APLICACIÓN DE MENSAJERÍA INSTANTÁNEA MODELO CLIENTE/SERVIDOR POR PROTOCOLO TCP PARA EL DEPARTAMENTO DE OPERACIONES DE LA EMPRESA ORANGE BUSINESS SERVICES COLOMBIA S.A.

CRISTIAN JULIAN AMAYA PRIETO

Profesor Tutor:
EYBERTH ROJAS

UNIVERSIDAD SANTO TOMÁS
FACULTAD DE INGENIERÍA ELECTRÓNICA
BOGOTÁ
2017

TABLA DE CONTENIDO

1.PROBLEMA 3
2. ANTECEDENTES 5
2.1. Chat empresarial o corporativo 5
2.2. Chat de servicio 6
2.3. Mensajería instantánea (Redes sociales) 6
2.4. Grupos de discusión o Foros 7
3. JUSTIFICACIÓN 8
4. MARCO HUMANISTA 10
5. OBJETIVOS 12
5.1.Objetivo general 12
5.2.Objetivos específicos 12
6. MARCO TEÓRICO 13
6.1.Modelo TCP/IP 13
6.2.Protocolo IPv4 – Capa de Internet 14
6.3.Protocolo TCP – Capa de Transporte 14
6.3.1.Formato de los datos en TCP 15
6.3.2.Puertos de red 16
6.4.Protocolo DHCP – Capa de aplicación 18
6.5. Wireshark 18
6.6.Visual Studio 2015 18
6.7. Lenguaje C# 19
7. DISEÑO METODOLÓGICO 20
8. EJECUCIÓN DEL PROYECTO 22
8.1. Resultados 36
9. CONCLUSIONES 38
10. BIBLIOGRAFIA 39

1. PROBLEMA

¿De qué manera es posible establecer un canal de comunicación alternativo a los medios convencionales, que ofrezca un servicio eficiente y de fácil acceso entre los clientes y el departamento de operaciones de la empresa Orange Business Services Colombia?

Cada día es más común ver cómo la opinión del cliente es un eje principal para el planteamiento de los objetivos de una empresa, la satisfacción de éste consumidor final desencadena tres beneficios muy importantes para su impacto en la sociedad, el primero de ellos es la lealtad, ya que un cliente complacido generalmente vuelva a adquirir un producto o servicio con la empresa; el segundo es el beneficio de la publicidad voz a voz (recomendación propia), la cual es generada por un cliente satisfecho que transmite sus experiencias a otros consumidores; y el tercero es participación en el mercado, gracias a que el cliente deja a un lado la competencia[1]. Para lograr este propósito es necesario estipular las bases entre las dos partes (empresa - cliente) al realizar una transacción de compra. Cuando un consumidor adquiere un producto o servicio con un proveedor, se establece un contrato bilateral para garantía del proceso. El cumplimiento de dichos parámetros existentes en este contrato demuestra el nivel de compromiso que posee la empresa, tanto para los clientes pequeños así como para los grandes consumidores.

Actualmente, las empresas se encaminan cada vez más en la satisfacción de las necesidades de los clientes, garantizar la transparencia en los procesos llevados para cada uno, brindar tranquilidad para el mismo y de igual manera para los directivos de la empresa. Para lograr dicho objetivo existen diferentes medios de comunicación con los cuales este puede obtener información o presentar alguna queja por un servicio adquirido con un proveedor. Uno de estos medios es el teléfono, el cual es un dispositivo eficiente y de pronta respuesta aunque a un costo elevado, por otro lado está el correo electrónico, el cual es el medio de comunicación más utilizado actualmente por las empresas ya que brinda almacenamiento y clasificación, además que permite una transferencia rápida de archivos al hacer uso de internet, la cual es una desventaja para el primer medio de comunicación mencionado [2]. Gracias al internet, las grandes compañías, principalmente las que ofrecen servicios de comunicación, hacen uso del servicio de mensajería instantánea (Chat) destinado a los clientes que requieren interactuar con su proveedor, quien actúa en este escenario como servidor.

El chat ha sido de gran utilidad gracias a que permite una eficiente e inmediata comunicación y con una pronta respuesta, sin mencionar el bajo costo que requiere para su uso. En el caso de la empresa Orange, el chat de servicio demandó los siguientes requerimientos para su implementación en la compañía: permitir las conexiones entrantes con el servidor, según el número de clientes que se encuentren en espera de algún pedido; tener una plena identificación de los usuarios conectados, quienes podrán tener comunicación directa con la persona que opera el chat en Orange sin importar el orden en que éstos se conecten; para ofrecer una mayor garantía de los productos entregados, la

empresa requiere de la opción de permitir el envío de imágenes desde el cliente hacia el operario del servidor, en casos en los que requiera presentar un soporte visual ante alguna queja o petición; adicionalmente, las conversaciones realizadas con el usuario deben ser almacenadas (exportadas) como soporte ante cualquier inconveniente que se manifieste.

De esta manera, el chat de servicio permite abrir un camino hacia una mejor interacción con los usuarios, con la posibilidad de dar respuestas y solucionar inquietudes en tiempo real, en relación con el estado de las compras que se realicen, así como también manifestar su opinión sobre la calidad de los productos adquiridos.

2. ANTECEDENTES

Las aplicaciones chat han llegado a ocupar un lugar importante en el mundo de las telecomunicaciones y sus usuarios, incluyen desde personas particulares hasta estamentos corporativos. Han facilitado la comunicación desde el año 1989 sobre la red de Internet, con la posibilidad de enlazar dispositivos en cualquier parte del mundo. En sus inicios fueron implementadas en universidades utilizando las redes locales y más adelante entre distintas instituciones a través de internet. Hoy en día, cualquier persona con un teléfono inteligente o con un computador puede acceder a una o varias cuentas desde diferentes aplicaciones Chat destinadas a diferentes objetivos.

Los servicios que brindan las aplicaciones de mensajería instantánea o Chat pueden ser clasificados según su uso [3], entre algunas de estas clasificaciones están:

2.1. Chat empresarial o corporativo:

Son las aplicaciones destinadas a su uso entre el personal de la empresa, su función es mantener una constante comunicación interna para aumentar la productividad y la facilidad en el manejo de la información dentro de la compañía. Entre algunos de estos programas están:

2.1.1. Skype For Business: utilizado actualmente en la empresa Orange® a nivel global, al igual que miles de compañías que tienen la necesidad de comunicarse entre su personal interno a través del mismo puerto usado por Skype. Proporcionan audio, vídeo y conferencias web en Internet, con la posibilidad de programar reuniones con antelación o dar inicio en cualquier momento. Ofrece la capacidad de conexión con un máximo de 250 personas a una misma reunión, a través de teléfonos inteligentes, tabletas, ordenadores, teléfonos y equipos de sala de reunión [4] y es compatible con las herramientas de Office utilizadas diariamente para el envío de archivos.

2.1.2. Chat Seguro: es un software de comunicación dirigido al entorno corporativo, ofrece lista de contactos divididos por Unidad, Sector y Favoritos. El historial de las conversaciones queda a disposición del administrador de la conversación. Tiene como característica principal el Análisis Inteligente del histórico de mensajes, el cual utiliza un algoritmo que analiza el histórico de mensajes que compara con un vocabulario de palabras que expresan positividad o negatividad, además asigna una puntuación a la conversación dependiendo de las expresiones tomadas como positivas o negativas.

2.2. **Chat de servicio**:

Son plataformas que generalmente se encuentran enlazadas en las mismas páginas Web de algunas empresas, principalmente en operadores móviles. Como su nombre lo indica, están destinados a servir a los clientes para atender dudas e inquietudes y por lo general solicitan información personal del consumidor para acceder al chat. Se pueden encontrar software en internet que ofrecen este servicio con un costo adicional, entre algunos de ellos están:

- LivePerson: es una empresa de tecnología estadounidense que desarrolla productos de mensajería en línea para clientes y visitantes en tiempo real a través de los sitios Web, móviles y redes sociales. El costo de este servicio va desde US$49 al mes para pequeñas comunidades, dependiendo del número de usuarios que se conecten a la aplicación [5].

- Volano: es un software para la plataforma Java creado en 1996 y es usado por las páginas de internet de empresas alrededor del mundo, permite conferencias, ventas en línea, salas de chat, entre otros servicios, el precio de la licencia es de US$499[6].

Entre las plataformas creadas por los operadores móviles se encuentran:

- Agente virtual – Movistar: es un chat en línea para clientes nuevos y antiguos de Movistar. Para ingresar a este servicio son necesarios los datos del cliente para verificación, también se requiere ingresar el tipo de información que el cliente requiere. Ofrece la opción de Videochat.

- Fandatichat – Etb: utilizado por la Empresa de Telecomunicaciones de Bogotá, permite contactar con un asesor para acceder a información de servicios de telefonía móvil y planes de hogar. Requiere únicamente de nombre, teléfono y correo del usuario para acceder al servicio.

Existen compañías enfocadas en otras ramas que igualmente ofrecen el servicio de chat para sus clientes, como es el caso de las aerolíneas, las cuales también atienden un alto número de inquietudes de pasajeros, esto con el fin de darle siempre al cliente un alto nivel de seguridad.

2.3. **Mensajería instantánea (Redes sociales):**

Son las aplicaciones destinadas a comunicar círculos sociales a nivel personal, aunque su uso también ha sido una opción a nivel laboral. Permiten crear grupos entre los usuarios, enviar notas de voz y archivos como imágenes o documentos.

Entre las principales aplicaciones de mensajería instantánea de mayor uso a nivel mundial se encuentran:

2.3.1. Whatsapp: creada en Estados Unidos, cuenta con 350 millones de usuarios activos mensuales, siendo la segunda aplicación más utilizada después de WeChat (China) [7]. Entre sus funcionalidades están el envío de fotos y videos, realizar llamadas sobre internet, compartir ubicación y crear difusiones para múltiples usuarios.

