16.89	192.168.100.22	TLSv1.2	271	Application Data
357	3.226922	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]

Figura 83

Paquetes al enviar instrucción redireccionada de apagado de la sala al Nodo 2

$$Tiempo\ respuesta = 12.310 - 10.981 = 1.329\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
426	10.981878	192.168.100.22	172.104.16.89	TLSv1.2	231	Application Data
427	11.083843	172.104.16.89	192.168.100.22	TCP	54	5222 → 57508 [ACK]
428	11.083899	192.168.100.22	172.104.16.89	TLSv1.2	109	Application Data
442	11.180020	172.104.16.89	192.168.100.22	TLSv1.2	116	Application Data
444	11.183158	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
445	11.183241	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
484	11.377610	172.104.16.89	192.168.100.22	TLSv1.2	257	Application Data
501	11.420986	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
753	12.266934	172.104.16.89	192.168.100.22	TLSv1.2	274	Application Data
755	12.310734	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]

Figura 84

Paquetes al enviar instrucción redireccionada de apagado del comedor al Nodo 2

$$Tiempo\ respuesta = 20.747 - 19.364 = 1.383\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
781	19.364356	192.168.100.22	172.104.16.89	TLSv1.2	234	Application Data
782	19.459614	172.104.16.89	192.168.100.22	TCP	54	5222 → 57508 [ACK]
783	19.459667	192.168.100.22	172.104.16.89	TLSv1.2	109	Application Data
784	19.554878	172.104.16.89	192.168.100.22	TLSv1.2	116	Application Data
785	19.559479	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
786	19.559527	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
846	19.789793	172.104.16.89	192.168.100.22	TLSv1.2	257	Application Data
865	19.837156	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
1342	20.695149	172.104.16.89	192.168.100.22	TLSv1.2	277	Application Data
1346	20.747417	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]

De las figuras 84, 85 y 86 se consigue un tiempo de respuesta promedio de 1.340s, es decir, la comunicación tarda poco más de un segundo en completarse, esto corresponde a prácticamente el doble de tiempo que en una comunicación de ruta directa. Este resultado es el esperado ya que se aumenta el tiempo de comunicación entre el nodo 2 y el nodo 1 para obtener la respuesta y llevar a cabo la acción solicitada por el cliente. A continuación, se muestran los resultados de realizar el mismo experimento, pero solicitando al primer nodo la acción de apagado para los actuadores del segundo nodo.

Figura 85

Paquetes al enviar instrucción redireccionada de apagado del baño al Nodo 1

$$Tiempo\ respuesta = 4.917 - 3.219 = 1.698\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
109	3.219795	192.168.100.22	172.104.16.89	TLSv1.2	229	Application Data
110	3.359254	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
111	3.359365	192.168.100.22	172.104.16.89	TLSv1.2	109	Application Data
112	3.452901	172.104.16.89	192.168.100.22	TLSv1.2	116	Application Data
113	3.458917	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
114	3.458984	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
136	3.676000	172.104.16.89	192.168.100.22	TLSv1.2	257	Application Data
142	3.716120	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
318	4.872372	172.104.16.89	192.168.100.22	TLSv1.2	274	Application Data
319	4.917242	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]

Figura 86

Paquetes al enviar instrucción redireccionada de apagado del cuarto A al Nodo 1

$$Tiempo\ respuesta = 15.043 - 13.636 = 1.407\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
498	13.636521	192.168.100.22	172.104.16.89	TLSv1.2	226	Application Data
500	13.736338	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
501	13.736393	192.168.100.22	172.104.16.89	TLSv1.2	109	Application Data
522	13.831636	172.104.16.89	192.168.100.22	TLSv1.2	116	Application Data
523	13.834835	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
524	13.834929	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
528	14.030365	172.104.16.89	192.168.100.22	TLSv1.2	257	Application Data
549	14.078059	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
728	14.916260	172.104.16.89	192.168.100.22	TLSv1.2	298	Application Data
729	14.943021	192.168.100.22	172.104.16.89	TLSv1.2	116	Application Data
784	15.043922	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]

Figura 87

Paquetes al enviar instrucción redireccionada de apagado del cuarto B al Nodo 1

$$Tiempo\ respuesta = 22.320 - 20.906 = 1.414\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
829	20.906340	192.168.100.22	172.104.16.89	TLSv1.2	230	Application Data
830	21.001578	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
831	21.001633	192.168.100.22	172.104.16.89	TLSv1.2	109	Application Data
832	21.096134	172.104.16.89	192.168.100.22	TLSv1.2	116	Application Data
833	21.101635	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
834	21.101729	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
859	21.376186	172.104.16.89	192.168.100.22	TLSv1.2	257	Application Data
878	21.422563	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
1049	22.273372	172.104.16.89	192.168.100.22	TLSv1.2	273	Application Data
1068	22.320068	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]

De las figuras 87, 88 y 89 se consigue un tiempo de respuesta promedio de 1.506s, es decir, la comunicación tarda un segundo y medio en completarse, esto corresponde a poco más del doble de tiempo que en una comunicación de ruta directa. Este resultado es el esperado ya que se aumenta el tiempo de comunicación entre el nodo 2 y el nodo 1 para obtener la respuesta y llevar a cabo la acción solicitada por el cliente, además, si se toma en cuenta la diferencia de 0.2s entre la anterior prueba y esta, se puede deber a la congestión del tráfico de la red en el instante de realizar la petición. A continuación, se muestran los resultados de realizar el mismo experimento, pero en vez de solicitar el apagado de los actuadores se pide el encendido de estos.

Figura 88

Paquetes al enviar instrucción redireccionada de encendido de la cocina al Nodo 2

$$Tiempo\ respuesta = 3.318 - 2.007 = 1.311\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
5	2.007202	192.168.100.22	172.104.16.89	TLSv1.2	193	Application Data
6	2.148171	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
7	2.148296	192.168.100.22	172.104.16.89	TLSv1.2	109	Application Data
60	2.242030	172.104.16.89	192.168.100.22	TLSv1.2	116	Application Data
62	2.247926	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
63	2.248012	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
97	2.402418	172.104.16.89	192.168.100.22	TLSv1.2	257	Application Data
116	2.447889	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
313	3.269670	172.104.16.89	192.168.100.22	TLSv1.2	273	Application Data
334	3.318054	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]

Figura 89

Paquetes al enviar instrucción redireccionada de encendido de la sala al Nodo 2

$$Tiempo\ respuesta = 17.420 - 16.097 = 1.323\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
391	16.097484	192.168.100.22	172.104.16.89	TLSv1.2	234	Application Data
392	16.202234	172.104.16.89	192.168.100.22	TCP	54	5222 → 57508 [ACK]
393	16.202346	192.168.100.22	172.104.16.89	TLSv1.2	109	Application Data
412	16.296530	172.104.16.89	192.168.100.22	TLSv1.2	116	Application Data
431	16.304286	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
432	16.304376	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
490	16.489590	172.104.16.89	192.168.100.22	TLSv1.2	257	Application Data
509	16.530793	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
736	17.373436	172.104.16.89	192.168.100.22	TLSv1.2	273	Application Data
740	17.420822	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]

Figura 90

Paquetes al enviar instrucción redireccionada de encendido del comedor al Nodo 2

$$Tiempo\ respuesta = 31.865 - 30.317 = 1.548\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
754	30.317593	192.168.100.22	172.104.16.89	TLSv1.2	235	Application Data
755	30.442437	172.104.16.89	192.168.100.22	TCP	54	5222 → 57508 [ACK]
756	30.442550	192.168.100.22	172.104.16.89	TLSv1.2	109	Application Data
757	30.559950	172.104.16.89	192.168.100.22	TLSv1.2	116	Application Data
758	30.560474	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
759	30.560573	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
815	30.795907	172.104.16.89	192.168.100.22	TLSv1.2	257	Application Data
834	30.838808	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
1349	31.709027	172.104.16.89	192.168.100.22	TLSv1.2	299	Application Data
1352	31.720991	192.168.100.22	172.104.16.89	TLSv1.2	116	Application Data
1355	31.865310	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]

De las figuras 90, 91 y 92 se consigue un tiempo de respuesta promedio de 1.394s, es decir, la comunicación tarda poco más de un segundo en completarse, esto corresponde a prácticamente el doble de tiempo que en una comunicación de ruta directa. Este resultado es el esperado ya que se aumenta el tiempo de comunicación entre el nodo 2 y el nodo 1 para obtener la respuesta y llevar a cabo la acción solicitada por el cliente. A continuación, se muestran los resultados de realizar el mismo experimento, pero solicitando al primer nodo la acción de apagado para los actuadores del segundo nodo.

Figura 91

Paquetes al enviar instrucción redireccionada de encendido del baño al Nodo 1

$$Tiempo\ respuesta = 13.509 - 12.146 = 1.363\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
58	12.146963	192.168.100.22	172.104.16.89	TLSv1.2	195	Application Data
59	12.245207	172.104.16.89	192.168.100.22	TCP	54	5222 → 57508 [ACK]
60	12.245294	192.168.100.22	172.104.16.89	TLSv1.2	109	Application Data
61	12.338899	172.104.16.89	192.168.100.22	TLSv1.2	116	Application Data
62	12.345118	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
63	12.345149	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
272	12.564583	172.104.16.89	192.168.100.22	TLSv1.2	257	Application Data
293	12.608809	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
468	13.466875	172.104.16.89	192.168.100.22	TLSv1.2	273	Application Data
487	13.509427	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]

Figura 92

Paquetes al enviar instrucción redireccionada de encendido del cuarto A al Nodo 1

$$Tiempo\ respuesta = 30.896 - 29.162 = 1.734\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
598	29.162947	192.168.100.22	172.104.16.89	TLSv1.2	229	Application Data
600	29.257854	172.104.16.89	192.168.100.22	TCP	54	5222 → 57508 [ACK]
601	29.257910	192.168.100.22	172.104.16.89	TLSv1.2	109	Application Data
602	29.351698	172.104.16.89	192.168.100.22	TLSv1.2	116	Application Data
603	29.357991	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
604	29.358085	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
641	29.657762	172.104.16.89	192.168.100.22	TLSv1.2	257	Application Data
642	29.707173	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
876	30.849368	172.104.16.89	192.168.100.22	TLSv1.2	273	Application Data
877	30.896884	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]

Figura 93

Paquetes al enviar instrucción redireccionada de encendido del cuarto B al Nodo 1

$$Tiempo\ respuesta = 37.175 - 35.795 = 1.380\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
1062	35.795666	192.168.100.22	172.104.16.89	TLSv1.2	231	Application Data
1063	35.895937	172.104.16.89	192.168.100.22	TCP	54	5222 → 57508 [ACK]
1064	35.896009	192.168.100.22	172.104.16.89	TLSv1.2	109	Application Data
1070	35.989807	172.104.16.89	192.168.100.22	TLSv1.2	116	Application Data
1071	35.995356	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
1072	35.995466	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
1091	36.218928	172.104.16.89	192.168.100.22	TLSv1.2	257	Application Data
1110	36.265665	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
1280	37.127665	172.104.16.89	192.168.100.22	TLSv1.2	273	Application Data
1296	37.175678	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]

De las figuras 93, 94 y 95 se consigue un tiempo de respuesta promedio de 1.506s, es decir, la comunicación tarda un segundo y medio en completarse, esto corresponde a poco más del doble de tiempo que en una comunicación de ruta directa. Este resultado es el esperado ya que se aumenta el tiempo de comunicación entre el nodo 2 y el nodo 1 para obtener la respuesta y llevar a cabo la acción solicitada por el cliente. A continuación, se muestran los resultados de realizar la solicitud de temperatura con una ruta redireccionada.

Figura 94

Paquetes al enviar instrucción redireccionada de temp. del N1 al N2

$$Tiempo\ respuesta = 7.922 - 1.245 = 6.677\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
4	1.245799	192.168.100.22	172.104.16.89	TLSv1.2	248	Application Data
5	1.389302	172.104.16.89	192.168.100.22	TCP	54	5222 → 57508 [ACK]
6	1.389429	192.168.100.22	172.104.16.89	TLSv1.2	109	Application Data
86	1.484341	172.104.16.89	192.168.100.22	TLSv1.2	116	Application Data
88	1.489177	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
89	1.489244	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
300	1.649865	172.104.16.89	192.168.100.22	TLSv1.2	283	Application Data
301	1.650648	192.168.100.22	172.104.16.89	TLSv1.2	117	Application Data
354	1.749676	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
574	2.530803	172.104.16.89	192.168.100.22	TLSv1.2	273	Application Data
575	2.576450	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
715	7.878691	172.104.16.89	192.168.100.22	TLSv1.2	277	Application Data
716	7.922770	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]

Figura 95

Paquetes al enviar instrucción redireccionada temp. del N2 al N1

$$Tiempo\ respuesta = 12.673 - 6.030 = 6.643\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
46	6.030145	192.168.100.22	172.104.16.89	TLSv1.2	249	Application Data
47	6.125351	172.104.16.89	192.168.100.22	TCP	54	5222 → 57508 [ACK]
48	6.125405	192.168.100.22	172.104.16.89	TLSv1.2	109	Application Data
64	6.222960	172.104.16.89	192.168.100.22	TLSv1.2	116	Application Data
65	6.225554	172.104.16.89	192.168.100.22	TCP	56	5222 → 57508 [ACK]
66	6.225676	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
73	6.422720	172.104.16.89	192.168.100.22	TLSv1.2	257	Application Data
92	6.464715	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
281	7.296467	172.104.16.89	192.168.100.22	TLSv1.2	273	Application Data
299	7.344500	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]
432	12.625814	172.104.16.89	192.168.100.22	TLSv1.2	275	Application Data
464	12.673144	192.168.100.22	172.104.16.89	TCP	54	57508 → 5222 [ACK]

Cómo se puede ver en las figuras 96 y 97 el tiempo de respuesta de ambos nodos es prácticamente el mismo, 6.6s. Tienen una diferencia de casi un segundo con respecto a la comunicación directa, esto es debido a la comunicación extra que deben realizar los nodos.

Ruta directa cliente Xabber (Android)

Los tiempos obtenidos al enviar la instrucción directa de apagado a los nodos 1 y 2 desde las diferentes ubicaciones fueron los siguientes:

Figura 96

Paquetes al enviar instrucción directa de apagado de la cocina al Nodo 1

$$Tiempo\ respuesta = 0.487 - 0.000 = 0.487\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.10.1	172.104.16.89	TLSv1.2	411	Application Data
2	0.001413	172.104.16.89	10.1.10.1	TCP	40	5222 → 45792 [ACK]
3	0.094743	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
4	0.095154	10.1.10.1	172.104.16.89	TCP	40	45792 → 5222 [ACK]
5	0.396184	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
6	0.396944	10.1.10.1	172.104.16.89	TCP	40	45792 → 5222 [ACK]
7	0.486795	172.104.16.89	10.1.10.1	TLSv1.2	310	Application Data
8	0.487366	10.1.10.1	172.104.16.89	TCP	40	45792 → 5222 [ACK]

Figura 97

Paquetes al enviar instrucción directa de apagado de la sala al Nodo 1

$$Tiempo\ respuesta = 12.149 - 11.608 = 0.541\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
9	11.608543	10.1.10.1	172.104.16.89	TLSv1.2	409	Application Data
10	11.609495	172.104.16.89	10.1.10.1	TCP	40	5222 → 45792 [ACK]
11	11.703108	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
12	11.703436	10.1.10.1	172.104.16.89	TCP	40	45792 → 5222 [ACK]
13	12.053309	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
14	12.054148	10.1.10.1	172.104.16.89	TCP	40	45792 → 5222 [ACK]
15	12.147717	172.104.16.89	10.1.10.1	TLSv1.2	308	Application Data
16	12.149165	10.1.10.1	172.104.16.89	TCP	40	45792 → 5222 [ACK]

Figura 98

Paquetes al enviar instrucción directa de apagado del comedor al Nodo 1

$$Tiempo\ respuesta = 19.914 - 19.297 = 0.617\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
17	19.297833	10.1.10.1	172.104.16.89	TLSv1.2	469	Application Data
18	19.298615	172.104.16.89	10.1.10.1	TCP	40	5222 → 45792 [ACK]
19	19.488849	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
20	19.489562	10.1.10.1	172.104.16.89	TCP	40	45792 → 5222 [ACK]
21	19.812263	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
22	19.812743	10.1.10.1	172.104.16.89	TCP	40	45792 → 5222 [ACK]
23	19.910514	172.104.16.89	10.1.10.1	TLSv1.2	336	Application Data
24	19.910950	10.1.10.1	172.104.16.89	TCP	40	45792 → 5222 [ACK]
25	19.913973	10.1.10.1	172.104.16.89	TLSv1.2	102	Application Data
26	19.914690	172.104.16.89	10.1.10.1	TCP	40	5222 → 45792 [ACK]

De las figuras 98, 99 y 100 se puede evidenciar la comunicación del cliente con el nodo 1 consiguiendo un tiempo de respuesta promedio de 0.548s, es decir, la comunicación tarda poco más de medio segundo en completarse, similar al tiempo con el cliente de PC.