2.3.2. Facebook Messenger: aplicación desarrollada por la red social Facebook, permite la comunicación entre los usuarios de la red social, la cual supera los 1.000 millones de usuarios por mes [8]. Permite compartir archivos, imágenes, gifs, videos, la ubicación y el uso de stickers, también es posible crear grupos de discusión.

2.4. Grupos de discusión o Foros:

Son páginas Web que permiten al usuario entrar en un grupo donde discuten sobre algún tema en específico. El participante únicamente requiere crear un Nombre de Usuario o Nickname para ingresar a un canal de su preferencia, donde discute sobre una temática en particular.

Una aplicación muy utilizada es mIRC, la cual utiliza el protocolo IRC (Internet Relay Chat) en tiempo real y que permite debates entre dos o más personas. Se diferencia de la mensajería instantánea en que los usuarios no deben acceder a establecer la comunicación de antemano, de tal forma que todos los usuarios que se encuentran en un canal pueden comunicarse entre sí, aunque no hayan tenido ningún contacto anterior. Los usuarios del IRC utilizan una aplicación cliente para conectarse con un servidor, en el que funciona una aplicación IRCd (IRC daemon o servidor de IRC) que gestiona los canales.

Internet Relay Chat es un sistema de chat popularmente utilizado por millones de personas alrededor del mundo para interactuar con amigos y reunir grupos de gente con los mismos gustos. Es sencillo de usar y permite el fácil acceso a diferentes canales sobre temas en específico [9], los hay especializados en música y en libros, entre otros. Otra modalidad muy utilizada es la de los juegos, en el que se destacan los Cyberjuegos, ofreciendo cientos de canales en todos los servidores.

3. JUSTIFICACIÓN

De la misma manera que las aplicaciones Chat existentes han logrado unir personas a nivel personal y laboral, consiguiendo el objetivo de transmitir información relevante. En la empresa Orange Business Services Colombia S.A., la implementación de la aplicación de mensajería instantánea con el modelo de cliente/servidor, es un nuevo servicio, el cual permite abrir un mejor camino en la relación con los usuarios, apuntando a las mejores calificaciones sobre la percepción recibida por ellos de la imagen de la empresa. A través de encuestas realizadas periódicamente por la compañía se ha logrado conocer la opinión de los clientes en Colombia y en los países latinoamericanos, con el ideal de superar las expectativas y poner en alto el nombre de la empresa a nivel nacional e internacional. Es así, mediante la implementación de avanzadas herramientas de comunicación que se persuade el objetivo de ofrecer servicios de calidad y garantía para el consumidor final.

Siguiendo este propósito de mejorar la calidad en las comunicaciones en la empresa, se realizó un análisis de mercados sobre plataformas chat disponibles en internet, que ofrecen el servicio de alojamiento en una página web por un valor fijo mensual. En la siguiente tabla se comparan algunos de los chat de servicio, enfocados en el mismo objetivo:

NOMBRE	VALOR MENSUAL
Datta-chat	COP$35.000[10]
Zopim	US$25[11]
Olark	US$17 [12]
Live Person	US$49
ClickDesk	US$24 [13]

Tabla 1. Comparación plataformas Chat con valor fijo mensual.

En el departamento de operaciones de Orange Business Services Colombia, la aplicación Chat permitirá una fácil comunicación con el cliente de manera gratuita (discriminando costos de energía y uso de internet) en comparación con las plataformas expuestas en la Tabla 1; de esta manera, el usuario podrá solucionar sus inquietudes sobre el estado de sus entregas de manera pronta y con el respaldo de poder verificar todo tipo de información a través de los diferentes medios que se tienen a disposición en la empresa para el control de las ventas: manejo de inventario local, estado de la compra en la plataforma virtual privada de registro y control de equipos “Camaleón” y con la comunicación rápida con el equipo perteneciente al departamento de operaciones de la empresa, quienes tienen absoluto conocimiento de las ventas programadas y conocen los cronogramas para entrega de los equipos.

Dicha comunicación también es de vital importancia por la necesidad de garantizar un correcto proceso logístico para las entregas, ya que es común encontrar clientes cuyas

instalaciones requieren de notificación con antelación para el ingreso de carga a sus edificios.

Así mismo, tiene la posibilidad de informar cuando se presenten inconvenientes con los equipos entregados, en casos en los que sobren o falten componentes necesarios para su instalación; de igual manera, podrá manifestar su inconformidad cuando el equipo presente alguna inconsistencia en su aspecto físico, soportando la evidencia con imágenes que pueden enviar al personal a cargo en la empresa.

4. MARCO HUMANISTA

A lo largo de estos cinco años de estudios en la Universidad Santo Tomás ha sido posible aprender sobre las distintas y enormes ramas que compone la Ingeniería Electrónica. La adquisición conocimientos en relación con las dinámicas de la tecnología en el mundo actual permite abrir un amplio camino hacia el éxito académico y el profesionalismo. Para lograr este objetivo no basta únicamente con la fundamentación en el aprendizaje teórico y práctico de la electrónica, para ello es precisa la apropiación de aspectos como la responsabilidad, honestidad, solidaridad y ética laboral, pilares inculcados durante la instancia en la Alma Mater, atendiendo a su espíritu humanista de formación integral.

Ahora bien, en el contexto de la práctica laboral, se exalta la importancia de la capacidad intelectual ligada a las aptitudes humanas y sociales, dos aspectos vitales que poseen una alta demanda en las empresas enfocadas en el ritmo de globalización actual. En efecto, de manera progresiva se ha empezado a notar una preocupación en las empresas por favorecer el desarrollo humano de los trabajadores, evidenciado en el aumento de la implementación de departamentos de recursos humanos, enfocados a forjar componentes éticos, para ambientes laborales más sanos y claros en su responsabilidad social generando un impacto en el ámbito productivo más atento a la realidad social de la sociedad. Así mismo, se promueven capacitaciones enfocadas a la cualificación de los canales de comunicación destinados a mejorar, por un lado, las relaciones interpersonales en las empresas, y por otro lado, a la optimización de los procesos productivos.

La necesidad de mantener constante comunicación con los clientes ha llevado a implementar medios como el chat de servicio, una plataforma alternativa y de fácil acceso que permite una interacción más cercana entre el usuario y el proveedor del servicio, porque contribuye al aumento de la confianza entre las dos partes al establecer una comunicación instantánea. Es así como la implementación de un chat de servicio ofrece un mayor grado de estabilidad en las compañías, por ello, Orange Business Services tiene la necesidad de la aplicación de este medio en el departamento de operaciones.

En efecto, las aplicaciones Chat utilizadas en campos laborales, personales y comunitarios, han facilitado las comunicaciones entre personas y de esta manera han mejorado su calidad de vida. En las empresas, especialmente en el sector de las telecomunicaciones, han sido implementadas para la búsqueda de la satisfacción de las necesidades de los clientes, ya que permite escuchar sus sugerencias y poder obtener información sobre el estado de sus peticiones. De esta manera, el usuario obtiene toda la información necesaria y pertinente al servicio adquirido, este es un beneficio bilateral, por un lado, la empresa garantiza la transparencia en sus procesos llevando un estricto orden logístico, de la manera en que satisfacen los parámetros de entrega según los requerimientos del cliente (lugar, fecha, hora, procedimiento específico para la entrega,

etc.), los cuales, el mismo usuario puede manifestar dichos requerimientos a través de la aplicación Chat proporcionada; y por el otro lado, el cliente adquiere seguridad y un mayor grado de fidelidad con la compañía.

En el proceso de desarrollo de aplicaciones y programadores, los Ingenieros de sistemas, son el eje principal para su ejecución y buen funcionamiento, no obstante, en la actualidad, otros campos de la ingeniería han entrado a formar parte de esta dinámica (electrónica, telecomunicaciones, mecánica, entre otras), con el fin de expandir la implementación de estas innovaciones en diferentes campos del saber y proyectar nuevos usos. La transversalidad que existe entre dichos campos, ha sido una excelente oportunidad para acrecentar el conocimiento propio de la Ingeniería Electrónica y evidenciar su impacto en un sector de alta demanda como el de las comunicaciones. En nuestro país, éste ha ido evolucionando de manera paulatina, en comparación con otras naciones que son líderes en el desarrollo de software destinados al sector de las telecomunicaciones, de allí la necesidad apremiante de crear nuevas y eficientes herramientas, por el ingenio colombiano. De esta manera, resulta del todo relevante el aporte de la Ingeniería Electrónica para proyectar el avance tecnológico de nuestra sociedad, en un mundo altamente competitivo y globalizado.

5. OBJETIVOS

5.1. Objetivo general:

Desarrollar una aplicación cliente/servidor de mensajería instantánea con el lenguaje de programación C#, para la comunicación de los clientes con el departamento de operaciones de la empresa Orange Business Services Colombia S.A.

5.2. **Objetivos específicos**:

5.2.1. Proporcionar al cliente el archivo ejecutable para acceder a la aplicación Chat, sin demandar la necesidad de instalación de algún programa para su ejecución.

5.2.2. Soportar un máximo de 4 conexiones de manera simultánea, discriminando el orden en el que los clientes se enlacen con el servidor.

5.2.3. Disponer de un servidor configurado en modo Static con una única dirección IP para ofrecer fácil accesibilidad a los clientes.

5.2.4. Permitir el envío de imágenes desde el cliente hacia el servidor, para ofrecerle la posibilidad al usuario de presentar un soporte visual ante alguna inconformidad con el equipo recibido.

5.2.5. Almacenar las conversaciones en un archivo de lectura después de realizar la desconexión del servidor, con el fin de soportar la información discutida textualmente y evitar tergiversaciones.

5.2.6. Analizar los protocolos TCP e IPv4 durante la conexión con el servidor, transmisión y recepción de mensajes desde y hacia el cliente utilizando el software analizador de protocolos Wireshark, con el fin de inspeccionar las banderas del protocolo y garantizar la estabilidad en la comunicación.

6. MARCO TEÓRICO

Para la realización de la aplicación de mensajería instantánea mediante el modelo de comunicación TCP/IP son necesarios los siguientes conceptos relacionados con el tráfico de redes, configuraciones necesarias para el correcto funcionamiento e igualmente algunas definiciones del software y del lenguaje de programación a utilizar:

6.1. Modelo TCP/IP:

TCP/IP es un conjunto de protocolos de comunicación, sus siglas significan "Protocolo de control de transmisión/Protocolo de Internet". Proviene de los nombres de dos protocolos importantes del conjunto de protocolos, es decir, del protocolo TCP y del protocolo IP. Para poder aplicar el modelo TCP/IP en cualquier equipo, es decir, independientemente del sistema operativo, el sistema de protocolos TCP/IP se ha dividido en diversos módulos. Cada uno de éstos realiza una tarea específica. Además, estos módulos realizan sus tareas uno después del otro en un orden específico, es decir que existe un sistema estratificado [14]. Ésta es la razón por la cual se habla de modelo de capas.

El modelo TCP/IP es muy similar al modelo OSI (modelo de 7 capas) que fue desarrollado por la Organización Internacional para la Normalización (ISO) para estandarizar las comunicaciones entre equipos [15].La comparación entre las capas de ambos modelos se establece en la siguiente imagen:

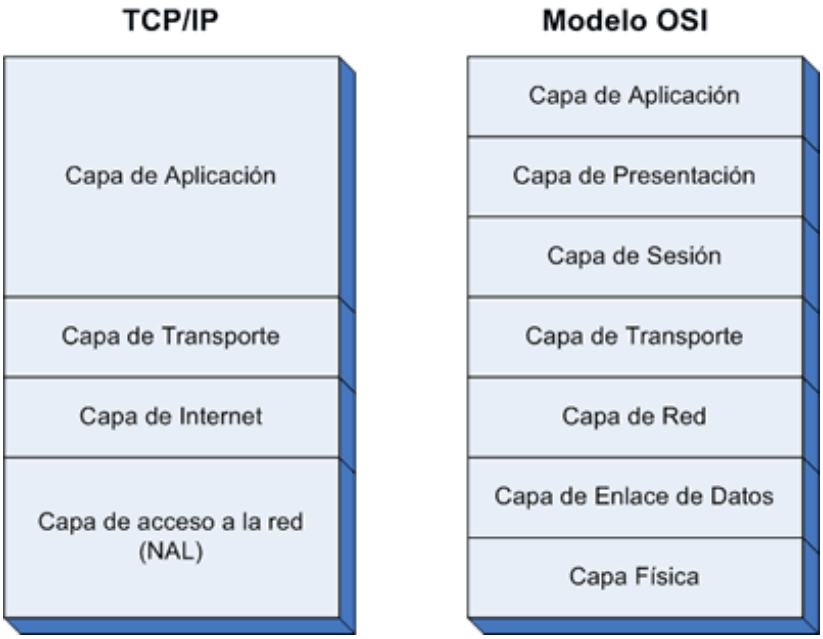

Figura 1. Comparación de las capas del modelo TCP/IP con el modelo OSI.
Fuente: *http://www.textoscientificos.com/redes/tcp-ip/comparacion-modelo-osi*

Como se observa en la figura 1, el modelo TCP/IP reduce el número de capas, pero cumpliendo las mismas funciones del modelo OSI, tanto en la transmisión como en la recepción. A continuación se describe cada una de las capas visualizadas en la tabla:

6.1.1. Capa de acceso a la red: determina la forma en la que los datos se deben enrutar, sea cual sea el tipo de red utilizado (LAN, WAN).

6.1.2. Capa de Internet: es responsable de proporcionar el paquete de datos (datagrama IP), mediante el protocolo de internet (IP).

6.1.3. Capa de transporte: brinda los datos de enrutamiento, junto con los mecanismos que permiten conocer el estado de la transmisión. Se utilizan los protocolos TCP y UDP.

6.1.4. Capa de aplicación: incorpora aplicaciones de red estándar (Telnet, SMTP, FTP, etc.).

6.2. Protocolo IPv4 – Capa de Internet:

IPv4 es la versión 4 del Protocolo de Internet (IP o Internet Protocol) y constituye la primera versión de IP que es implementada de forma extensiva. IPv4 es el principal protocolo utilizado en el Nivel de Red del Modelo TCP/IP para Internet. Este protocolo utiliza direcciones numéricas denominadas direcciones IP compuestas por cuatro números enteros (4 bytes) entre 0 y 255 [16]. Por ejemplo, 194.153.205.26 es una dirección IP en formato técnico.

Al utilizar direcciones de 32 bits (4 bytes), se limita el número de direcciones posibles a utilizar a 4,294,967,295 direcciones únicas. Sin embargo, muchas de estas están reservadas para propósitos especiales como redes privadas, Multidifusión (Multicast), etc., en consecuencia se reduce el número de direcciones IP que realmente se pueden utilizar. Por esta razón, se ha impulsado la creación de IPv6 (actualmente en desarrollo) como reemplazo eventual dentro de algunos años para IPv4.

6.3. Protocolo TCP – Capa de Transporte:

TCP (*Transmission Control Protocol*) es uno de los principales protocolos de la capa de transporte del modelo TCP/IP, el cual es orientado a la conexión, es decir, que permite que dos máquinas que están comunicadas controlen el estado de la transmisión. Las principales funciones del protocolo TCP son:

- Comunicar las aplicaciones en forma segura, gracias al sistema de acuse de recibo del protocolo TCP.

- Controlar la velocidad de los datos usando su capacidad para emitir mensajes de tamaño variable. Estos mensajes se llaman segmentos.

Durante una comunicación usando el protocolo TCP, las dos máquinas deben establecer una conexión, la máquina emisora (la que solicita la conexión) se llama cliente, y la máquina receptora se llama servidor, por esta razón se denomina un entorno Cliente-Servidor. Las máquinas de dicho entorno se comunican en modo en línea, es decir, que la comunicación se realiza en ambas direcciones.

6.3.1. Formato de los datos en TCP

Un segmento TCP está formado de la siguiente manera:

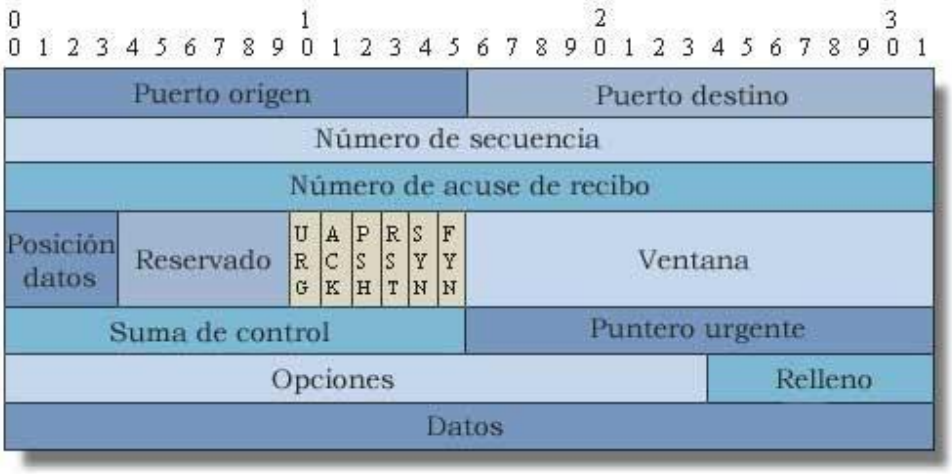

Figura 2. Representación del segmento TCP.
Fuente: *http://www.openredes.com/2011/03/31/formato-de-la-cabecera-de-segmentos-tcp/*

A continuación, se describen los diferentes campos o banderas del segmento TCP, visualizados en la figura 2:

- Puerto de origen (16 bits): Puerto relacionado con la aplicación en curso en la máquina origen.
- Puerto de destino (16 bits): Puerto relacionado con la aplicación en curso en la máquina destino.
- Número de secuencia (32 bits): Cuando el indicador SYN está fijado en 0, el número de secuencia es el de la primera palabra del segmento actual. Cuando SYN está fijado en 1, el número de secuencia es igual al número de secuencia inicial utilizado para sincronizar los números de secuencia (ISN).

- Número de acuse de recibo (32 bits): El número de acuse de recibo, también llamado número de descargo se relaciona con el número (secuencia) del último segmento esperado y no el número del último segmento recibido.
- Posición datos (4 bits): Permite ubicar el inicio de los datos en el paquete. Aquí, el margen es fundamental porque el campo opción es de tamaño variable.
- Reservado (6 bits): Un campo que actualmente no está en uso pero se proporciona para el uso futuro.
- Indicadores (6x1 bit): Los indicadores representan información adicional:
    - URG: Si este indicador está fijado en 1, el paquete se debe procesar en forma urgente.
    - ACK: Si este indicador está fijado en 1, el paquete es un acuse de recibo.
    - PSH (PUSH): Si este indicador está fijado en 1, el paquete opera de acuerdo con el método PUSH.
    - RST: Si este indicador está fijado en 1, se restablece la conexión.
    - SYN: El indicador SYN de TCP indica un pedido para establecer una conexión.
    - FIN: Si este indicador está fijado en 1, se interrumpe la conexión.
- Ventana (16 bits): Campo que permite saber la cantidad de bytes que el receptor desea recibir sin acuse de recibo.
- Suma de control (CRC): La suma de control se realiza tomando la suma del campo de datos del encabezado para poder verificar la integridad del encabezado.
- Puntero urgente (16 bits): Indica el número de secuencia después del cual la información se torna urgente.
- Opciones (tamaño variable): Diversas opciones.
- Relleno: Espacio restante después de que las opciones se rellenan con ceros para tener una longitud que sea múltiplo de 32 bits.