Ahora, la comunicación del cliente con el nodo 2 resultó así:

Figura 99

Paquetes al enviar instrucción directa de apagado del baño al Nodo 2

$$Tiempo\ respuesta = 0.573 - 0.000 = 0.573\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.10.1	172.104.16.89	TLSv1.2	410	Application Data
2	0.000624	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
3	0.132807	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
4	0.133349	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
5	0.477045	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
6	0.481241	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
7	0.573714	172.104.16.89	10.1.10.1	TLSv1.2	310	Application Data
8	0.573984	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]

Figura 100

Paquetes al enviar instrucción directa de apagado del cuarto A al Nodo 2

$$Tiempo\ respuesta = 8.591 - 8.101 = 0.490\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
9	8.101814	10.1.10.1	172.104.16.89	TLSv1.2	414	Application Data
10	8.103516	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
11	8.194031	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
12	8.194283	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
13	8.492537	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
14	8.492951	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
15	8.588108	172.104.16.89	10.1.10.1	TLSv1.2	313	Application Data
16	8.591460	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]

Figura 101

Paquetes al enviar instrucción directa de apagado del cuarto B al Nodo 2

$$Tiempo\ respuesta = 14.188 - 13.601 = 0.587\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
17	13.601563	10.1.10.1	172.104.16.89	TLSv1.2	473	Application Data
18	13.602276	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
19	13.698264	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
20	13.698634	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
21	14.093771	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
22	14.094534	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
23	14.185470	172.104.16.89	10.1.10.1	TLSv1.2	316	Application Data
24	14.188629	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]

De las figuras 101, 102 y 103 se puede evidenciar la comunicación del cliente con el nodo 2 consiguiendo un tiempo de respuesta promedio de 0.550s, es decir, la comunicación tarda poco más de medio segundo en completarse, similar al tiempo con el cliente de PC. A continuación, se presentan los tiempos obtenidos al enviar la instrucción directa de encendido a los nodos 1 y 2:

Figura 102

Paquetes al enviar instrucción directa de encendido de la cocina al Nodo 1

$$Tiempo\ respuesta = 0.569 - 0.000 = 0.569\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.10.1	172.104.16.89	TLSv1.2	413	Application Data
2	0.000489	172.104.16.89	10.1.10.1	TCP	40	5222 → 48037 [ACK]
3	0.094686	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
4	0.095076	10.1.10.1	172.104.16.89	TCP	40	48037 → 5222 [ACK]
5	0.474082	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
6	0.474785	10.1.10.1	172.104.16.89	TCP	40	48037 → 5222 [ACK]
7	0.569204	172.104.16.89	10.1.10.1	TLSv1.2	314	Application Data
8	0.569786	10.1.10.1	172.104.16.89	TCP	40	48037 → 5222 [ACK]

Figura 103

Paquetes al enviar instrucción directa de encendido de la sala al Nodo 1

$$Tiempo\ respuesta = 29.052 - 28.525 = 0.527\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
9	28.525054	10.1.10.1	172.104.16.89	TLSv1.2	468	Application Data
10	28.525773	172.104.16.89	10.1.10.1	TCP	40	5222 → 48037 [ACK]
11	28.618703	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
12	28.619099	10.1.10.1	172.104.16.89	TCP	40	48037 → 5222 [ACK]
13	28.919229	172.104.16.89	10.1.10.1	TLSv1.2	294	Application Data
14	28.919771	10.1.10.1	172.104.16.89	TCP	40	48037 → 5222 [ACK]
15	28.929763	10.1.10.1	172.104.16.89	TLSv1.2	102	Application Data
16	28.930613	172.104.16.89	10.1.10.1	TCP	40	5222 → 48037 [ACK]
17	29.015557	172.104.16.89	10.1.10.1	TLSv1.2	312	Application Data
18	29.052687	10.1.10.1	172.104.16.89	TCP	40	48037 → 5222 [ACK]

Figura 104

Paquetes al enviar instrucción directa de encendido del comedor al Nodo 1

$$Tiempo\ respuesta = 35.592 - 35.075 = 0.517\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
19	35.075033	10.1.10.1	172.104.16.89	TLSv1.2	471	Application Data
20	35.075756	172.104.16.89	10.1.10.1	TCP	40	5222 → 48037 [ACK]
21	35.169377	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
22	35.172597	10.1.10.1	172.104.16.89	TCP	40	48037 → 5222 [ACK]
23	35.488940	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
24	35.489474	10.1.10.1	172.104.16.89	TCP	40	48037 → 5222 [ACK]
25	35.591724	172.104.16.89	10.1.10.1	TLSv1.2	313	Application Data
26	35.592861	10.1.10.1	172.104.16.89	TCP	40	48037 → 5222 [ACK]

De las figuras 104, 105 y 106 se puede evidenciar la comunicación del cliente con el nodo 1 consiguiendo un tiempo de respuesta promedio de 0.537s, es decir, la comunicación tarda poco más de medio segundo en completarse, similar al tiempo cuando se envía la instrucción de apagado.

Ahora, los tiempos obtenidos con la instrucción de encendido del cliente con el nodo 2 resultó de la siguiente manera:

Figura 105

Paquetes al enviar instrucción directa de encendido del baño al Nodo 2

$$Tiempo\ respuesta = 0.489 - 0.000 = 0.489\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.10.1	172.104.16.89	TLSv1.2	487	Application Data
2	0.005851	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
3	0.093105	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
4	0.093395	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
5	0.396562	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
6	0.397638	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
7	0.488229	172.104.16.89	10.1.10.1	TLSv1.2	311	Application Data
8	0.489329	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]

Figura 106

Paquetes al enviar instrucción directa de encendido del cuarto A al Nodo 2

$$Tiempo\ respuesta = 11.518 - 10.912 = 0.606\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
9	10.912065	10.1.10.1	172.104.16.89	TLSv1.2	473	Application Data
10	10.912811	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
11	11.004650	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
12	11.005034	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
13	11.412248	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
14	11.413641	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
15	11.515484	172.104.16.89	10.1.10.1	TLSv1.2	340	Application Data
16	11.515790	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
17	11.517665	10.1.10.1	172.104.16.89	TLSv1.2	102	Application Data
18	11.518273	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]

Figura 107

Paquetes al enviar instrucción directa de encendido del cuarto B al Nodo 2

$$Tiempo\ respuesta = 19.167 - 18.666 = 0.501\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
19	18.666615	10.1.10.1	172.104.16.89	TLSv1.2	475	Application Data
20	18.669418	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
21	18.759833	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
22	18.798723	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
23	19.077553	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
24	19.080324	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
25	19.166978	172.104.16.89	10.1.10.1	TLSv1.2	316	Application Data
26	19.167729	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]

El promedio de tiempo de respuesta obtenido de las figuras 108 al 109 es 0.535s, sigue siendo inferior a un segundo y no tiene una diferencia notable en los tiempos de acuerdo al sitio. El comportamiento es prácticamente igual que con el nodo 1.

Ahora, en cuanto a comunicación de ruta directa, resta analizar las peticiones a ambos nodos sobre la temperatura captada por sus sensores. Los datos obtenidos son los siguientes:

Figura 108

Paquetes al enviar instrucción directa de temperatura al Nodo 1

$$Tiempo\ respuesta = 5.903 - 0.000 = 5.903\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.10.1	172.104.16.89	TCP	488	48037 → 5222 [PSH,
2	0.000501	172.104.16.89	10.1.10.1	TCP	40	5222 → 48037 [ACK]
3	0.093001	172.104.16.89	10.1.10.1	TCP	102	5222 → 48037 [ACK]
4	0.093586	10.1.10.1	172.104.16.89	TCP	40	48037 → 5222 [ACK]
5	0.481807	172.104.16.89	10.1.10.1	TCP	268	5222 → 48037 [ACK]
6	0.482890	10.1.10.1	172.104.16.89	TCP	40	48037 → 5222 [ACK]
7	0.576655	172.104.16.89	10.1.10.1	TCP	312	5222 → 48037 [ACK]
8	0.576933	10.1.10.1	172.104.16.89	TCP	40	48037 → 5222 [ACK]
9	5.899183	172.104.16.89	10.1.10.1	TCP	342	5222 → 48037 [ACK]
10	5.903073	10.1.10.1	172.104.16.89	TCP	40	48037 → 5222 [ACK]

Figura 109

Paquetes al enviar instrucción directa de temperatura al Nodo 2

$$Tiempo\ respuesta = 5.885 - 0.000 = 5.885\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.10.1	172.104.16.89	TCP	490	49475 → 5222 [PSH,
2	0.000667	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
3	0.098529	172.104.16.89	10.1.10.1	TCP	102	5222 → 49475 [ACK]
4	0.131890	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
5	0.497188	172.104.16.89	10.1.10.1	TCP	268	5222 → 49475 [ACK]
6	0.498151	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
7	0.587251	172.104.16.89	10.1.10.1	TCP	312	5222 → 49475 [ACK]
8	0.588021	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
9	5.884064	172.104.16.89	10.1.10.1	TCP	316	5222 → 49475 [ACK]
10	5.885692	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]

Cómo se puede ver en las figuras 110 y 111 el tiempo de respuesta de ambos nodos es prácticamente el mismo, 5.85s en promedio. Nuevamente, como con el cliente Pidgin, la respuesta, al tratarse de la medición de la temperatura, tarda más tiempo, ya que se debe tomar el tiempo en que tarda en hacer la medición el nodo.

Ruta redireccionada cliente Xabber (Android)

Primero se probó la comunicación redireccionada al solicitar al nodo 2 el apagado de los actuadores del nodo 1, los datos obtenidos fueron los siguientes:

Figura 110

Paquetes al enviar instrucción redireccionada de apagado de la cocina al Nodo 2

$$Tiempo\ respuesta = 1.465 - 0.000 = 1.465\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.10.1	172.104.16.89	TLSv1.2	410	Application Data
2	0.000765	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
3	0.093110	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
4	0.093400	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
5	0.473257	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
6	0.474476	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
7	1.462640	172.104.16.89	10.1.10.1	TLSv1.2	257	Application Data
8	1.465519	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]

Figura 111

Paquetes al enviar instrucción redireccionada de apagado de la sala al Nodo 2

$$Tiempo\ respuesta = 16.377 - 14.937 = 1.440\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
9	14.937630	10.1.10.1	172.104.16.89	TLSv1.2	471	Application Data
10	14.938529	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
11	15.032096	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
12	15.032401	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
13	15.412270	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
14	15.413376	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
15	16.375630	172.104.16.89	10.1.10.1	TLSv1.2	260	Application Data
16	16.377111	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]

Figura 112

Paquetes al enviar instrucción redireccionada de apagado del comedor al Nodo 2

$$Tiempo\ respuesta = 30.319 - 29.009 = 1.310\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
17	29.009801	10.1.10.1	172.104.16.89	TLSv1.2	473	Application Data
18	29.010377	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
19	29.103403	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
20	29.103741	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
21	29.401666	172.104.16.89	10.1.10.1	TLSv1.2	294	Application Data
22	29.403290	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
23	29.405968	10.1.10.1	172.104.16.89	TLSv1.2	102	Application Data
24	29.406766	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
25	30.284731	172.104.16.89	10.1.10.1	TLSv1.2	263	Application Data
26	30.319302	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]

De las figuras 112, 113 y 114 se consigue un tiempo de respuesta promedio de 1.405s, es decir, la comunicación tarda casi un segundo y medio en completarse, esto corresponde más del doble de tiempo que en una comunicación de ruta directa. Este resultado es el esperado ya que se aumenta el tiempo de comunicación entre el nodo 2 y el nodo 1 para obtener la respuesta y llevar a cabo la acción solicitada por el cliente. A continuación, se muestran los resultados de realizar el mismo experimento, pero solicitando al primer nodo la acción de apagado para los actuadores del segundo nodo.

Figura 113

Paquetes al enviar instrucción redireccionada de apagado del baño al Nodo 1

$$Tiempo\ respuesta = 1.426 - 0.000 = 1.426\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.10.1	172.104.16.89	TLSv1.2	468	Application Data
2	0.000848	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
3	0.095499	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
4	0.096147	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
5	0.485526	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
6	0.486937	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
7	1.425008	172.104.16.89	10.1.10.1	TLSv1.2	260	Application Data
8	1.426821	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]

Figura 114

Paquetes al enviar instrucción redireccionada de apagado del cuarto A al Nodo 1

$$Tiempo\ respuesta = 9.527 - 8.233 = 1.294\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
9	8.233848	10.1.10.1	172.104.16.89	TLSv1.2	409	Application Data
10	8.234583	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
11	8.328608	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
12	8.329014	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
13	8.629456	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
14	8.630029	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
15	9.525862	172.104.16.89	10.1.10.1	TLSv1.2	258	Application Data
16	9.527450	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]

Figura 115

Paquetes al enviar instrucción redireccionada de apagado del cuarto B al Nodo 1

$$Tiempo\ respuesta = 18.980 - 17.665 = 1.315\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
17	17.665462	10.1.10.1	172.104.16.89	TLSv1.2	469	Application Data
18	17.666189	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
19	17.761607	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
20	17.761994	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
21	18.166354	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
22	18.169058	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
23	18.978748	172.104.16.89	10.1.10.1	TLSv1.2	259	Application Data
24	18.980317	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]

De las figuras 115, 116 y 117 se consigue un tiempo de respuesta promedio de 1.345s, es decir, la comunicación tarda casi un segundo y medio en completarse, esto corresponde a prácticamente el doble de tiempo que en una comunicación de ruta directa. Este resultado es el esperado ya que se aumenta el tiempo de comunicación entre el nodo 2 y el nodo 1 para obtener la respuesta y llevar a cabo la acción solicitada por el cliente, además, si se toma en cuenta la diferencia de 0.2s entre la anterior prueba y esta, se puede deber a la congestión de la red en el instante de realizar la petición. A continuación, se muestran los resultados de realizar el mismo experimento, pero en vez de solicitar el apagado de los actuadores se pide el encendido de estos.