Los segmentos TCP se envían como datagramas de internet, la cabecera del protocolo de internet transporta varios campos de información, entre los que se incluyen las direcciones de los ‘host’ de origen y de destino, una cabecera de TCP sigue a la cabecera de internet, aportando información específica del protocolo de TCP. Esta división permite la existencia de otros protocolos de la capa de ‘host’ distintos de TCP.

6.3.2. Puertos de red

Es una interfaz para comunicarse con un programa a través de una red, son puertos lógicos asociados con un puerto físico o con un canal de comunicación y que proporciona un espacio para el almacenamiento temporal de la información que se va a transferir entre la localización de memoria de un ordenador y el canal de comunicación. La capa de transporte direcciona la información al puerto correspondiente en el proceso hacia la capa de aplicación [17].

Los puertos de red se dividen en tres rangos:

- Los puertos del 0 al 1023 son los puertos conocidos o reservados para procesos del sistema. Entre los puertos conocidos más utilizados según el tipo de aplicación se encuentran:

Puerto	Servicio o aplicación
21	FTP
23	Telnet
25	SMTP
53	Sistema de nombre de dominio
63	Whois
70	Gopher
79	Finger
80	HTTP
110	POP3
119	NNTP

Figura 3. Puertos de red conocidos.
Fuente: *http://es.ccm.net/contents/272-puerto-puertos-tcp-ip*

- Los puertos del 1024 al 49151 son los "puertos registrados", destinados a procesos ordinarios.
- Los puertos del 49152 al 65535 son puertos privados.

La combinación entre un puerto y una dirección IP se conoce como socket, el cual corresponde a un identificador de proceso en un entorno de red TCP/IP. De esta manera, la dirección IP sirve para identificar de manera única un equipo en la red mientras que el número de puerto especifica la aplicación a la que se dirigen los datos. Al cliente se le asigna un socket dinámicamente con el fin de permitir una comunicación fluida en los dos sentidos. Una vez establecida la conexión TCP, el intercambio entre el cliente y el servidor puede comenzar a través de ella [18].

"Los puertos lógicos son, al igual que los puertos físicos, necesarios para que los programas puedan comunicarse con el exterior. La diferencia es que se enlazan virtualmente en una conexión TCP con los programas, para tener una referencia, y que

los otros programas puedan conectarse a los nuestros y traspasar información" [19]. De esta manera, es posible enlazar ordenadores que se vinculen hacia aplicaciones similares, a través de la interfaz de puertos junto con las direcciones IP del cliente y del servidor.

6.4. Protocolo DHCP – Capa de aplicación:

El protocolo DHCP (Dynamic Host ConfigurationProtocol) es el protocolo de red encargado de la asignación de direcciones IP a cada una de las conexiones entrantes. De igual manera es posible realizar su configuración en modo Static o estática, una vez que una dirección IP estática se asigna un elemento de red, permanece allí hasta que se tome la decisión de cambiar la dirección por alguna razón. Así, por ejemplo, un elemento de red podría tener una dirección IP estática como 209.134.004.168. Hacer referencia a esa dirección en la red siempre apuntaría a que los elementos de red, al igual que su número de teléfono siempre se refieren a su teléfono. La ventaja de una dirección IP estática es principalmente la velocidad a la que se puede hacer referencia. Dado que el número no cambia nunca, y siempre se refiere al mismo elemento de red, se puede acceder inmediatamente sin procesamiento por encima [20].

6.5. Wireshark:

Anteriormente conocido como Ethereal, es un analizador de protocolos utilizado para realizar diagnósticos y solucionar problemas en redes de comunicaciones, para desarrollo de software y protocolos, y como una herramienta didáctica. Permite analizar paquetes de datos de redes Wifi y LAN [21]. Entre sus principales características están:

- Captura paquetes en vivo desde una interfaz de red.
- Muestra los paquetes con información detallada de los mismos.
- Abre y guarda paquetes capturados.
- Filtrado por protocolos de información de paquetes.
- Establecer estadísticas de las capturas.

Al realizar una captura, Wireshark analiza cada paquete con los protocolos existentes en cada transmisión, en el caso del protocolo TCP permite visualizar el contenido de cada campo en hexadecimal.

6.6. Visual Studio 2015

Visual Studio es la interfaz de desarrollo de Microsoft. Se compone de un conjunto de herramientas que permiten a los desarrolladores crear aplicaciones para las plataformas .NET. En la versión 2015 se compone de las ediciones express, que reúne todas las funcionalidades básicas para la creación de proyectos; profesional, dirigida a

desarrolladores profesionales; entreprise, para equipos profesionales que trabajan en proyectos; y community, que incluye todas las herramientas de desarrollo multiplataforma para aplicaciones móviles Windows, iOS y Android [22].

6.7. Lenguaje C#:

Es un lenguaje de programación diseñado para crear un amplio número de aplicaciones empresariales que se ejecutan en .NET Framework. “Supone una evolución de Microsoft C y Microsoft C++; es sencillo, moderno, proporciona seguridad de tipos y está orientado a objetos. El código creado mediante C# se compila como código administrado, lo cual significa que se beneficia de los servicios de Common Language Runtime. Estos servicios incluyen interoperabilidad entre lenguajes, recolección de elementos no utilizados, mejora de la seguridad y mayor compatibilidad entre versiones”[23]. Entre sus principales características se encuentran:

- Sencillez de uso: elimina muchos elementos añadidos por otros lenguajes lo que facilita su comprensión.
- Orientado a objetos: soporta todas las características de la programación orientada a objetos, como son la encapsulación, la herencia y el polimorfismo.
- Seguridad de tipos: incluye mecanismo de control de acceso a tipos de datos, lo que garantiza que no se produzcan errores difíciles de detectar.
- Instrucciones seguras: se imponen una serie de restricciones en el uso de instrucciones de control más comunes.

Cuando se realiza un programa con un lenguaje orientado a objetos, en este caso C#, significa que dicho programa está basado en la creación y uso de objetos, estos objetos pertenecen a una clase, que puede ser asimilado como nuevos tipos de datos[24]. Su esquema es bastante similar al lenguaje Java, siendo C# un lenguaje de programación avanzado, producto en conjunto de Java y los lenguajes C.

7. DISEÑO METODOLÓGICO

El desarrollo de las interfaces de los clientes y del servidor se realizan con el software Visual Studio 2015, a partir del lenguaje de programación C#. Desde el lado del cliente, la aplicación contará con una interfaz amigable con el usuario, de manera que pueda realizar la conexión con el servidor sin presentar inconveniente alguno. El cliente no tendrá la necesidad de instalar ningún programa ya que se le será suministrado el archivo ejecutable con el cual acceder fácilmente a la aplicación, exclusivamente al cliente que demande del servicio. Para el acceso a la aplicación, éste contará con la siguiente interfaz:

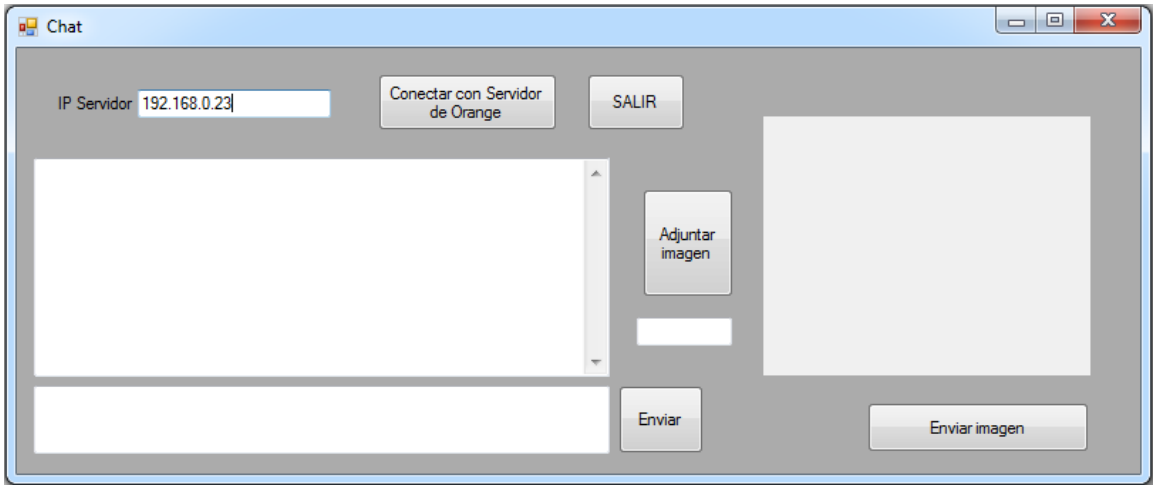

Figura 4. Interfaz del cliente.

Para el uso de la interfaz visualizada en la figura 4, el cliente ingresa la dirección IP del servidor suministrada con anterioridad, posteriormente realiza la conexión con el botón *Conectar con Servidor*. Al momento de obtener el enlace, el usuario podrá comunicarse directamente con el departamento de operaciones de la empresa Orange Business Services Colombia S.A., donde el operario del servidor atenderá la solicitud del usuario; seguidamente la información es procesada y soportada a través de los distintos medios que posee la empresa y según el requerimiento en específico, para finalmente dar una respuesta satisfactoria al cliente.

En los casos en que el usuario demande la necesidad de enviar un soporte visual al servidor, en relación con algún producto adquirido, la interfaz del cliente posee la funcionalidad de adjuntar una imagen a la conversación y enviarla al operario del servidor. Por último, en la parte superior el cliente posee la opción de salir cuando considere que su solicitud ha sido atendida.

Desde el lado del servidor, éste se encontrará disponible en los horarios laborales que maneja la empresa, tiempo en el cual permanecerá constantemente en estado pasivo, es decir que el servidor escucha y espera las conexiones TCP entrantes aleatoriamente, ya

que cuenta con ejecuciones en segundo plano, de manera que cada enlace se distribuye de forma independiente por puertos diferentes. La aplicación del cliente, considerada en estado abierta activa, realiza un pedido de conexión al servidor en el lugar donde la aplicación es abierta pasiva, la cual es aceptada por el servidor. La interfaz diseñada para el servidor se visualiza de la siguiente forma:

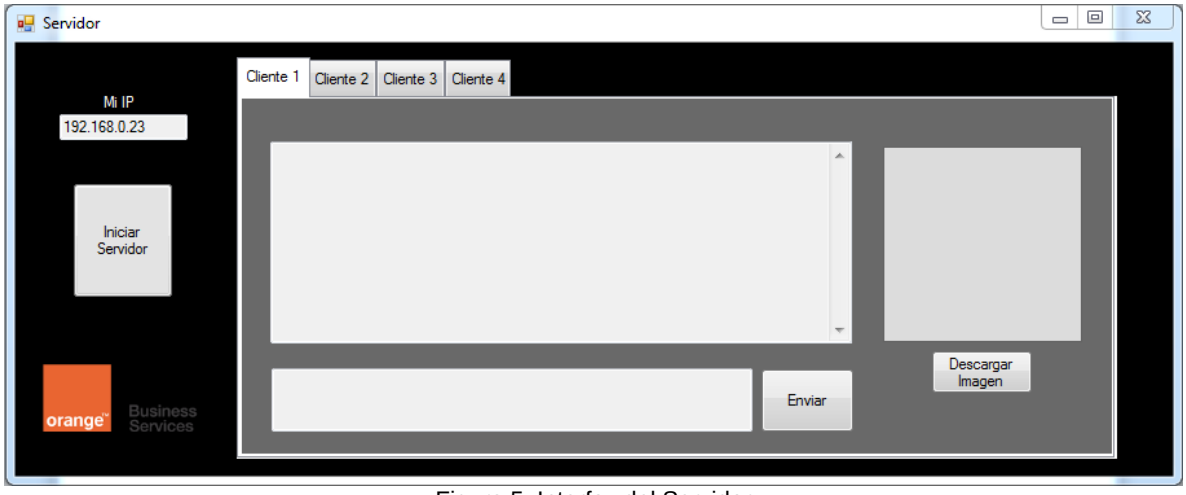

Figura 5. Interfaz del Servidor

Esta interfaz (figura 5) posee las funcionalidades de conexión y desconexión del servidor según los requerimientos del operario o coordinador, permite la interacción con los clientes por medio de pestañas lo cual ofrece una mayor facilidad de acceso para cada uno de ellos. La aplicación indicará la conexión de nuevos clientes a través de *Message Box*, quienes a su vez suministrarán la notificación de mensajes nuevos. Así mismo, el servidor recibe las imágenes que proporcione el cliente, las cuales podrá almacenar y realizar las acciones respectivas según la queja que éste manifieste. En complemento a la comunicación, se utiliza el analizador de protocolos Wireshark durante y después del enlace para monitoreo del mismo y garantía en su estabilidad. Al realizar la desconexión, las conversaciones pueden ser exportadas en un archivo para garantía y soporte de la comunicación establecida.

Si se desea implementar la aplicación cliente/servidor en la empresa, serán necesarios los permisos respectivos por los directivos para su disponibilidad, por ejemplo, a través de la página Web (en la nube), de manera que un cliente pueda acceder a este servicio de manera gratuita.

8. EJECUCIÓN DEL PROYECTO

El desarrollo de la aplicación Chat cliente/servidor para el departamento de operaciones de Orange fue llevado a cabo gracias al software Visual Studio 2015, el cual permite hacer uso de distintas librerías, entre ellas *System.Net.Sockets* para el uso del protocolo TCP, y que es fundamental para el uso de instrucciones en el manejo de Sockets y uso de la red. Posteriormente, se ingresa el código lógico para su buen funcionamiento con el lenguaje de programación C#, para obtener como resultado una aplicación funcional que garantice eficiencia para los usuarios involucrados.

Dentro del software Visual Studio 2015, inicialmente son ingresadas las librerías a utilizar, tanto en la aplicación del servidor como en la del cliente. En la siguiente imagen se muestran las librerías citadas:

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Threading.Tasks;
9   using System.Windows.Forms;
10  using System.Net;
11  using System.Net.Sockets;
12  using System.IO;
13  using System.Threading;
```

Figura 6. Librerías utilizadas para la aplicación Chat.

Algunas de las librerías visualizadas en la figura 6 se ejecutan en la ventana de programación de forma predeterminada, las cuales son necesarias para el uso de *Windows Form,* las últimas librerías citadas, señaladas con el paréntesis, son ingresadas como herramienta para el uso de sockets (IP, puerto).

A continuación, dentro del espacio de nombres se definen las variables que se van a utilizar, correspondientes a cada uno de los cuatro clientes que se van a comunicar con el servidor:

```
private TcpClient cliente1;
private TcpClient cliente2;
private TcpClient cliente3;
private TcpClient cliente4;
public StreamReader STR,STR2,STR3,STR4;
public StreamWriter STW,STW2,STW3,STW4;
public string msgrecibir,msgrecibir2,msgrecibir3,msgrecibir4;
public string msgenviar,msgenviar2,msgenviar3,msgenviar4;
```

Figura 7. Variables principales utilizadas en la aplicación.

En primer lugar se definen los nombres de los clientes para el uso de la red TCP, luego se definen las variables para la cadena de caracteres transmitida en los formatos de lectura *(STR)* y escritura *(STW)* para la recepción y transmisión respectivamente, y por último, se definen las variables para que éstas cadenas de caracteres se puedan visualizar dentro de los cuadros de texto *(TextBox).*

En la ventana de diseño de la interfaz, se ingresan las primeras funcionalidades para el servidor, las cuales corresponden a la identificación de la IP de éste, el botón para iniciar el servidor y una imagen corporativa de la aplicación. Se visualizan de la siguiente manera:

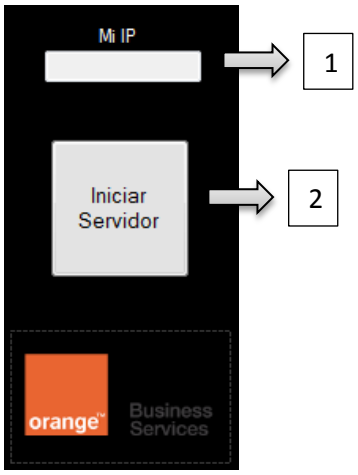

Figura 8. Funcionalidades iniciales del servidor.

En la parte superior de la figura 8 se creó el *TextBox,* indicado con el número 1 en la imagen, para visualizar la dirección IP correspondiente al ordenador en el que ésta siendo ejecutado el servidor, para esto se programó el siguiente código:

```
InitializeComponent();

IPAddress[] localIP = Dns.GetHostAddresses(Dns.GetHostName());
foreach (IPAddress DirIP in localIP)
{
    if (DirIP.AddressFamily == AddressFamily.InterNetwork)
    {
        textBox3.Text = DirIP.ToString();
    }
    }
```

Figura 9. Código para la visualización de la dirección IP del servidor.

Una vez se inicializa la ejecución del segmento de código de la figura 9, la aplicación obtiene las direcciones del host y realiza la comparación donde finalmente es convertida la dirección IP en texto para exponerla en éste *TextBox* (TextBox3).

El botón creado “Iniciar Servidor” (número 2 en la figura 8) posee el siguiente código para su ejecución:

```
private void button2_Click(object sender, EventArgs e)
{

    backgroundWorker5.RunWorkerAsync();
    backgroundWorker6.RunWorkerAsync();
    backgroundWorker7.RunWorkerAsync();
    backgroundWorker10.RunWorkerAsync();

}
```

Figura 10. Código para el botón 2 “Iniciar Servidor”.

Al ejecutar dicho botón *(button2)*, el segmento de código de la figura 10 da inicio a cuatro operaciones en segundo plano, correspondiente a los cuatro clientes a enlazarse. Los *BackgroundWorker* son el eje principal para el desarrollo de la aplicación, a través de ellos se realizan las solicitudes de conexión para cada cliente, al igual que la transmisión de mensajes. De esta manera se pueden ejecutar varias operaciones de manera simultánea sin ser afectadas entre sí, lo cual es una gran ventaja al permitir las conexiones de cada cliente sin importar su orden.

Una vez se ejecuten los *BackgroundWorker* de manera independiente, se describen las funciones para escuchar cada uno de los puertos utilizados para cada cliente, el código se visualiza de la siguiente manera:

```
private void backgroundWorker5_DoWork(object sender, DoWorkEventArgs e)
{
    TcpListener ConexA = new TcpListener(IPAddress.Any, 8080);
    ConexA.Start();
    cliente1 = ConexA.AcceptTcpClient();
    MessageBox.Show("CLIENTE 1 CONECTADO");
    STR = new StreamReader(cliente1.GetStream());
    STW = new StreamWriter(cliente1.GetStream());
    STW.AutoFlush = true;
    backgroundWorker1.RunWorkerAsync();
    backgroundWorker2.WorkerSupportsCancellation = true;
}
```

Figura 11. Código para los *backgroundWorker* de aceptación de Sockets desde el servidor.*8*

Para enlazar con el cliente, en la figura 11 se define inicialmente la variable para escuchar un Socket *(ConexA)*, el número del puerto está definido de manera independiente para cada uno de los usuarios. El servidor inicia el enlace, donde una vez el usuario realice la petición de conexión, el servidor acepta al cliente TCP, el cual es almacenado en la variable *cliente#.* Seguidamente se establecen las cadenas de caracteres de lectura y escritura para dicho cliente, por último, se inicializan los

backgroundWorker para el envío y recepción de mensajes, éste último, correspondiente al numeral 1, posee las siguientes líneas de código:

```
private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
{
    while (cliente1.Connected)
    {
        try
        {
            msgrecibir = STR.ReadLine();
            this.textBox2.Invoke(new MethodInvoker(delegate ()
            { textBox2.AppendText("Cliente: " + msgrecibir + "\n"); }));
            msgrecibir = "";
            MessageBox.Show("Nuevo mensaje de cliente 1");
                }
        catch(Exception x)
        {
            MessageBox.Show( "Cliente 1 desconectado\n" + x.Message.ToString());
            cliente1.Close();
            backgroundWorker5.RunWorkerAsync();
        }
    }
}
```

Figura 12. Código para los *backgroundWorker* de recepción de mensajes desde el servidor hacia el cliente.

En la figura 12 se describe el procedimiento realizado durante la transmisión de mensajes, el cual invoca la cadena de caracteres enviada desde la aplicación del cliente, y así poder mostrarla en el *TextBox2* de la ventana del servidor siempre y cuando el cliente se encuentre conectado; El servidor recibe una notificación del mensaje recibido y deja dicha variable vacía a la espera de un mensaje siguiente. Igualmente, notifica cuando la aplicación del cliente sea cerrada, deduciendo que éste se ha desconectado junto con el mensaje de error. En la ventana del servidor, se visualizan los mensajes recibidos de la siguiente manera:

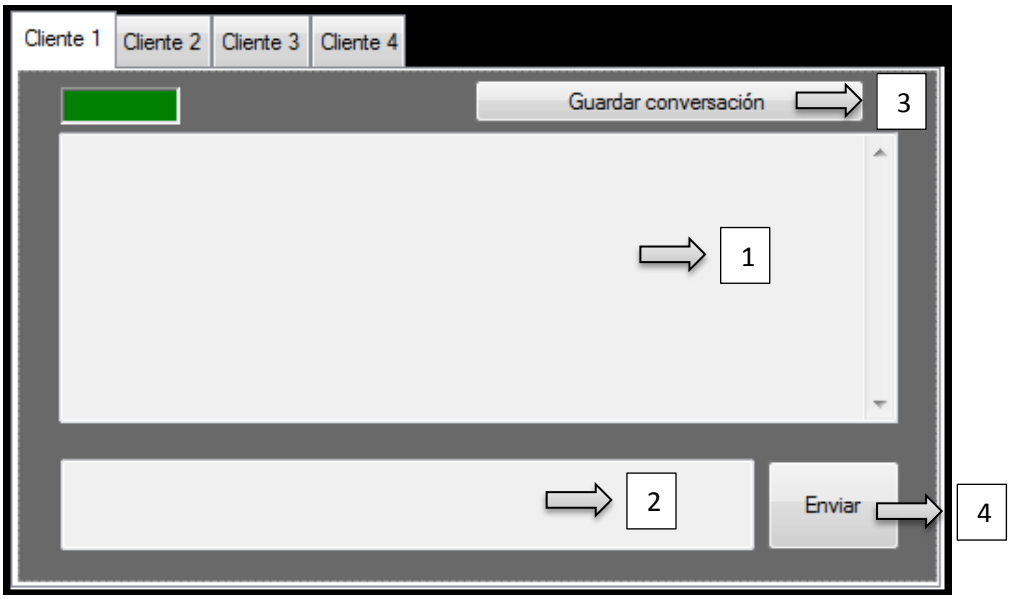

Figura 13. Ventana del servidor para el envío y recepción de mensajes.

En el cuadro superior, que corresponde al *Textbox2* indicado con el número 1 de la figura 13, el operario del servidor visualiza los mensajes, mientras que en la parte inferior se encuentra el cuadro para la escritura de mensajes (*Textbox1,* número 2 en la figura 13) y corresponde el *backgroundWorker2* anteriormente visualizado en la figura 11. En la parte superior derecha está la opción para guardar la conversación (número 3) y en la esquina inferior derecha se encuentra el botón "Enviar" para transmitir los mensajes a la aplicación del cliente (número 4). El código implementado para éste botón es el siguiente:

```
private void button1_Click(object sender, EventArgs e)
{
    if(textBox1.Text != "")
    {
        msgenviar = textBox1.Text;
        backgroundWorker2.RunWorkerAsync();
    }
    textBox1.Clear();
}
```

Figura 14. Código para el botón de envío de mensajes desde el servidor hacia el cliente.

Al presionar el segmento de código de la figura 14, correspondiente al botón "Enviar" *(button1)*, el mensaje que escriba el operario del servidor en el *textBox1* de la parte inferior (número 2 en la figura 13) es almacenado en la variable *msgenviar*, de manera que pueda ser manipulada para visualizarla en el cuadro superior, al igual que los mensajes que reciba. Para habilitar la opción de utilizar la tecla *Enter* del teclado para activar este botón en el envío de mensajes, se implementó el siguiente código:

```
private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == 13)
    {
        if (textBox1.Text != "")
        {
            msgenviar = textBox1.Text;
            backgroundWorker2.RunWorkerAsync();
            textBox1.Clear();
        }
    }
}
```

Figura 15. Código para habilitar el envío de mensajes utilizando la tecla *Enter*.

Como se observa en la figura 15, las instrucciones son similares, de manera que el operario del servidor tendrá las dos opciones para enviar el mensaje digitado en el *textBox1,* indicado con el número 2 de la ventana del servidor (figura 13).

Luego de que el mensaje enviado por el servidor sea almacenado en la variable *msgenviar*, se invoca a través del *backgroundWorker2* de la siguiente manera:

```
private void backgroundWorker2_DoWork(object sender, DoWorkEventArgs e)
{
    if(cliente1.Connected)
    {
        STW.WriteLine(msgenviar);
        this.textBox2.Invoke(new MethodInvoker(delegate ()
        { textBox2.AppendText("Yo: " + msgenviar + "\n"); }));
    }
    else
    {
        MessageBox.Show("Envio Fallido");
    }
    backgroundWorker2.CancelAsync();
}
```

Figura 16. Código para el *backgroundWorker* de envío de mensajes desde el servidor hacia el cliente.

De esta manera, el operario del servidor visualiza los mensajes (enviados y recibidos), en el cuadro superior de la ventana (número 1 en la figura 13). Adicionalmente, notifica a éste un error en la transmisión del mensaje en el caso en que el cliente no se encuentre aun conectado.

En la figura 13, se observa igualmente el botón para guardar las conversaciones realizadas (número 3), el cual posee el siguiente código para su ejecución:

```
private void button7_Click(object sender, EventArgs e)
{
    SaveFileDialog guardar = new SaveFileDialog();
    guardar.Filter = "Documento de texto|*.txt*";
    guardar.FileName = "Cliente "+nom1+" "+DateTime.Today;
    if (guardar.ShowDialog() == DialogResult.OK)
    {
        StreamWriter escribir = new StreamWriter(guardar.FileName);
        foreach (object line in textBox2.Lines)
        {
            escribir.WriteLine(line);
        }
        escribir.Close();
    }
}
```

Figura 17. Código para almacenar la conversación con el cliente en el servidor.

Para la ejecución del botón *(button7)* de la figura 17, se requiere de la creación de una variable de la clase *SaveFileDialog*, en la cual es almacenado el cuadro de texto de la conversación *textbox2*, al igual que se establece el nombre y la extensión del archivo que se va a crear, para finalmente ser exportado al ordenador del servidor. En primera instancia el archivo se guarda con el nombre del cliente junto con la fecha en la que se realiza esta operación, sin embargo, el operario del servidor puede modificar dicho nombre según algún contenido en específico.

En la parte derecha de la ventana del servidor, se encuentran las funcionalidades para la recepción y descarga de imágenes enviadas desde la aplicación del cliente. Estas opciones se visualizan de la siguiente manera:

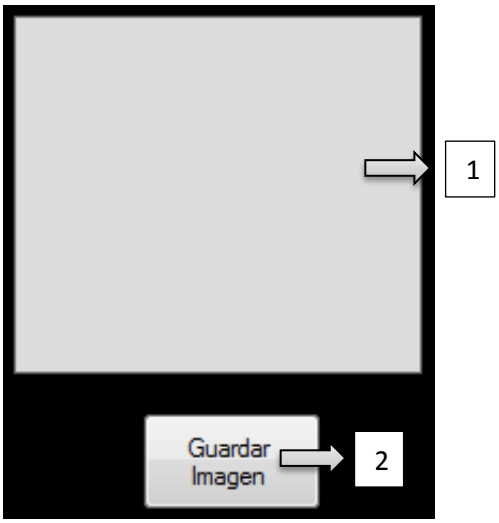

Figura 18.Ventana del servidor para la recepción de imágenes.

Para el correcto funcionamiento del complemento de la figura 18, se utilizaron igualmente algunas instrucciones ya explicadas en la transmisión de mensajes y en el almacenamiento de la conversación. El código para la recepción de imágenes y visualización en la ventana del servidor es el siguiente:

```
void ServerClient()
{
  try
{

        imgn  = new TcpListener(53100);
        imgn.Start();
        skt = imgn.AcceptSocket();
        cadn = new NetworkStream(skt);
        pictureBox2.Image = Image.FromStream(cadn);
        MessageBox.Show("Ha recibido un archivo de imágen.");

        imgn.Stop();
    if (skt.Connected == true)
          {
         while (true)
                {
                ServerClient();
                }
          }
}
catch(Exception ex) { }
}
```

Figura 19. Código para la recepción de imágenes en el servidor.

Similar al proceso para conexión con el cliente, en la recepción de imágenes se establece la variable que escucha un número de puerto para este procedimiento de la figura 19; luego de que existe el enlace se define una variable de la clase *NetworkStream*, de manera que la imagen se pueda visualizar en un cuadro de imagen o *PictureBox2,* indicado con el número 1 en la figura 18. El operario del servidor recibe la notificación del archivo recibido. Para guardar la imagen recibida se usó el siguiente código para el botón “Guardar imagen” (número 2 en la figura 18):