Figura 116

Paquetes al enviar instrucción redireccionada de encendido de la cocina al Nodo 2

$$\boldsymbol{Tiempo\ respuesta = 1.421 - 0 = 1.421\ s}$$

1	0.000000	10.1.10.1	172.104.16.89	TLSv1.2	386 Application Data
2	0.003149	172.104.16.89	10.1.10.1	TCP	40 5222 → 49475 [ACK]
3	0.003613	10.1.10.1	172.104.16.89	TLSv1.2	95 Application Data
4	0.007788	172.104.16.89	10.1.10.1	TCP	40 5222 → 49475 [ACK]
5	0.230237	172.104.16.89	10.1.10.1	TLSv1.2	102 Application Data
6	0.235846	10.1.10.1	172.104.16.89	TCP	40 49475 → 5222 [ACK]
7	0.432954	172.104.16.89	10.1.10.1	TLSv1.2	294 Application Data
8	0.433443	10.1.10.1	172.104.16.89	TCP	40 49475 → 5222 [ACK]
9	0.436757	10.1.10.1	172.104.16.89	TLSv1.2	102 Application Data
10	0.437281	172.104.16.89	10.1.10.1	TCP	40 5222 → 49475 [ACK]
11	1.383863	172.104.16.89	10.1.10.1	TLSv1.2	258 Application Data
12	1.421956	10.1.10.1	172.104.16.89	TCP	40 49475 → 5222 [ACK]

Figura 117

Paquetes al enviar instrucción redireccionada de encendido de la sala al Nodo 2

$$Tiempo\ respuesta = 18.156 - 16.888 = 1.268\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
13	16.888862	10.1.10.1	172.104.16.89	TLSv1.2	473	Application Data
14	16.889597	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
15	16.982485	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
16	16.982884	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
17	17.280337	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
18	17.282289	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
19	18.154756	172.104.16.89	10.1.10.1	TLSv1.2	261	Application Data
20	18.156386	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]

Figura 118

Paquetes al enviar instrucción redireccionada de encendido del comedor al Nodo 2

$$Tiempo\ respuesta = 29.069 - 27.663 = 1.406\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
21	27.663729	10.1.10.1	172.104.16.89	TLSv1.2	475	Application Data
22	27.664757	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
23	27.757488	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
24	27.758083	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
25	28.083604	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
26	28.086191	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
27	29.068354	172.104.16.89	10.1.10.1	TLSv1.2	263	Application Data
28	29.069945	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]

De las figuras 118, 119 y 120 se consigue un tiempo de respuesta promedio de 1.365s, es decir, la comunicación tarda poco más de un segundo en completarse, esto corresponde a prácticamente el doble de tiempo que en una comunicación de ruta directa. Este resultado es el esperado ya que se aumenta el tiempo de comunicación entre el nodo 2 y el nodo 1 para obtener la respuesta y llevar a cabo la acción solicitada por el cliente. A continuación, se muestran los resultados de realizar el mismo experimento, pero solicitando al primer nodo la acción de encendido para los actuadores del segundo nodo.

Figura 119

Paquetes al enviar instrucción redireccionada de encendido del baño al Nodo 1

$$Tiempo\ respuesta = 1.651 - 0.000 = 1.651\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.10.1	172.104.16.89	TLSv1.2	413	Application Data
2	0.001303	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
3	0.093480	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
4	0.093881	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
5	0.476461	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
6	0.477374	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
7	1.651274	172.104.16.89	10.1.10.1	TLSv1.2	261	Application Data
8	1.651879	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]

Figura 120

Paquetes al enviar instrucción redireccionada de encendido del cuarto A al Nodo 1

$$Tiempo\ respuesta = 13.151 - 11.806 = 1.345\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
9	11.806160	10.1.10.1	172.104.16.89	TLSv1.2	468	Application Data
10	11.806942	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
11	11.900180	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
12	11.902818	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
13	12.271622	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
14	12.272398	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
15	13.149319	172.104.16.89	10.1.10.1	TLSv1.2	259	Application Data
16	13.151070	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]

Figura 121

Paquetes al enviar instrucción redireccionada de encendido del cuarto B al Nodo 1

$$Tiempo\ respuesta = 20.402 - 19.066 = 1.336\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
17	19.066209	10.1.10.1	172.104.16.89	TLSv1.2	471	Application Data
18	19.066764	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
19	19.193911	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
20	19.194414	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
21	19.495869	172.104.16.89	10.1.10.1	TLSv1.2	294	Application Data
22	19.496696	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
23	19.503817	10.1.10.1	172.104.16.89	TLSv1.2	102	Application Data
24	19.505308	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
25	20.364148	172.104.16.89	10.1.10.1	TLSv1.2	260	Application Data
26	20.402228	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]

De las figuras 121, 122 y 123 se consigue un tiempo de respuesta promedio de 1.444s, es decir, la comunicación tarda casi un segundo y medio en completarse, esto corresponde a poco más del doble de tiempo que en una comunicación de ruta directa. Este resultado es el esperado ya que se aumenta el tiempo de comunicación entre el nodo 2 y el nodo 1 para obtener la respuesta y llevar a cabo la acción solicitada por el cliente. A continuación, se muestran los resultados de realizar la solicitud de temperatura con una ruta redireccionada.

Figura 122

Paquetes al enviar instrucción redireccionada de temperatura al Nodo 1

$$Tiempo\ respuesta = 6.643 - 0.000 = 6.643\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.10.1	172.104.16.89	TLSv1.2	433	Application Data
2	0.000963	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
3	0.093506	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
4	0.093907	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
5	0.391129	172.104.16.89	10.1.10.1	TLSv1.2	268	Application Data
6	0.391657	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
7	1.269351	172.104.16.89	10.1.10.1	TLSv1.2	259	Application Data
8	1.270764	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
9	6.637839	172.104.16.89	10.1.10.1	TLSv1.2	263	Application Data
10	6.643675	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]

Figura 123

Paquetes al enviar instrucción redireccionada de temperatura al Nodo 2

$$Tiempo\ respuesta = 6.701 - 0.000 = 6.701\ s$$

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.1.10.1	172.104.16.89	TLSv1.2	431	Application Data
2	0.000639	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
3	0.093947	172.104.16.89	10.1.10.1	TLSv1.2	102	Application Data
4	0.094377	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
5	0.450637	172.104.16.89	10.1.10.1	TLSv1.2	294	Application Data
6	0.452000	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
7	0.459430	10.1.10.1	172.104.16.89	TLSv1.2	103	Application Data
8	0.460518	172.104.16.89	10.1.10.1	TCP	40	5222 → 49475 [ACK]
9	1.495010	172.104.16.89	10.1.10.1	TLSv1.2	259	Application Data
10	1.533959	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]
11	6.700332	172.104.16.89	10.1.10.1	TLSv1.2	261	Application Data
12	6.701621	10.1.10.1	172.104.16.89	TCP	40	49475 → 5222 [ACK]

Cómo se puede ver en las figuras 124 y 125 el tiempo de respuesta de ambos nodos es prácticamente el mismo, 6.67s en promedio. Tienen una diferencia de casi un segundo con respecto a la comunicación directa, esto es debido a la comunicación extra que deben realizar los nodos.

Throughput y retransmisiones

Para medir la cantidad de bytes por segundo transmitidos del cliente al servidor o viceversa se utilizó la herramienta I/O Graph, esta permite utilizar filtros de visualización para mostrar el tráfico de interés en este caso la conversación entre Pidgin de Windows o Xabber de Android y el servidor público. Para medir el *Throughput* se seleccionó la carga útil de la trama TCP, ya que aquí se encuentra el flujo XML que corresponde a los mensajes enviados mediante el protocolo XMPP, también se aplicó un filtro para contar el número de retransmisiones que se pueden producir cuando no se recibe un ACK de respuesta.

Se puede notar en los gráficos de *throughput* que existen barras muy próximas en diferentes instantes, esto se debe a que generalmente la barra más pequeña corresponde al mensaje enviado desde el cliente mientras que la otra corresponde al mensaje de respuesta del servidor. En las opciones del eje y

se seleccionó el campo "*MAX*" que permite obtener durante un intervalo de un segundo el valor máximo de carga útil, este corresponde, en el protocolo XMPP, a mensajes de tipo chat en los que se encuentra contenido entre etiquetas el destinatario y el texto encriptado, por esta razón se puede presentar el caso de que durante el mismo intervalo de tiempo, antes de alcanzar un segundo, se encuentre el mensaje del cliente y del servidor, por lo tanto se selecciona automáticamente el de mayor tamaño, este comportamiento se puede observar en algunas gráficas.

Throughput y retransmisiones en ruta directa cliente Pidgin (Windows)

Para analizar el *Throughput* se realizaron graficas que cuentan con las tres instrucciones de apagado o encendido en una captura para agilizar el proceso, tanto de toma de información como para su análisis. A continuación, se muestran los datos obtenidos y su análisis.

Figura 124

Throughput y retransmisiones al enviar instrucciones directas de apagado al nodo 1

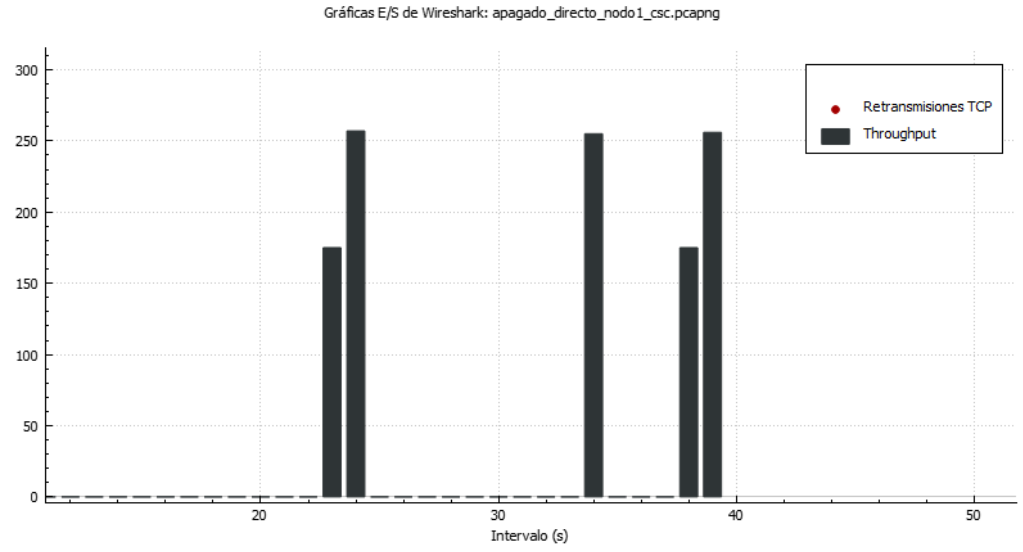
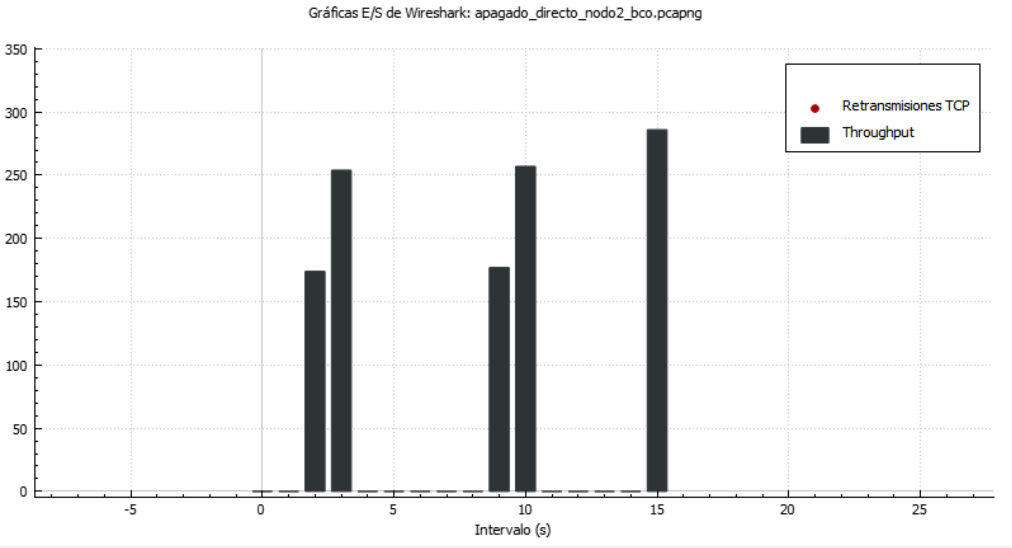


Figura 125

Throughput y retransmisiones al enviar instrucciones directas de apagado al nodo 2



Se puede observar en la figura 126 y 127 que las transmisiones desde el cliente de Windows van a 175 Bytes/s en promedio, mientras que los mensajes trasmitidos desde el servidor son aproximadamente de 258 Bytes/s, esto se debe a que la información de respuesta contenida en la etiqueta *body* del flujo xml es mayor que la solicitud de apagado enviada por el cliente. También se puede notar que no existen retransmisiones durante la conversación.

Figura 126

Throughput y retransmisiones al enviar instrucciones directas de encendido al nodo 1

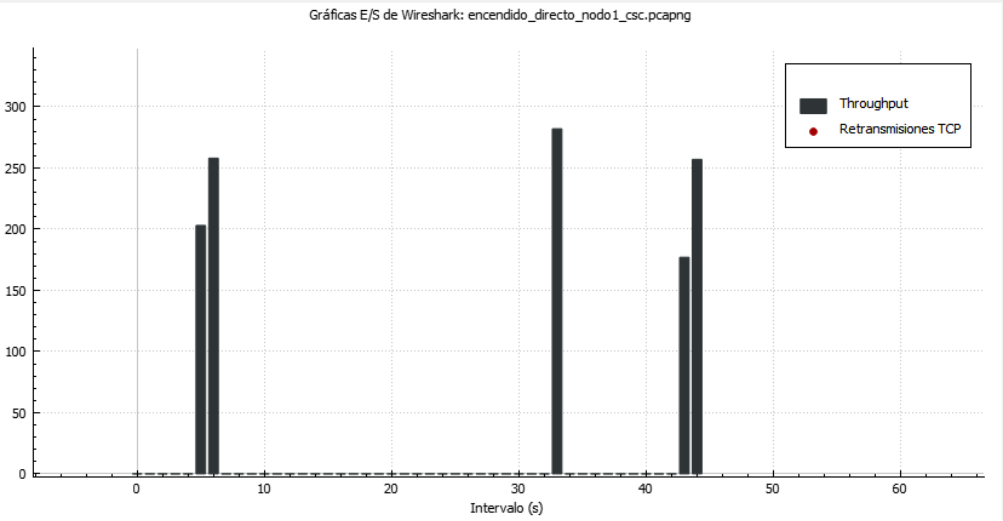
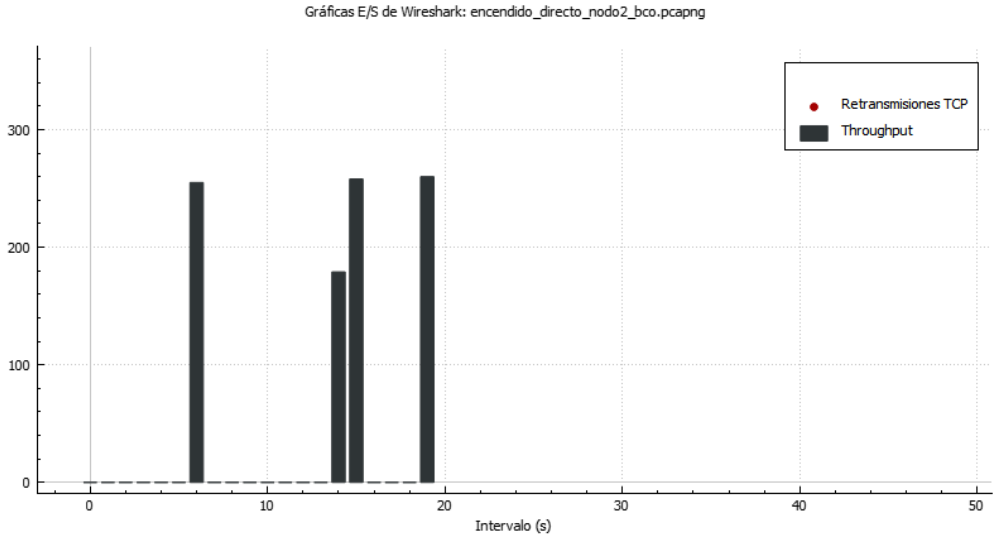


Figura 127

Throughput y retransmisiones al enviar instrucciones directas de encendido al nodo 2



Se puede observar en la figura 128 y 129 que las transmisiones desde el cliente de Windows van a 180 Bytes/s en promedio, esto se obtuvo de los datos de Wireshark ya que en la gráfica se superponen las barras, mientras que los mensajes trasmitidos desde el servidor son aproximadamente de 260 Bytes/s. Cabe

recalcar que en las transmisiones al nodo 2 existió simetría en el *Throughput* mientras que en las del nodo 1 fluctuaron. También se puede notar que no existen retransmisiones durante la conversación.

Figura 128

Throughput y retransmisiones al enviar instrucciones directas de temperatura al nodo 1

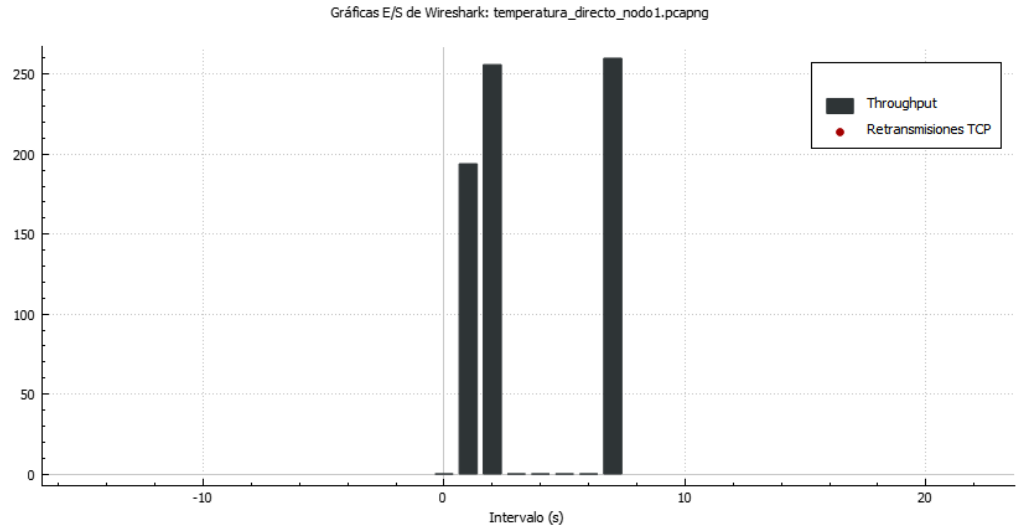
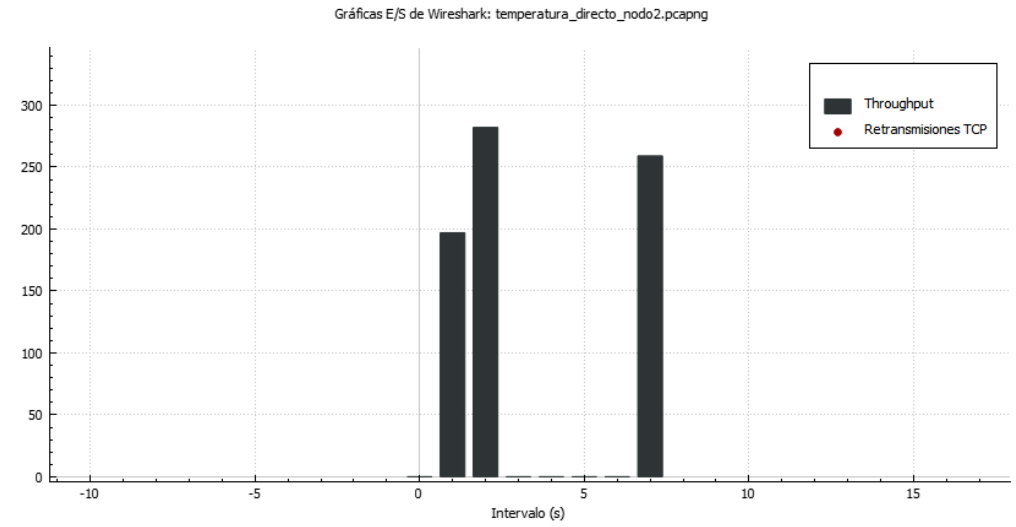


Figura 129

Throughput y retransmisiones al enviar instrucciones directas de temperatura al nodo 2



Se puede observar en la figura 130 y 131 que las transmisiones desde el cliente de Windows tienen un *Throughput* menor al que las que van de servidor al cliente, además el servidor envía dos respuestas una explicando que en un momento se enviarán los datos y la otra con los datos solicitados de la toma de temperatura ya procesados. Se nota un promedio de transmisión de 190Bytes/s en los mensajes del cliente, mientras que los mensajes trasmitidos desde el servidor son aproximadamente de 265 Bytes/s. También se puede notar que no existen retransmisiones durante la conversación.

Throughput y retransmisiones en ruta redireccionada cliente Pidgin (Windows)

Figura 130

Throughput y retransmisiones al enviar instrucciones redireccionadas de apagado al nodo 1

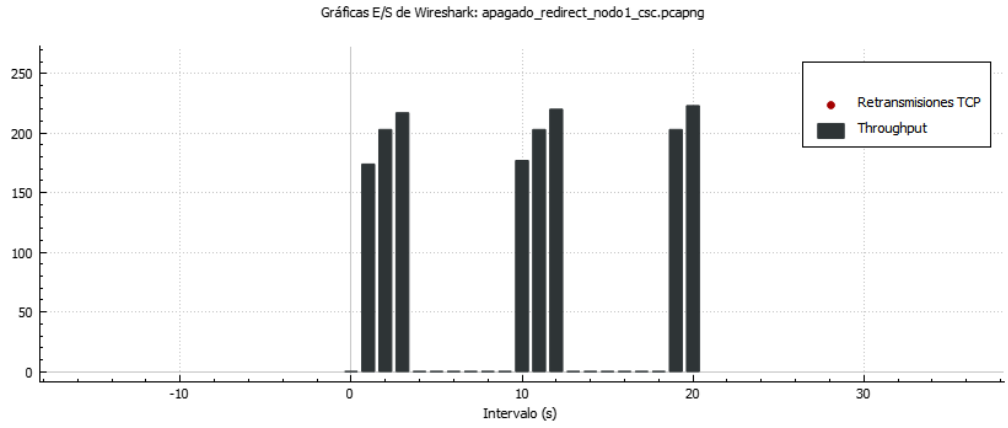
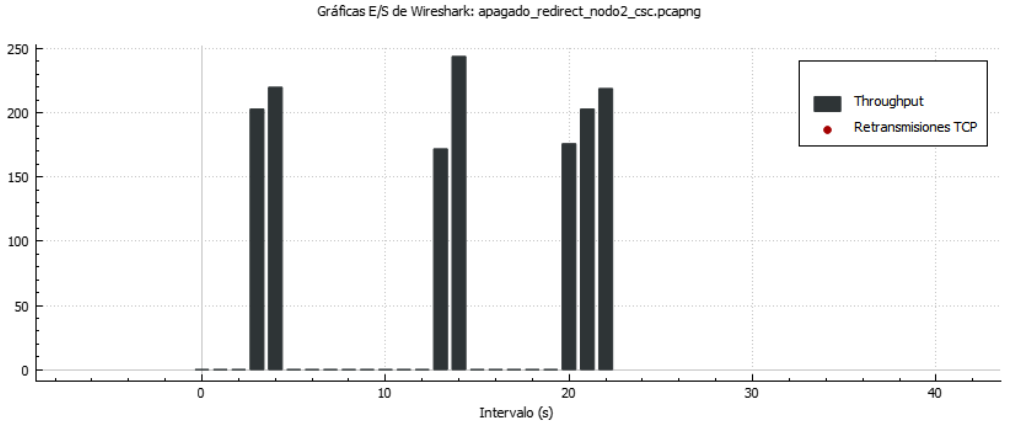


Figura 131

Throughput y retransmisiones al enviar instrucciones redireccionadas de apagado al nodo 2



Se puede observar en las figuras 132 y 133 que se ven hasta tres barras por conversación, esto se debe a que el programa captó una confirmación entre servidor y el cliente, esta es la primera barra y no se tomará en cuenta para el análisis. Las transmisiones desde el cliente de Windows van a 205 Bytes/s en promedio, mientras que los mensajes trasmitidos desde el servidor son aproximadamente de 225 Bytes/s. También se puede notar que no existen retransmisiones durante la conversación.

Figura 132

Throughput y retransmisiones al enviar instrucciones redireccionadas de encendido al nodo 1

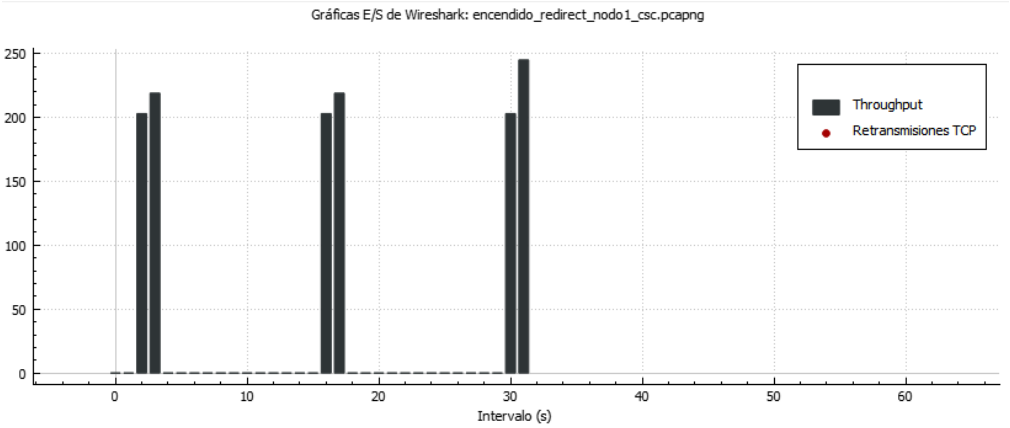
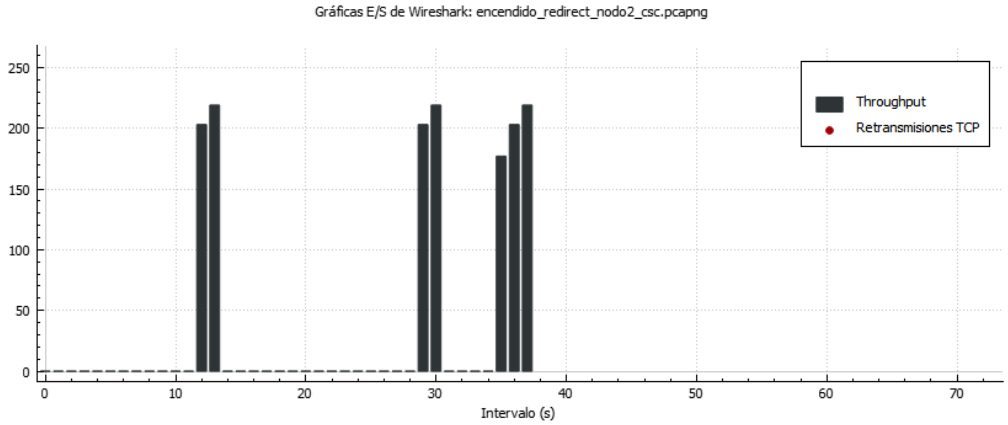


Figura 133

Throughput y retransmisiones al enviar instrucciones redireccionadas de encendido al nodo 2



Es posible notar en las figuras 134 y 135 que las transmisiones desde el cliente de Windows van a 205 Bytes/s en promedio, mientras que los mensajes trasmitidos desde el servidor son aproximadamente de 220 Bytes/s. También se puede notar que no existen retransmisiones durante la conversación.

Figura 134

Throughput y retransmisiones al enviar instrucciones redireccionadas de temperatura al nodo 1

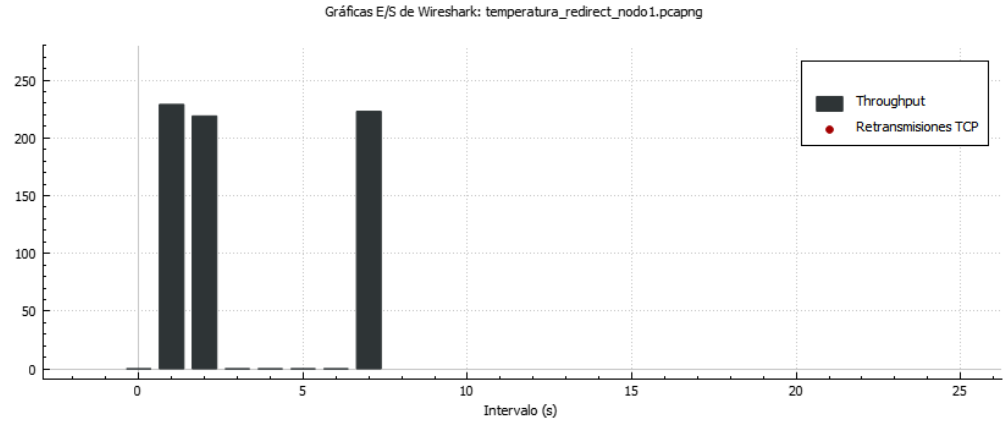
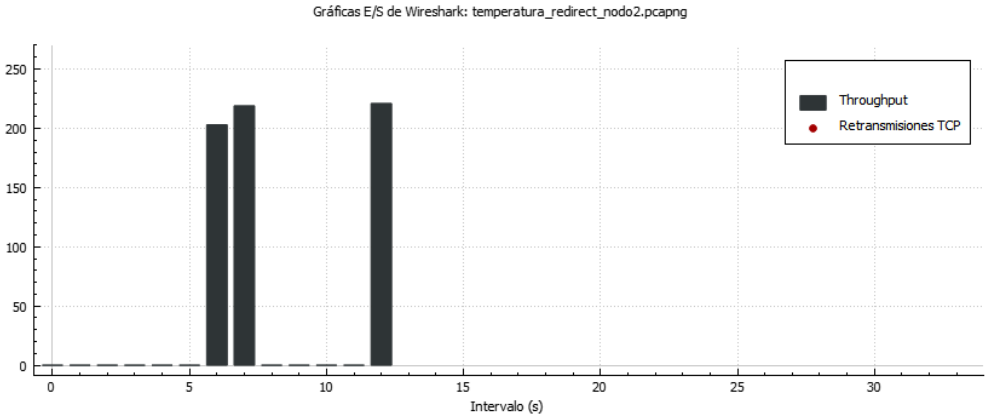


Figura 135

Throughput y retransmisiones al enviar instrucciones redireccionadas de temperatura al nodo 2



Se puede observar en la figura 136 y 137 que las transmisiones desde el cliente de Windows tienen un *Throughput* menor al que las que van de servidor al cliente, además el servidor envía dos respuestas una explicando que en un momento se enviarán los datos y la otra con los datos solicitados de la toma de temperatura ya procesados. Se nota un promedio de transmisión de 210Bytes/s en los mensajes del cliente, mientras que los mensajes trasmitidos desde el servidor son aproximadamente de 220 Bytes/s. También se puede notar que no existen retransmisiones durante la conversación.

Throughput y retransmisiones en ruta directa cliente Xabber (Android)

Figura 136

Throughput y retransmisiones al enviar instrucciones directas de apagado al nodo 1

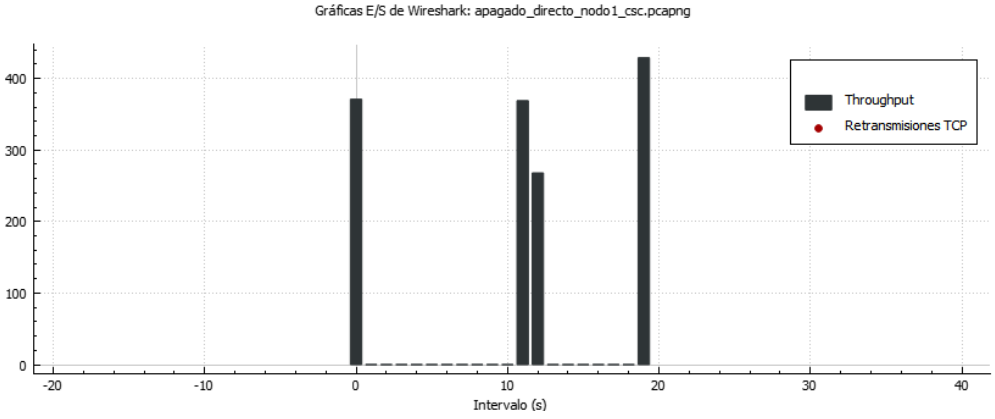
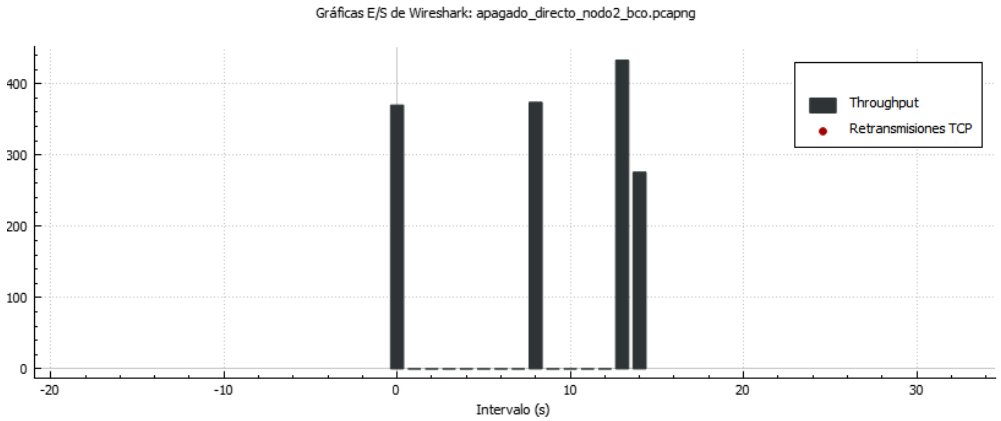


Figura 137

Throughput y retransmisiones al enviar instrucciones directas de apagado al nodo 2



Se puede observar en las figuras 138 y 139 que, a diferencia con el cliente de Android, los *Throughput* enviados desde el cliente son más grandes que los enviados por el servidor. Las transmisiones desde el cliente de Android van a 386.66 Bytes/s en promedio, mientras que los mensajes trasmitidos desde el servidor son aproximadamente de 280 Bytes/s, esto se debe a que la información de la petición puede contener términos de negociación del cliente con el servidor, tal como el tamaño mínimo o máximo de cada mensaje, entre otros. También se puede notar que no existen retransmisiones durante la conversación.

Figura 138

Throughput y retransmisiones al enviar instrucciones directas de encendido al nodo 1

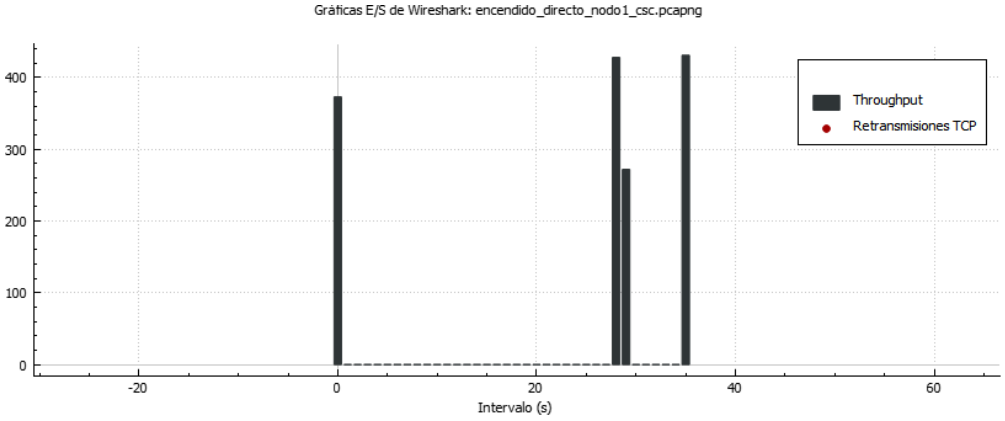
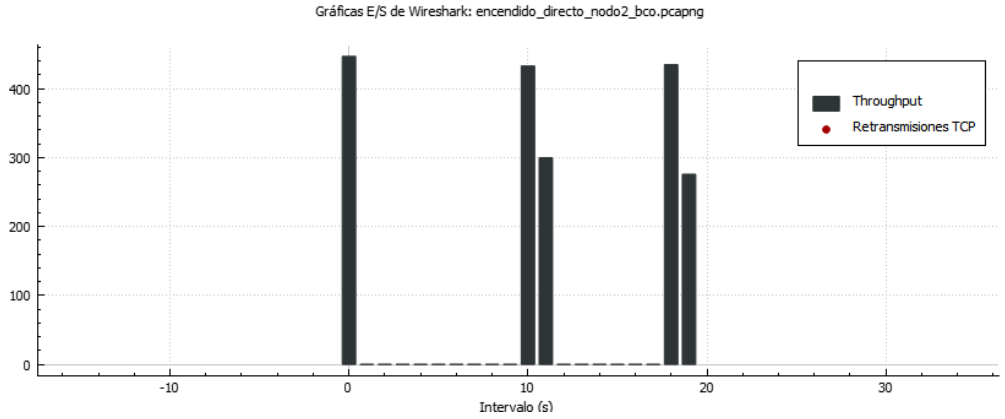


Figura 139

Throughput y retransmisiones al enviar instrucciones directas de encendido al nodo 2



Se nota de las figuras 140 y 141 que las transmisiones desde el cliente de Android van a 430 Bytes/s en promedio, mientras que los mensajes trasmitidos desde el servidor son aproximadamente de 290 Bytes/s. Es posible observar que no existen retransmisiones durante la conversación.

Figura 140

Throughput y retransmisiones al enviar instrucciones directas de temperatura al nodo 1

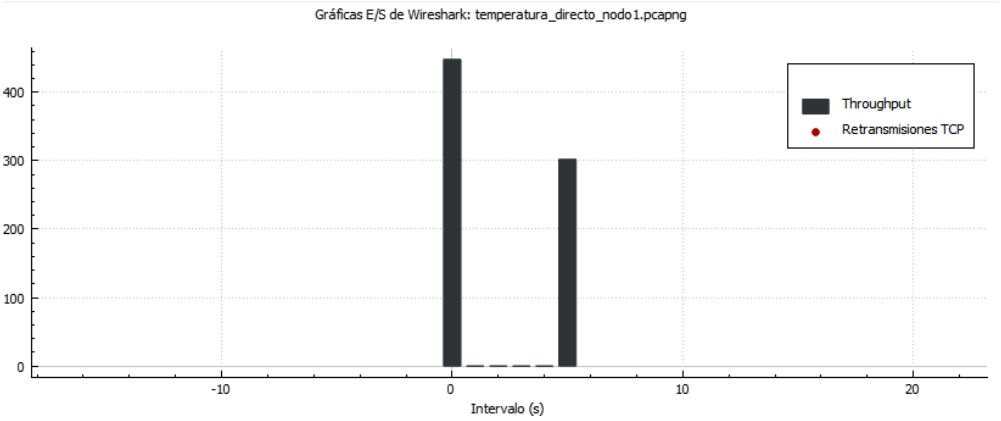
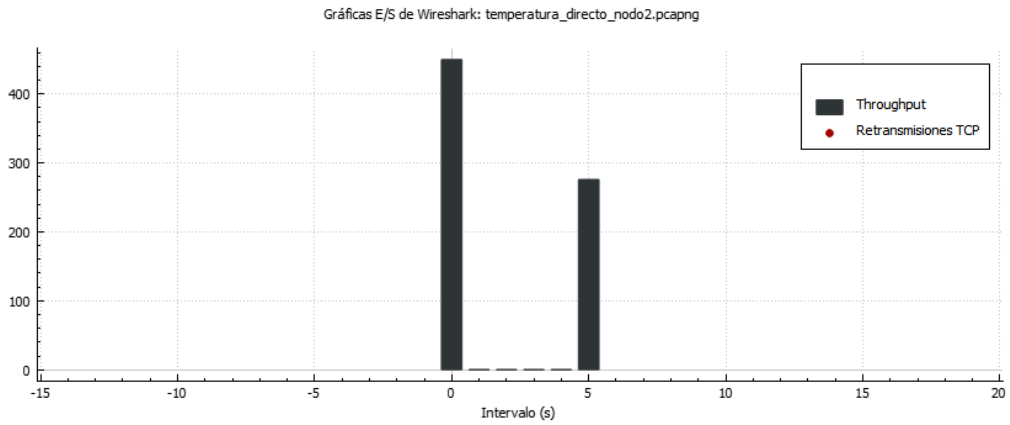


Figura 141

Throughput y retransmisiones al enviar instrucciones directas de temperatura al nodo 2



Es posible notar en las figuras 142 y 143 que las transmisiones desde el cliente de Android tienen un *Throughput* mayor al que las que van de servidor al cliente, además el servidor envía dos respuestas una explicando que en un momento se enviarán los datos y la otra con los datos solicitados de la toma de temperatura ya procesados, las primeras respuestas se entregan en el mismo segundo que la petición y es por ello que no salen en las gráficas de las figuras 142 y 143. Se nota un promedio de transmisión de 440Bytes/s en los mensajes del cliente, mientras que los mensajes trasmitidos desde el servidor son aproximadamente de 290 Bytes/s. También se puede notar que no existen retransmisiones durante la conversación.

Throughput y retransmisiones en ruta redireccionada cliente Xabber (Android)

Figura 142

Throughput y retransmisiones al enviar instrucciones redireccionadas de apagado al nodo 1

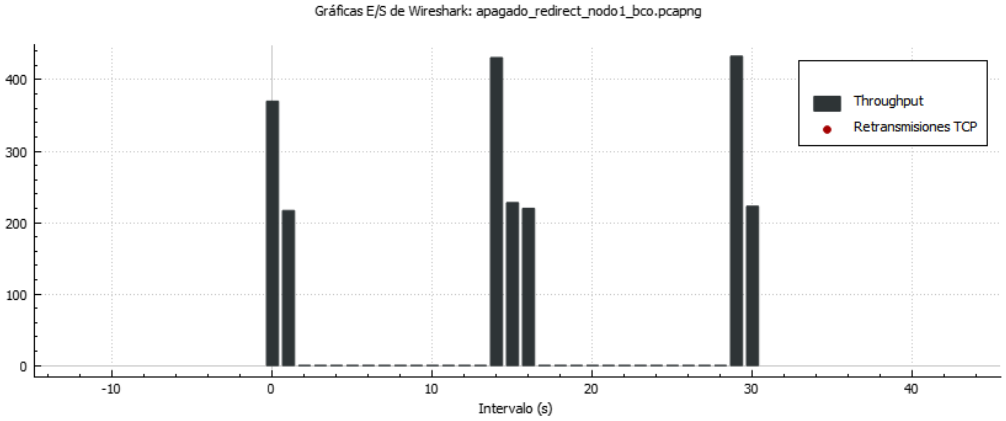
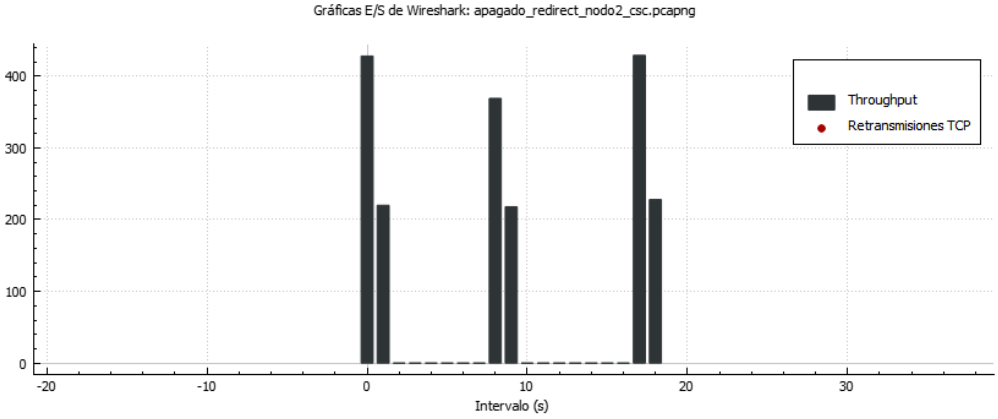


Figura 143

Throughput y retransmisiones al enviar instrucciones redireccionadas de apagado al nodo 2



Se puede observar en las figuras 144 y 145 que se ven hasta tres barras por conversación, esto se debe a que el programa captó una confirmación entre servidor y el cliente, esta es la última barra y no se tomará en cuenta para el análisis. Las transmisiones desde el cliente de Android van a 400 Bytes/s en promedio, mientras que los mensajes trasmitidos desde el servidor son aproximadamente de 220 Bytes/s. También se puede notar que no existen retransmisiones durante la conversación.

Figura 144

Throughput y retransmisiones al enviar instrucciones redireccionadas de encendido al nodo 1

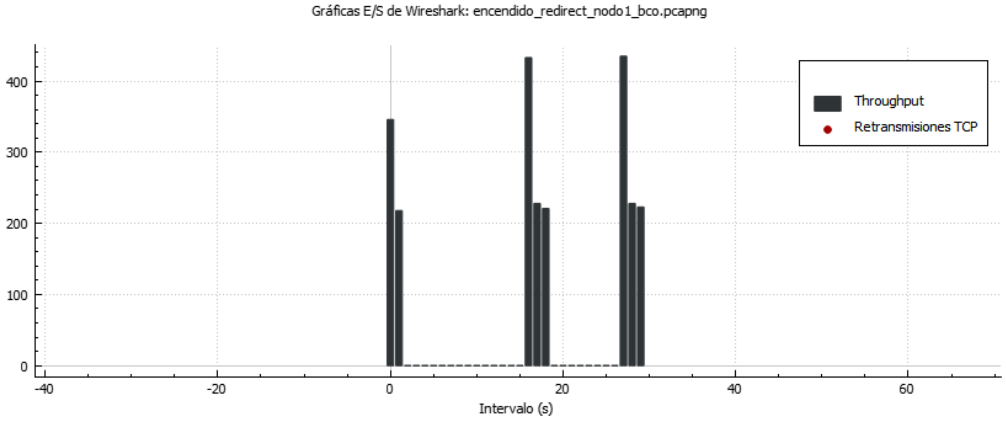
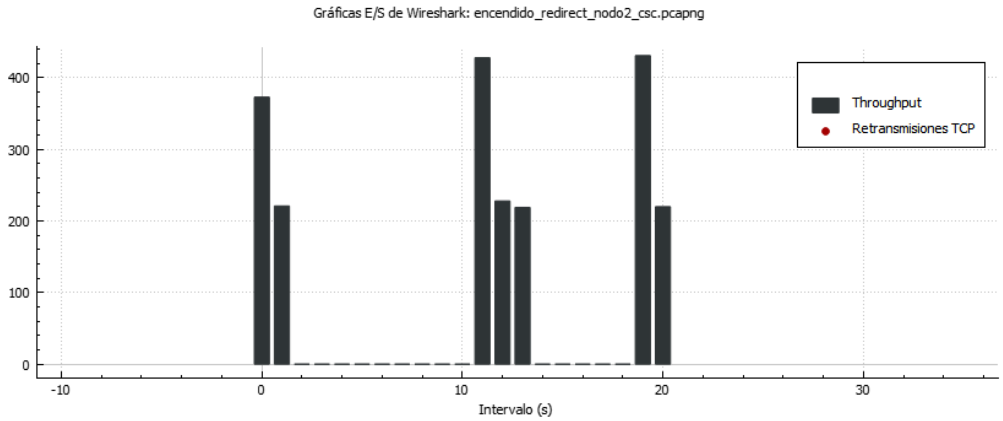


Figura 145

Throughput y retransmisiones al enviar instrucciones redireccionadas de encendido al nodo 2



Se nota en las figuras 146 y 147 que las transmisiones desde el cliente de Android van a 415Bytes/s en promedio, mientras que los mensajes trasmitidos desde el servidor son aproximadamente de 220 Bytes/s. También se puede notar que no existen retransmisiones durante la conversación.

Figura 146

Throughput y retransmisiones al enviar instrucciones redireccionadas de temperatura al nodo 1

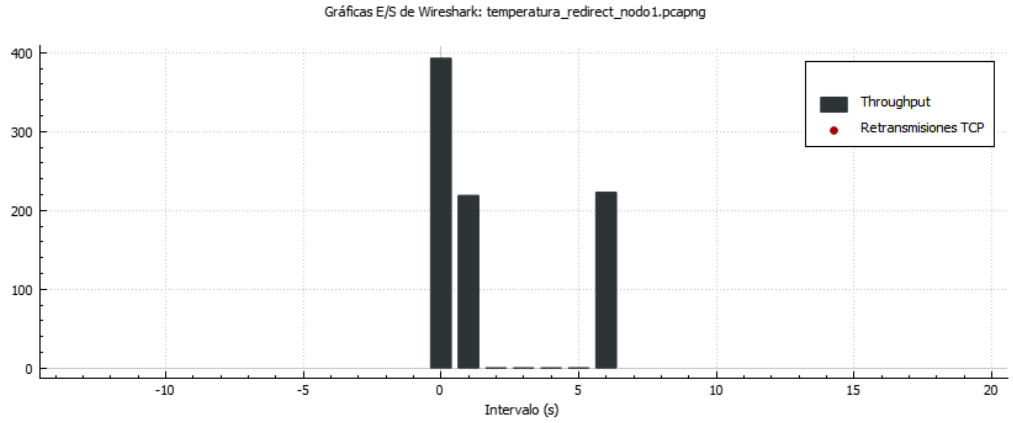
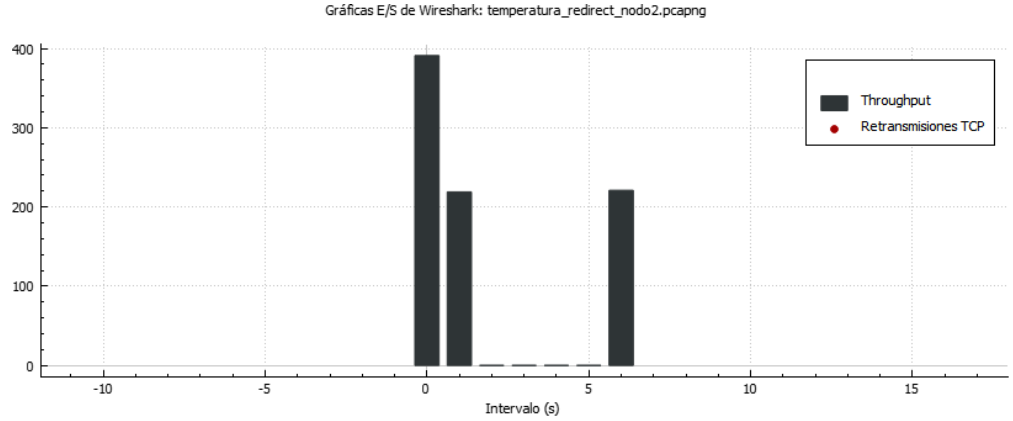


Figura 147

Throughput y retransmisiones al enviar instrucciones redireccionadas de temperatura al nodo 2



Es posible obtener de las figuras 148 y 149 que las transmisiones desde el cliente de Android tienen un *Throughput* mayor al que las que van de servidor al cliente, además el servidor envía dos respuestas una explicando que en un momento se enviarán los datos y la otra con los datos solicitados de la toma de temperatura ya procesados. Se nota un promedio de transmisión de 390Bytes/s en los mensajes del cliente, mientras que los mensajes trasmitidos desde el servidor son aproximadamente de 220 Bytes/s. También se puede notar que no existen retransmisiones durante la conversación.

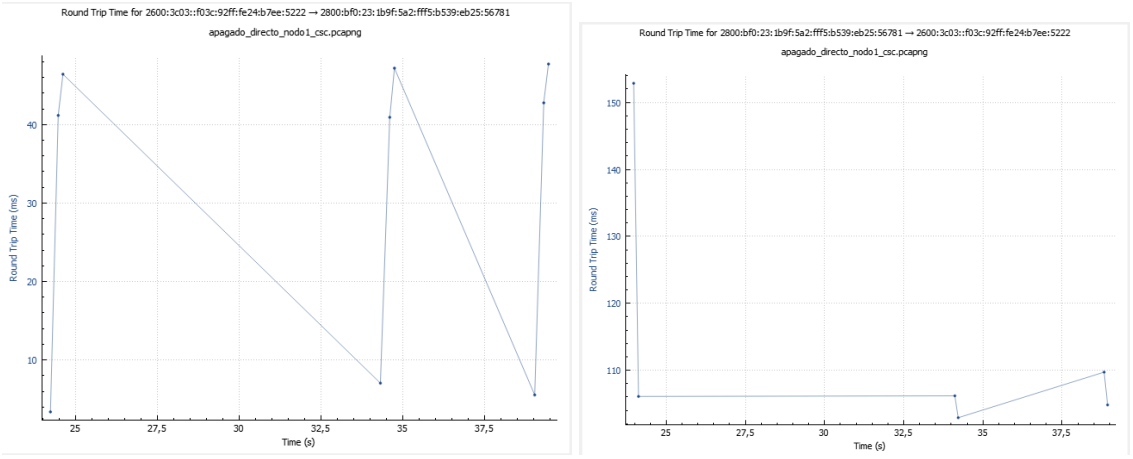
Round Trip Time

El tiempo de ida y vuelta (RTT) permite conocer el tiempo que tarda un paquete en enviarse al destino y recibir una respuesta (ACK) del mismo. Se puede medir en dos sentidos, primero del servidor hacia el cliente y segundo del cliente hacia el servidor.

RTT en comunicación de ruta directa cliente Pidgin (Windows)

Figura 148

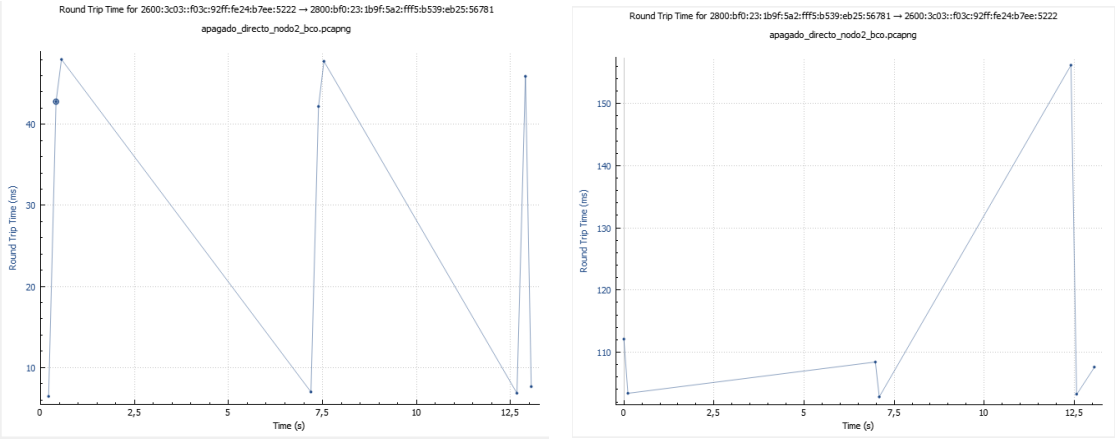
RTT al enviar la instrucción directa de apagado al nodo 1



A la izquierda de la figura 150 se muestra la interacción servidor – cliente de la petición enviada al nodo 1, se puede evidenciar que al enviar el mensaje de confirmación de apagado en los tres casos (cocina, comedor y sala) tiene un RTT de 46ms, este tiempo es bajo ya que se mide la respuesta desde la red local. En el lado derecho se muestra la interacción cliente - servidor en la que la petición de apagado se envía y tiene un RTT de 153ms como máximo en el primer mensaje y luego un promedio de 108ms en los otros dos, cabe recalcar que este tiempo es mayor ya que la comunicación se hace con un servidor cuya locación física se encuentra lejos, para el caso de este estudio en Estados Unidos.

Figura 149

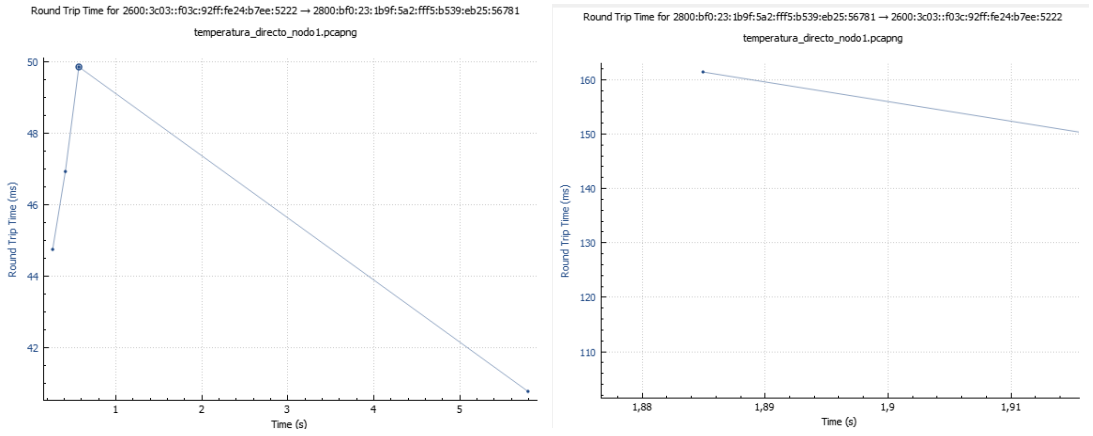
RTT al enviar la instrucción directa de apagado al nodo 2



A la izquierda de la figura 151 se muestra la interacción servidor – cliente de la petición enviada al nodo 2, se puede evidenciar que al enviar el mensaje de confirmación de apagado en los tres casos (baño, cuarto A y cuarto B) tiene un RTT de 48ms, este tiempo es bajo ya que se mide la respuesta desde la red local. En el lado derecho se muestra la interacción cliente - servidor en la que la petición de apagado se envía y tiene un RTT de 156ms máximos para el mensaje final y luego un promedio de 110ms en los otros.

Figura 150

RTT al enviar la instrucción directa de temperatura al nodo 1

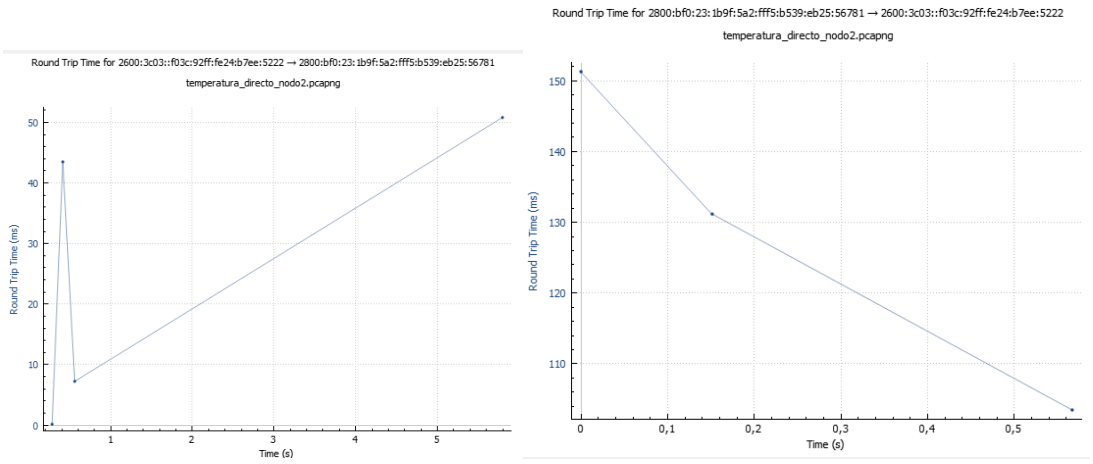


A la izquierda de la figura 152 se muestra la interacción servidor – cliente de la petición enviada al nodo 1, se puede evidenciar que al notificar del inicio del proceso se obtiene un RTT de 45ms mientras que

la respuesta con la temperatura tiene un RTT de 50ms. A la derecha de la figura se observa la comunicación cliente - servidor en la que la petición tiene un RTT de 161ms.

Figura 151

RTT al enviar la instrucción directa de temperatura al nodo 2

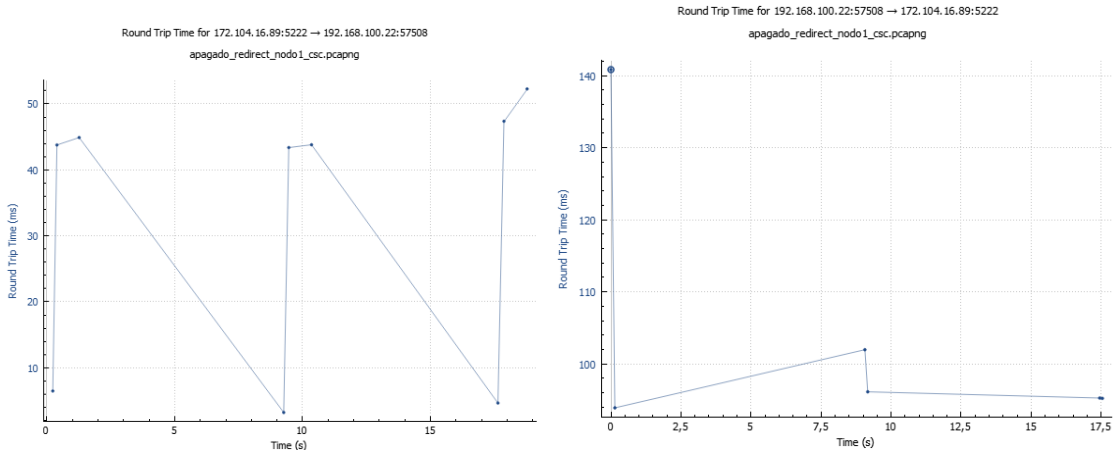


A la izquierda de la figura 153 se muestra la interacción servidor – cliente de la petición enviada al nodo 2, se puede evidenciar que al enviar la notificación del inicio del proceso se obtiene un RTT de 44ms mientras que el de la respuesta de la temperatura tiene un RTT de 51ms. A la derecha de la figura se indica la interacción cliente - servidor en la que la petición tiene un RTT de 151ms.

RTT en comunicación de ruta redireccionada cliente Pidgin (Windows)

Figura 152

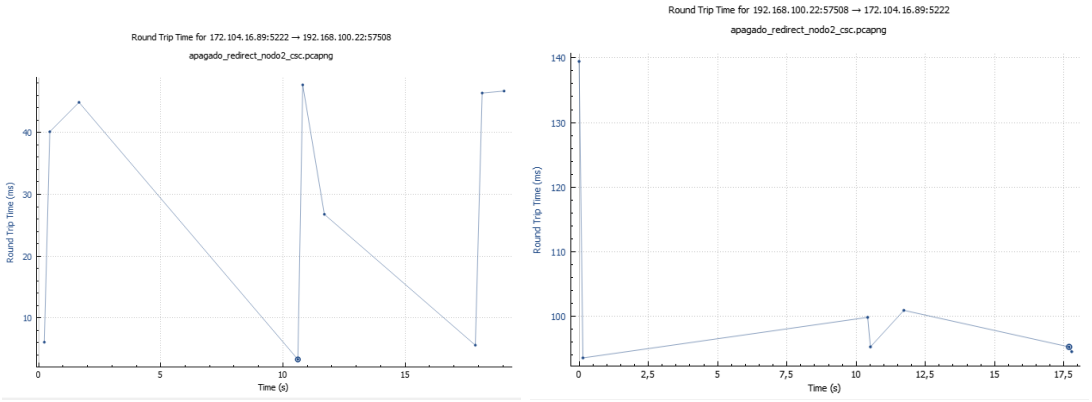
RTT al enviar la instrucción redireccionada de apagado de nodo 1 al nodo 2



A la izquierda de la figura 154 se muestra la interacción servidor – cliente de la petición enviada al nodo 2, se puede evidenciar que al enviar el mensaje de confirmación de apagado en los tres casos (cocina, comedor y sala) tiene un RTT de 46ms. En el lado derecho se muestra la interacción cliente - servidor en la que la petición de apagado se envía y tiene un RTT de 141ms como máximo en el primer mensaje y luego un promedio de 101ms en los otros dos, se puede identificar que la redirección no afecta en el RTT.

Figura 153

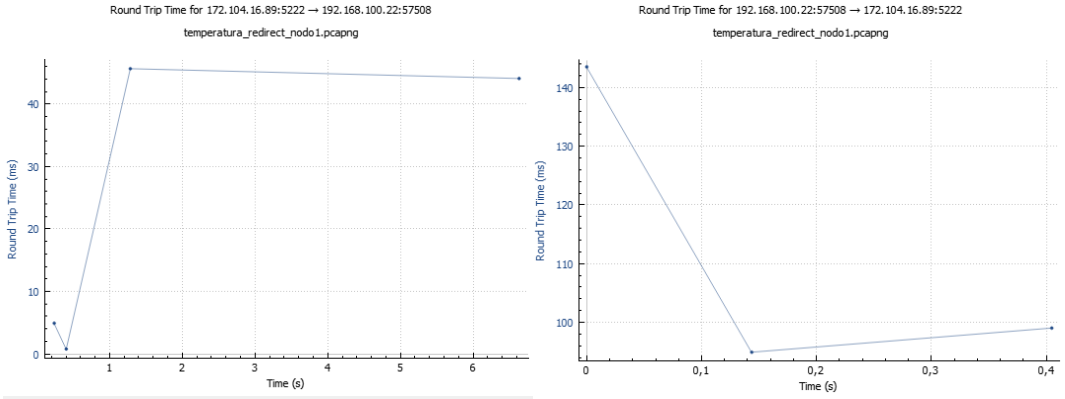
RTT al enviar la instrucción redireccionada de apagado de nodo 2 al nodo 1



A la izquierda de la figura 155 se muestra la interacción servidor – cliente de la petición enviada al nodo 1, se puede evidenciar que al enviar el mensaje de confirmación de apagado en los tres casos (baño, cuarto A y cuarto B) tiene un RTT de 46ms en promedio. En el lado derecho se muestra la interacción cliente - servidor en la que la petición de apagado se envía y tiene un RTT de 140ms como máximo en el primer mensaje y luego un promedio de 100ms en los otros dos.

Figura 154

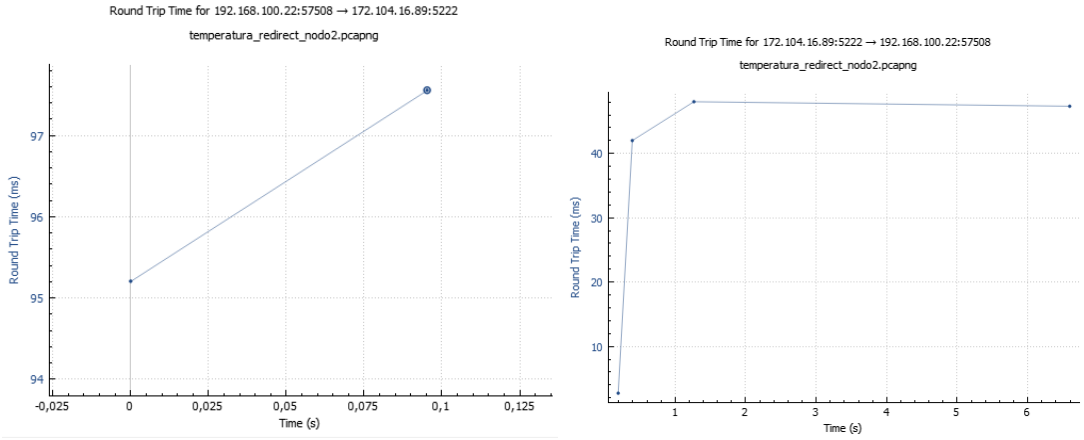
RTT al enviar la instrucción redireccionada de temperatura al nodo 2



Al lado izquierdo de la figura 156 se muestra la interacción servidor – cliente de la petición enviada al nodo 2 de la temperatura del sensor del nodo 1, se puede evidenciar que al enviar la notificación del inicio del proceso se obtiene un RTT de 46ms mientras que en la respuesta con la temperatura se obtiene un RTT de 45ms. A la derecha de la figura se observa la interacción en el sentido cliente - servidor en la que la petición obtiene un RTT de 144ms.

Figura 155

RTT al enviar la instrucción redireccionada de temperatura al nodo 1

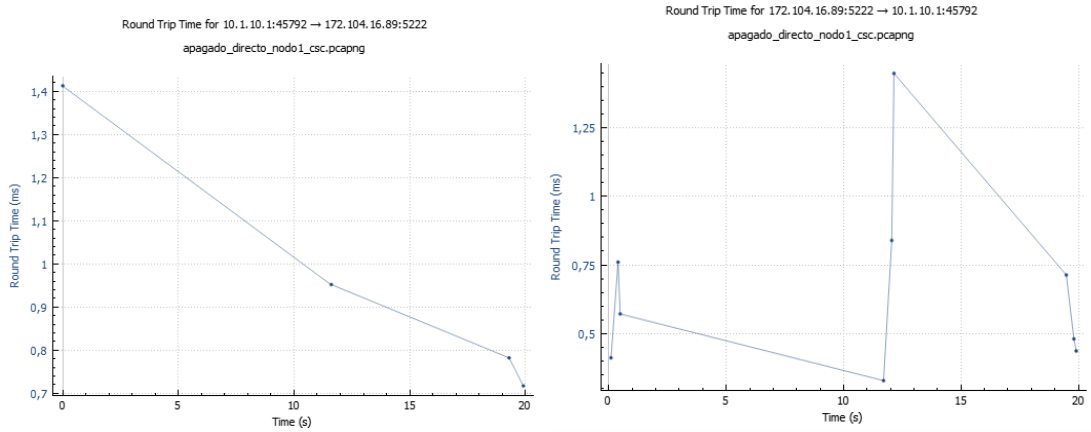


Al lado izquierdo de la figura 157 se muestra la interacción servidor – cliente de la petición enviada al nodo 1 de la temperatura del sensor del nodo 2, se puede evidenciar que al enviar el mensaje del inicio del proceso se obtiene un RTT de 95ms, pero la respuesta con la información de la temperatura obtiene un RTT de 97ms. Al lado derecho de la figura se evidencia la interacción cliente - servidor en la que la petición de temperatura consigue un RTT de 48ms.

RTT en comunicación de ruta directa cliente Xabber (Android)

Figura 156

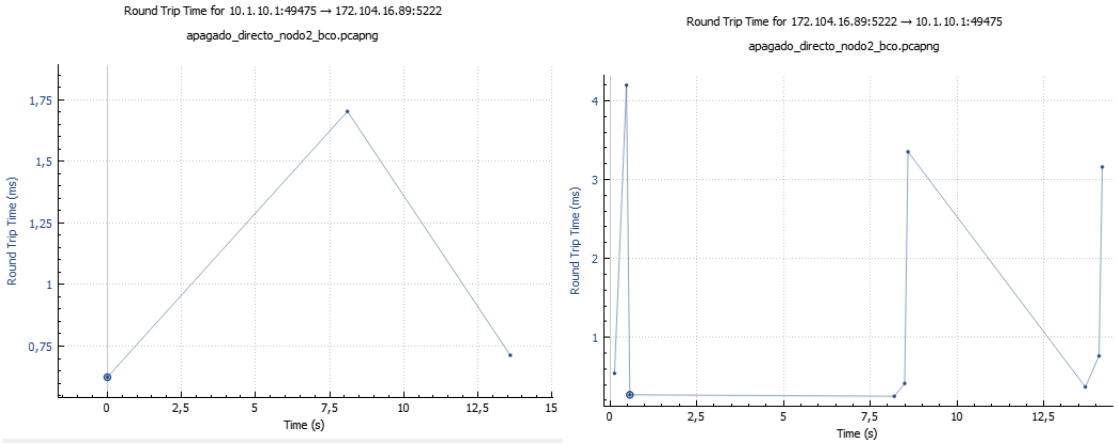
RTT al enviar la instrucción directa de apagado al nodo 1



Al lado derecho de la figura 158 se muestra la interacción servidor – cliente de la petición enviada al nodo 1, se puede evidenciar que al enviar el mensaje de confirmación de apagado en los tres casos (cocina, comedor y sala) tiene un RTT de 0.96ms en promedio, este tiempo es extremadamente bajo y presenta una relación proporcionalmente inversa al *Throughput*. En el lado izquierdo se muestra la interacción cliente - servidor en la que la petición de apagado se envía y tiene un RTT de 1.4ms como máximo en el primer mensaje y luego un promedio de 0.9ms en los otros dos.

Figura 157

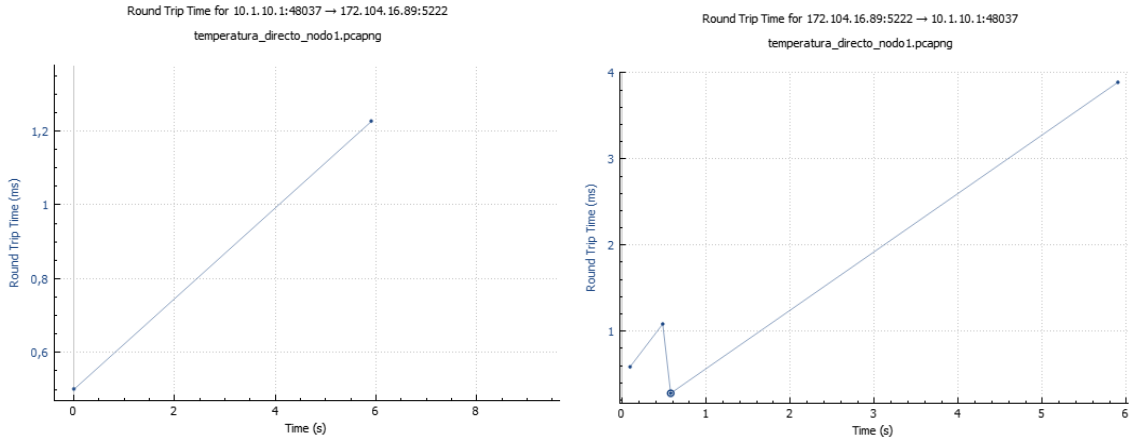
RTT al enviar la instrucción directa de apagado al nodo 2



Al lado derecho de la figura 159 se muestra la interacción servidor – cliente de la petición enviada al nodo 2, se puede evidenciar que al enviar el mensaje de confirmación de apagado en los tres casos (baño, cuarto A y cuarto B) tiene un RTT de 3.5ms en promedio. En el lado izquierdo se muestra la interacción cliente - servidor en la que la petición de apagado se envía y tiene un RTT de 1.70ms como máximo en el segundo mensaje y luego un promedio de 0.7ms en los otros dos.

Figura 158

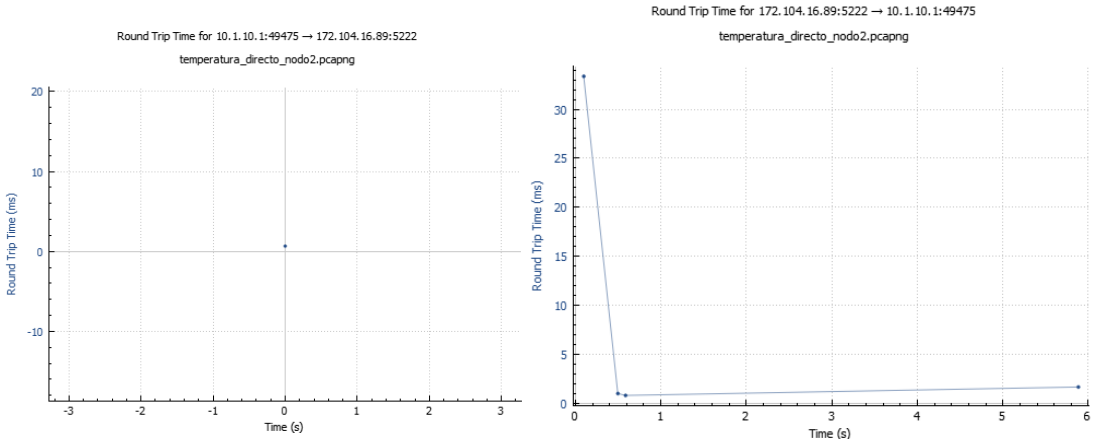
RTT al enviar la instrucción directa de temperatura al nodo 1



Al lado derecho de la figura 160 se muestra la interacción servidor – cliente de la petición enviada al nodo 1, se puede evidenciar que al enviar la notificación del inicio del proceso se obtiene un RTT de 1ms, pero la respuesta con la información de la temperatura consigue un RTT de 4ms. A la derecha de la figura se indica la interacción cliente - servidor donde la petición de temperatura obtiene un RTT de 1.2ms.

Figura 159

RTT al enviar la instrucción directa de temperatura al nodo 2

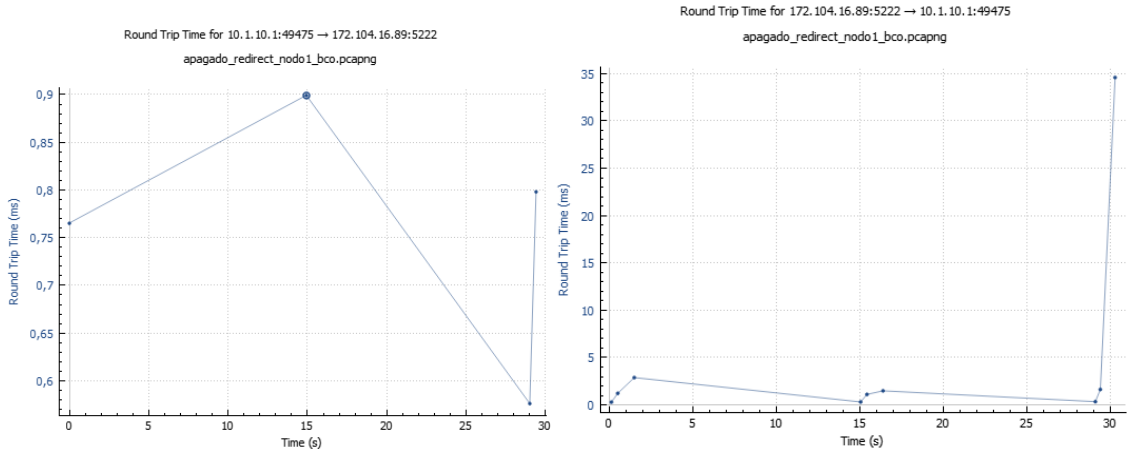


Al lado derecho de la figura 161 se muestra la interacción servidor – cliente de la petición enviada al nodo 2, se puede evidenciar que al notificar el incio del proceso se obtiene un RTT de 34ms, pero la respuesta con la temperatura consigue un RTT de 1ms, esta diferencia puede deberse a congestión de la red. Al lado derecho de la figura se indica la interacción sentido cliente - servidor donde la petición de temperatura obtiene un RTT de 1ms.

RTT en comunicación de ruta redireccionada cliente Xabber (Android)

Figura 160

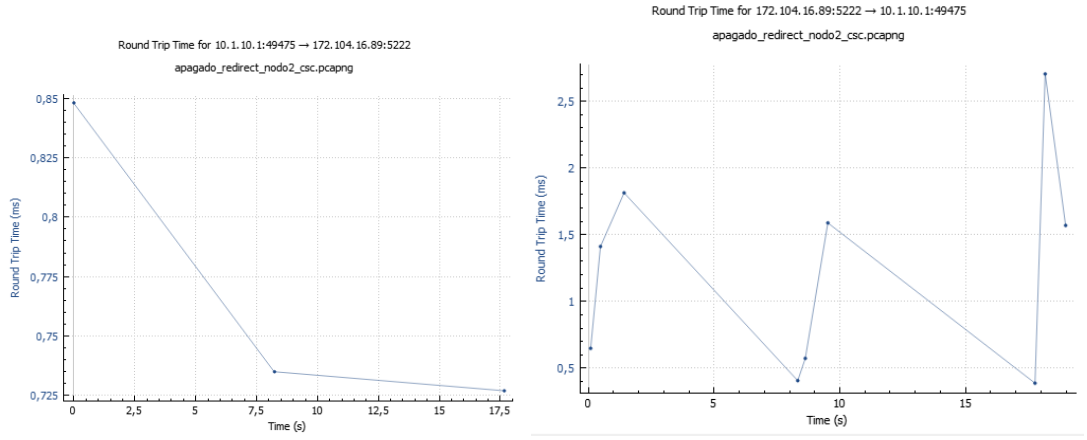
RTT al enviar la instrucción redireccionada de apagado del N1 al N2



Al lado derecho de la figura 162 se muestra la interacción servidor – cliente de la petición enviada al nodo 2, se puede evidenciar que al enviar el mensaje de confirmación de apagado en los tres casos (cocina, comedor y sala) tiene un RTT de 18.75ms de promedio. En el lado izquierdo se muestra la interacción cliente - servidor en la que la petición de apagado se envía y tiene un RTT de 0.9ms como máximo en el segundo mensaje y luego un promedio de 0.77ms en los otros dos.

Figura 161

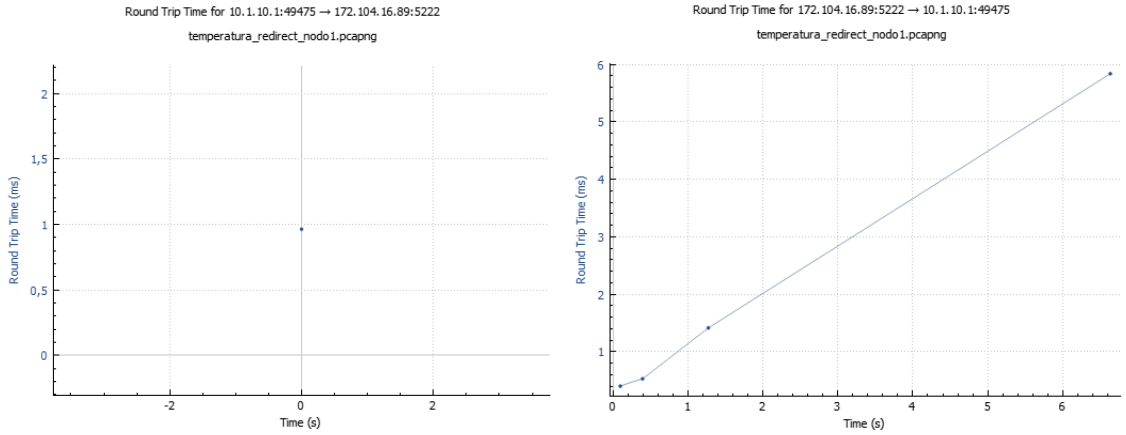
RTT al enviar la instrucción redireccionada de apagado del N2 al N1



Al lado derecho de la figura 163 se muestra la interacción servidor – cliente de la petición enviada al nodo 1, se puede evidenciar que al enviar el mensaje de confirmación de apagado en los tres casos (baño, cuarto A y cuarto B) tiene un RTT de 2.2ms de promedio. En el lado izquierdo se muestra la interacción cliente - servidor en la que la petición de apagado se envía y tiene un RTT de 0.85ms como máximo en el primer mensaje y luego un promedio de 0.73ms en los otros dos.

Figura 162

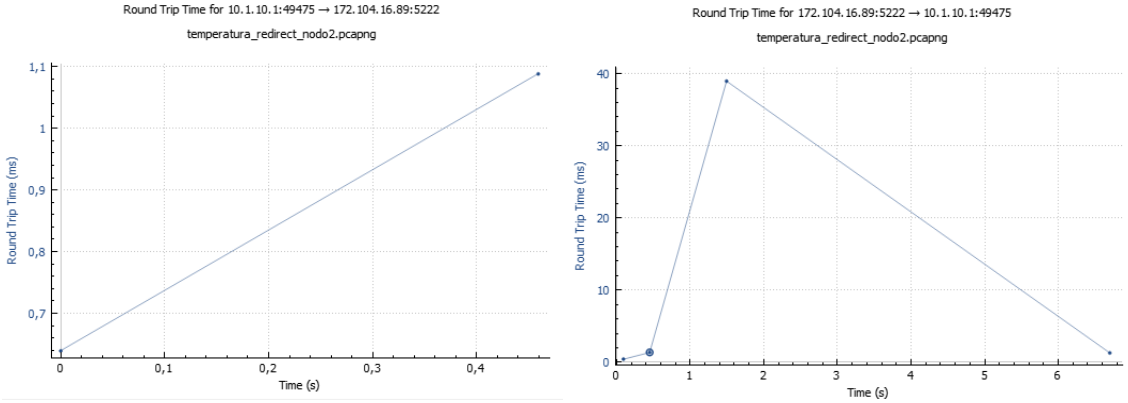
RTT al enviar la instrucción redireccionada de temperatura del N1 al N2



Al lado derecho de la figura 164 se muestra la interacción servidor – cliente de la petición enviada al nodo 2 de la temperatura del sensor del nodo 1, se puede evidenciar que al enviar la notificación del inicio del proceso obtiene un RTT de 1.5ms, pero la respuesta con la información de la temperatura consigue un RTT de 5.8ms. A la izquierda de la figura se observa la interacción sentido cliente - servidor en la que la petición de temperatura consigue un RTT de 0.9ms.

Figura 163

RTT al enviar la instrucción redireccionada de temperatura del nodo 2 al nodo 1



Al lado derecho de la figura 165 se muestra la interacción servidor – cliente de la petición enviada al nodo 1 de la temperatura del sensor del nodo 2, se puede evidenciar que al enviar la notificación del inicio del proceso se obtiene un RTT de 1ms, pero la respuesta con la información de la temperatura consigue un RTT de 39ms. A la izquierda de la figura se observa la interacción sentido cliente - servidor donde la petición de temperatura consigue un RTT de 1.09ms.

Capítulo V: Conclusiones y recomendaciones

Conclusiones

- Tras implementar un protocolo de mensajería instantánea con una red IoT se obtuvo de resultado que la comunicación funciona de manera totalmente funcional y eficiente de acuerdo a las pruebas realizadas. Además, esto es de igual manera independientemente del sistema operativo que tenga el dispositivo o nodo, en este trabajo de investigación se utilizaron tres distintos sistemas operativos que fueron Windows, Android y Raspbian, en todos ellos los resultados fueron prácticamente iguales con diferencias despreciables.
- Fue posible establecer un proceso de comunicación entre todos los nodos con un protocolo de mensajería XMPP a través de internet que permitió el acceso desde cualquier punto geográfico.
- Los análisis de la red realizados en sus aspectos de tiempo de respuesta, Throughput y Round Trip Time dieron resultados que permitieron establecer este tipo red con el protocolo XMPP como viables para su aplicación en situaciones reales, debido a que los resultados de dichas pruebas fueron positivas.
- El tiempo en que tardan en llegar las respuestas tras una comunicación redireccionada no representa un problema para la implementación de este tipo de red IoT, tomando en cuenta además la característica gratuita que tiene este protocolo de mensajería.

Recomendaciones

- Al conocer la diferencia apreciable de tiempo de respuesta entre una comunicación directa y una redireccionada se recomienda implementar un algoritmo de tal manera que sea el servidor quien envíe la petición al nodo adecuado aun cuando el usuario haya solicitado dicha información al nodo incorrecto.
- Si se va a realizar un proyecto a mediana escala es recomendable utilizar servidores privados o de pago para que no se creen cuellos de botella en el tráfico de información, además puede ser necesario añadir un servidor de procesamiento de datos.
- Si bien los datos obtenidos en el presente trabajo de investigación fueron positivos, su escala es muy pequeña, por ello se recomienda realizar una investigación a gran escala que utilice muchos más nodos que se conecten al mismo tiempo, de tal manera que se pueda obtener un resultado más preciso de la eficiencia de este tipo de red IoT mediante protocolos XMPP.

Referencias Bibliográficas

ASUS. (s.f.). *ASUS*. Obtenido de Tinker Board: https://www.asus.com/latin/Single-Board-Computer/Tinker-Board/

ByungHoon, B., & Kim, T. (2017). *Information Security Applications.* Korea: WISA.

Cárdenas, A. (6 de mayo de 2020). *Secmotic*. Obtenido de Hablemos de dispositivos IoT: Aplicaciones y plataformas: https://secmotic.com/dispositivos-iot/

Carles, J. (14 de marzo de 2020). *Instalar Raspbian con Raspberry Pi Imager de forma sencilla*. Obtenido de https://geekland.eu/instalar-raspbian-con-raspberry-pi-imager/

Cruz Vega, M., Oliete Vivas, P., Morales Rios, C., González Luis, C., Cendón Martín, B., & Hernández Seco, A. (2015). *Las tecnologías IOT dentro de la industria conectada: Internet of things.* Madrid: Escuela de Organización Industrial.

Del Valle, L. (s.f.). *ProgramacionFacil*. Obtenido de DS18B20 sensor de temperatura para líquidos con Arduino: https://programarfacil.com/blog/arduino-blog/ds18b20-sensor-temperatura-arduino/

Delgado, L. (21 de mayo de 2013). *Curso de privacidad y protección de comunicaciones digitales*. Obtenido de Comunicaciones seguras mediante mensajería instantánea: http://www.criptored.upm.es/crypt4you/temas/privacidad-proteccion/leccion3/leccion3.html#:~:text=El%20protocolo%20XMPP%20(Extensible%20Messaging,bajo%20el%20nombre%20de%20Jabber.

Domodesk. (s.f.). *Domodesk*. Obtenido de A FONDO: ¿QUÉ ES IOT (EL INTERNET DE LAS COSAS)?: https://www.domodesk.com/221-a-fondo-que-es-iot-el-internet-de-las-cosas.html

Elder, J. (2 de Septiembre de 2019). *avast blog*. Obtenido de Cómo Kevin Ashton nombró El Internet de las Cosas: https://blog.avast.com/es/kevin-ashton-named-the-internet-of-things

Fracttal. (s.f.). *Las 9 aplicaciones más importantes del Internet de las Cosas (IoT)*. Obtenido de https://www.fracttal.com/blog/2018/10/10/9-aplicaciones-importantes-iot

González, G. (29 de julio de 2014). *Hipertextual*. Obtenido de ¿Qué es el protocolo XMPP y dónde se usa?: https://hipertextual.com/archivo/2014/07/protocolo-xmpp/

Gracia, M. (s.f.). *Deloitte*. Obtenido de IoT - Internet Of Things: https://www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html

Hachman, M. (2002). *ARM Cores Climb Into 3G Territory*. Obtenido de http://www.extremetech.com/extreme/52180-arm-cores-climb-into-3g-territory

Juliá, S. (s.f.). *GADAE NETWEB*. Obtenido de Tipos de redes informáticas según su alcance: https://www.gadae.com/blog/tipos-de-redes-informaticas-segun-su-alcance/

Lederkremer, M. (2019). *Redes Informáticas.* RedUsers.

Li, P. (s.f.). *Stanza*. Obtenido de Stanza by Example: http://lbstanza.org/stanzabyexample.html

Microsoft. (s.f.). *Microsoft Azure*. Obtenido de Introducción a la seguridad de IoT: https://azure.microsoft.com/es-es/overview/internet-of-things-iot/iot-security-cybersecurity/

Mintz, M. (10 de marzo de 2003). *MSN Messenger Protocol*. Obtenido de http://www.hypothetic.org/docs/msn/general/overview.php

Moffitt, J. (2010). *Professional XMPP Programming with JavaScript and jQuery.* Indianapolis: Wiley Publishing.

Molina, J. (2014). *Servicios de Red e Internet (GRADO SUPERIOR).* RA-MA.

Nallapaneni, M. K. (junio de 2018). *ResearchGate*. Obtenido de IoT network types, data flow in IoT, data flow in IoT with blockchain technology : https://www.researchgate.net/figure/IoT-network-types-data-flow-in-IoT-data-flow-in-IoT-with-blockchain-technology_fig2_325661355

Petrikowski, N. (2014). *Getting to Know the Raspberry Pi.* The Rosen Publishing Group, Inc.

Quiñonez, O. (2019). *Internet de las Cosas (IoT).* Ibukku LLC.

Raspberri Pi. (s.f.). *Raspberri Pi*. Obtenido de https://www.raspberrypi.org/

RedHat. (s.f.). *RedHat*. Obtenido de Qué es IoT: https://www.redhat.com/es/topics/internet-of-things/what-is-iot

Rojas, S. (1 de agosto de 2020). *VSSistemas*. Obtenido de Arquitectura empresarial con capas para el Internet de las Cosas | 2ª parte: https://www.vs-sistemas.com/Blog/Actualidad/arquitectura-empresarial-con-capas-para-iot

Saint-Andre, P., Smith, K., & Tronçon, R. (2009). *XMPP: The Definitive Guide: Building Real-Time Applications with Jabber Technologies.* O'Reilly Media, Inc.

Sandoval, A. (25 de abril de 2020). *Think Big / Empresas*. Obtenido de Las redes de IoT, su evolución y beneficios: https://empresas.blogthinkbig.com/las-redes-utilizadas-en-el-iot/

Stone, R. (2019). *Raspberry Pi 4 Manual Completo: Una guía paso a paso a la nueva Raspberry Pi 4 y a la creación de proyectos innovadores.* Babelcube.

Tellis, P. (2010). *libyahoo2 - A C library for Yahoo! Messenger*. Obtenido de http://libyahoo2.sourceforge.net/

Thales. (s.f.). *Thales*. Obtenido de La seguridad del IoT – Conecte su negocio de manera segura con soluciones integradas y de seguridad en la nube: https://www.thalesgroup.com/es/countries/americas/latin-america/dis/iot/seguridad-en-iot

VIU. (21 de marzo de 2018). *Universidad Internacional de Valencia*. Obtenido de Plataforma IOT, conceptos generales y características importantes: https://www.universidadviu.com/plataforma-iot-conceptos-generales-caracteristicas-importantes/

Yúbal, F. (2018). *Llega la nueva Raspberry Pi 3 Model B+: mismo precio, pero más velocidad y WiFi de doble banda*. Obtenido de https://www.xataka.com/ordenadores/llega-la-nueva-raspberry-pi-3-model-b-mismo-precio-pero-mas-velocidad-y-wifi-de-doble-banda

Zhang, P. (2010). *Advanced Industrial Control Technology.* William Andrew.