```
private void button4_Click(object sender, EventArgs e)
{
    saveFileDialog1.Filter = "Imagen (*.jpg)|*.jpg";
     if(saveFileDialog1.ShowDialog()==DialogResult.OK)
    {
      string desc = saveFileDialog1.FileName;
    pictureBox2.Image.Save(desc);
    }
}
```

Figura 20. Código para guardar la imagen recibida en el servidor.

En el segmento de código del botón *(button4)* de la figura 20, el operario del servidor ingresa el nombre con el que va a quedar guardada la imagen en el ordenador y es almacenada en formato jpg.

Desde la aplicación del cliente, las instrucciones programadas son bastante similares a las utilizadas en el servidor. La ventana principal diseñada se visualiza de la siguiente manera:

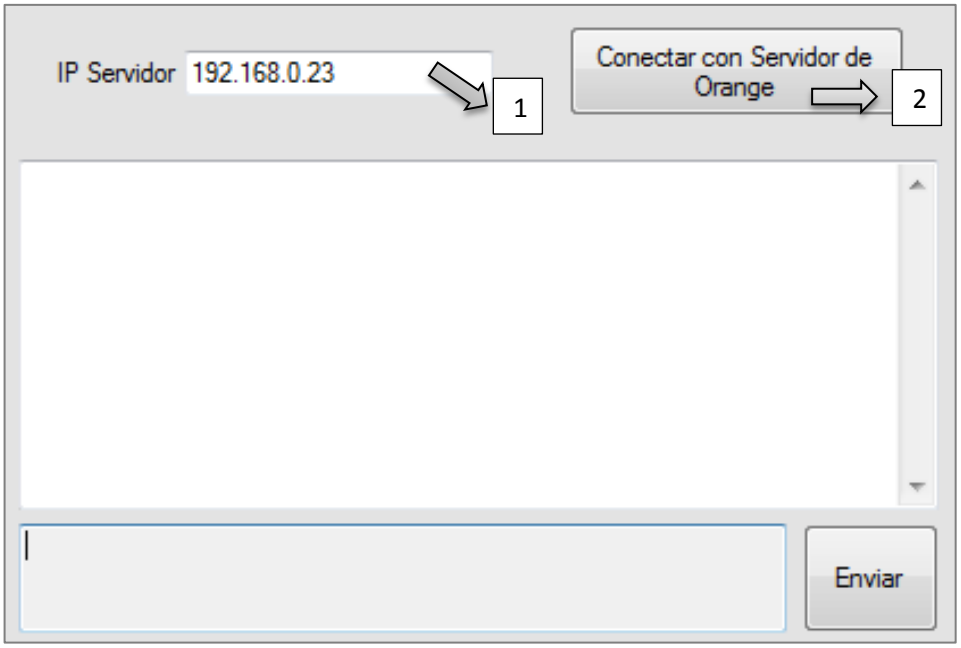

Figura 21. Ventana principal para el envío de mensajes en la aplicación cliente.

Como se observa en la figura 21, en el cuadro de texto de la IP del servidor (número 1) ya se encuentra ajustada la dirección real de éste, sin embargo, es posible modificarlo en el caso de que exista un cambio esporádico.

En la esquina superior derecha de la aplicación cliente (figura 21) se encuentra el botón para conectar con el servidor (número 2), el cual posee el siguiente código:

```
private void button2_Click(object sender, EventArgs e)
{
    servidor = new TcpClient();
    IPEndPoint IP_End = new IPEndPoint(IPAddress.Parse(textBox3.Text), 8080);

    try
    {
      servidor.Connect(IP_End);
    if (servidor.Connected)
    {
      textBox2.AppendText("Conectado al Servidor" + "\n");
            textBox1.ReadOnly = false;
            STR = new StreamReader(servidor.GetStream());
            STW = new StreamWriter(servidor.GetStream());
            STW.AutoFlush = true;

            backgroundWorker1.RunWorkerAsync();
            backgroundWorker2.WorkerSupportsCancellation = true;
        }
    }
    catch (Exception x)
    {
        MessageBox.Show("Servidor Desconectado\n"+x.Message.ToString());
    }
}
```

Figura 22. Código para botón de conexión con el servidor.

Al inicio del código de la figura 22 para el botón *(button2)* de la aplicación del cliente, se define el socket del servidor con el que se va a conectar *(IP_End)*, el cual se compone de la dirección IP mencionada anteriormente y el puerto predeterminado para ese cliente.

Posteriormente de que la conexión sea exitosa, el usuario de la aplicación cliente recibe la notificación, de manera que desde ese momento puede comunicarse directamente con la empresa. Por otro lado, si el servidor no ha dado iniciación, el cliente recibe una notificación de que el servidor se encuentra desconectado junto con el mensaje de error.

Para los cuadros de texto de envío y recepción de mensajes, visualizados en la figura 21, las instrucciones en el código de programación son iguales que las explicadas en el servidor. Así mismo, el usuario de la aplicación cliente recibe la notificación de desconexión del servidor, luego de que se haya establecido una comunicación.

En cuanto a la instrucción que posee el cliente para el envío de imágenes hacia el servidor, la aplicación proporciona la siguiente interfaz:

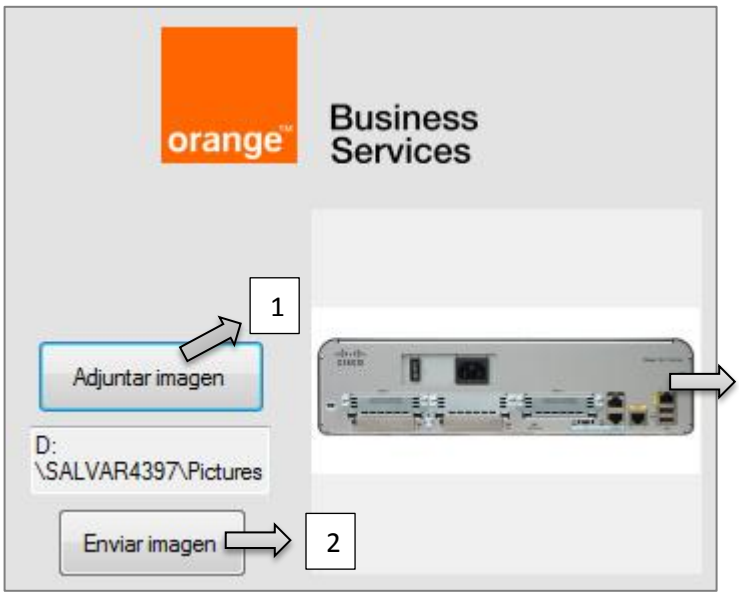

Figura 23. Interfaz de la aplicación cliente para el envío de imágenes.

En la anterior imagen se observa la ventana de fácil acceso que posee el usuario para esta instrucción. En la parte izquierda se encuentra el botón para adjuntar la imagen a enviar (número 1), junto con el botón de enviar (número 2), a la derecha está el *pictureBox* para visualizar dicha imagen (número 3). El botón "Adjuntar imagen" posee el siguiente código para su ejecución:

```
private void adjuntar_Click(object sender, EventArgs e)
{
    openFileDialog1.ShowDialog();
    string adj = openFileDialog1.FileName;
    pictureBox1.Image = Image.FromFile(adj);
    text_path.Text = adj;
}
```

Figura 24. Código para botón "Adjuntar imagen" en la aplicación cliente.

Con éste botón (*adjuntar)* explicado en la figura 24, se abre una nueva ventana con la cual acceder al archivo guardado en el ordenador, posteriormente se visualiza la imagen a la derecha ajustada al *picturebox1*, al igual que la dirección de la ubicación en el *textBox (text_path)* ubicado al inferior del botón.

Para el botón "Enviar imagen", indicado con el número 2 en la figura 23, se implementa el siguiente segmento de código:

```
private void button_imgn_Click(object sender, EventArgs e)
{
    try
    {
        mem = new MemoryStream();
        pictureBox1.Image.Save(mem, pictureBox1.Image.RawFormat);
        byte[] buffer = mem.GetBuffer();
        mem.Close();
        cliente1 = new TcpClient(textBox3.Text, 8080);
        cadn = cliente1.GetStream();
        br = new BinaryWriter(cadn);
        br.Write(buffer);
        br.Close();
        cadn.Close();
        cliente1.Close();
        MessageBox.Show("Imágen enviada con éxito");
    }
    catch (Exception) { MessageBox.Show("Falló envío de imágen. Servidor desconectado"); }
}
```

Figura 25. Código de botón para enviar imagen en la aplicación cliente.

En primer lugar, con éste botón *(button_imgn)* de la figura 25, se crea una variable de la clase *MemoryStream* con la cual obtener la matriz de bytes. Seguidamente se define el socket "cliente", quien, en este contexto es el servidor, con la dirección IP del mismo (*textBox3*) y el mismo puerto establecido en la aplicación del servidor. Por último, se escribe la matriz de bytes sobre el *NetworkStream* para el envío y recepción de datos establecido. El usuario recibe la notificación de que su imagen fue enviada, por otro lado, si el servidor no se encuentra disponible no se logra realizar la operación, informando al usuario que el servidor esta desconectado.

Desde el analizador de protocolos Wireshark, se inspeccionan cada una de las etapas que interfieren en el proceso de conexión y enlace entre el servidor y el cliente, en la ventana principal se selecciona la red con la que se va a trabajar:

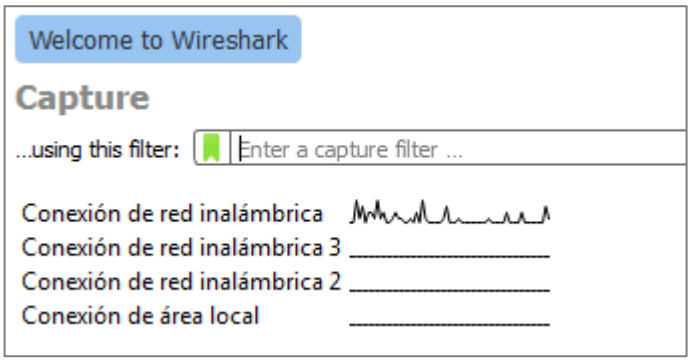

Figura 26. Ventana principal del analizador de protocolos Wireshark.

Cuando se expande la información de los paquetes transmitidos sobre la red seleccionada, se aplica un filtro en la parte superior para visualizar únicamente el protocolo TCP, sobre el cual se realiza la transmisión en el chat. Posteriormente, se inicia

el servidor, abriendo los puertos a las conexiones entrantes con los diferentes clientes y de esta manera realizar el pedido de conexión desde la aplicación cliente. En este proceso, el analizador de protocolos captura la siguiente información:

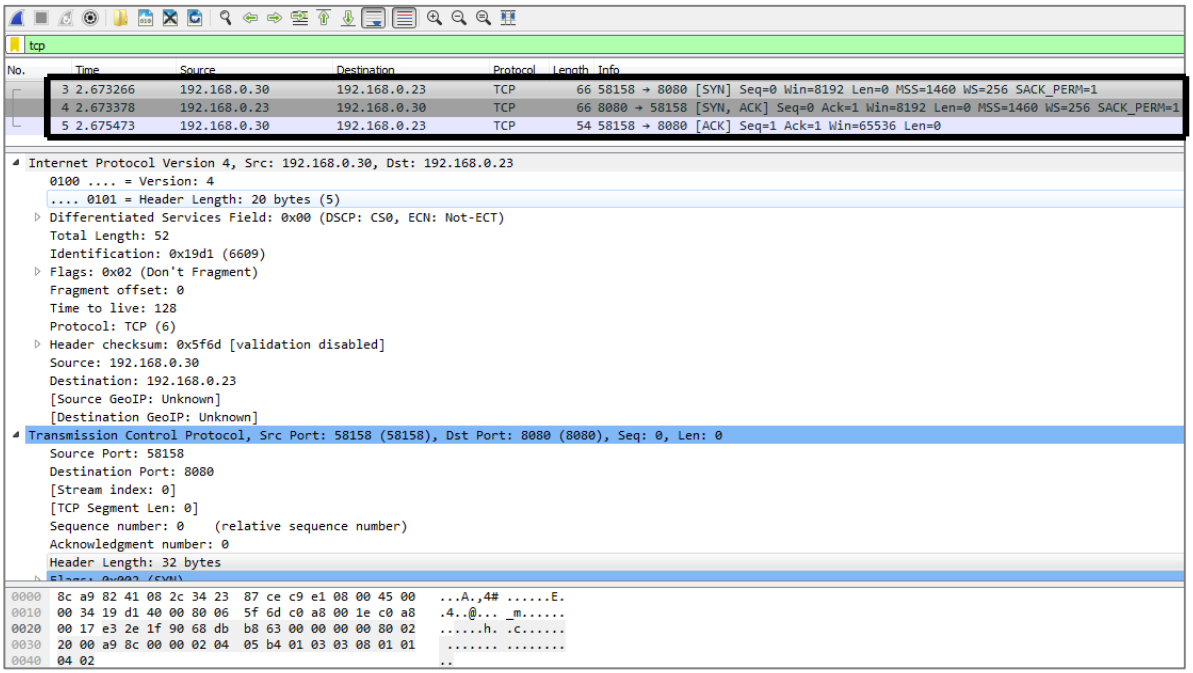

Figura 27. Captura de solicitud de conexión de la aplicación cliente al servidor con el analizador de protocolos Wireshark.

En el recuadro negro de la parte superior de la figura 27 se visualizan tres paquetes detectados en este proceso, al seleccionar el primero de ellos se observan los protocolos IPv4 y TCP existentes en el instante de petición de conexión del cliente. Al desplegar el menú de opciones en el protocolo de internet, se detallan las características del mismo IPv4, lo que corresponde al datagrama. Dentro de estas especificaciones se observa la versión del protocolo (*versión 4*) y la dirección IP de origen de la petición (*192.168.0.30*), es decir, el cliente. Al seleccionar sobre esta dirección se visualiza en la parte inferior su valor en hexadecimal en una cadena de 32 bits, de igual manera para cada especificación dentro de los protocolos.

Seguidamente, se despliega el protocolo de control de transmisión (TCP), donde se encuentra el puerto de destino (*8080*), que corresponde al valor asignado para dicho cliente. De igual manera se observa la bandera de acuse de recibo (*Acknowledgementnumber – ACK*) con un valor de cero, por lo cual, en el siguiente hilo con origen servidor y destino cliente, este valor es igual a 1, indicando que la petición ha sido recibida y aceptada.

En segunda medida, se analiza la transmisión de envío y recepción de mensajes, para esto se envía un mensaje desde la aplicación cliente hacia el servidor e inspeccionamos con Wireshark:

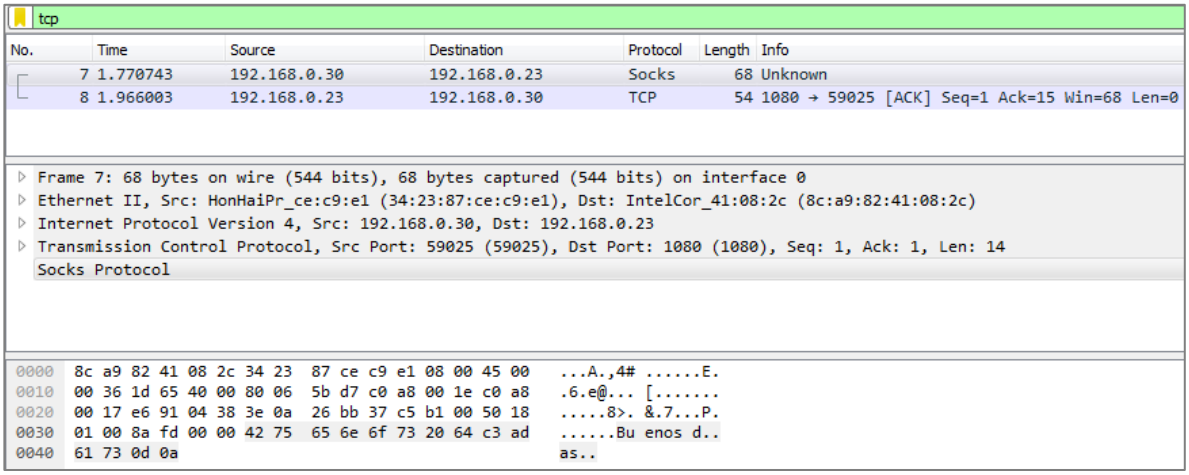

Figura 28. Captura de envío de mensaje desde la aplicación cliente al servidor con Wireshark.

Se observa el paquete de transmisión desde el cliente hacia el servidor, con las mismas especificaciones de direcciones y de puerto, en la descripción en la parte inferior se puede visualizar el mensaje enviado (*"Buenos días"*), en el cuadro derecho de la parte inferior con la codificación en hexadecimal en la parte derecha.

Por último, se analiza la transmisión durante el envío de una imagen desde la aplicación cliente:

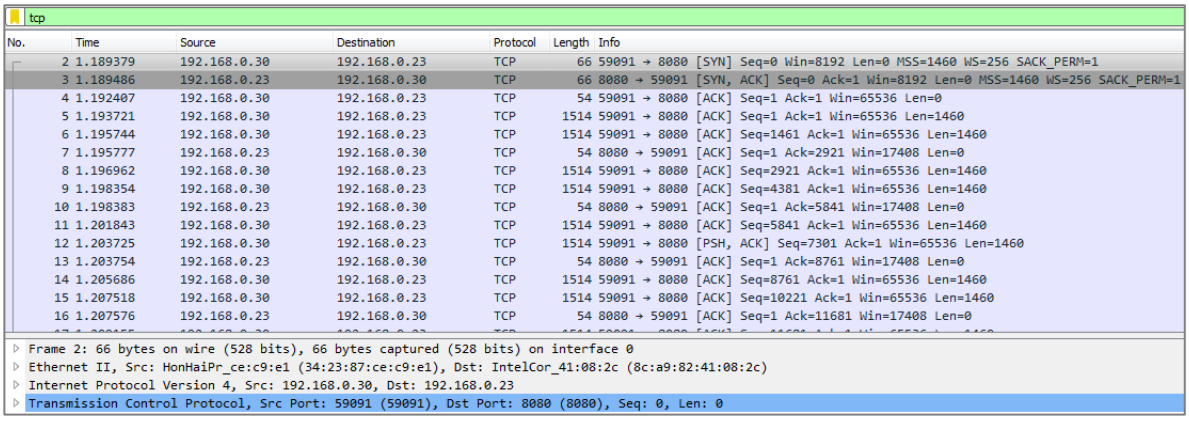

Figura 29. Captura de envío de imagen desde la aplicación del cliente hacia el servidor con Wireshark.

Se logran observar una gran cantidad de paquetes desde la dirección IP del cliente como origen, lo que corresponde en conjunto a la matriz de bytes obtenida a partir de la imagen enviada. De igual manera se visualizan paquetes en sentido contrario para confirmación del acuse de recibo, como parte del protocolo para el envío de la matriz.

8.1. Resultados

El chat satisface los objetivos propuestos, sus funcionalidades brindan al cliente los medios necesarios para lograr comunicarse con la empresa y recibir información sobre sus pedidos.

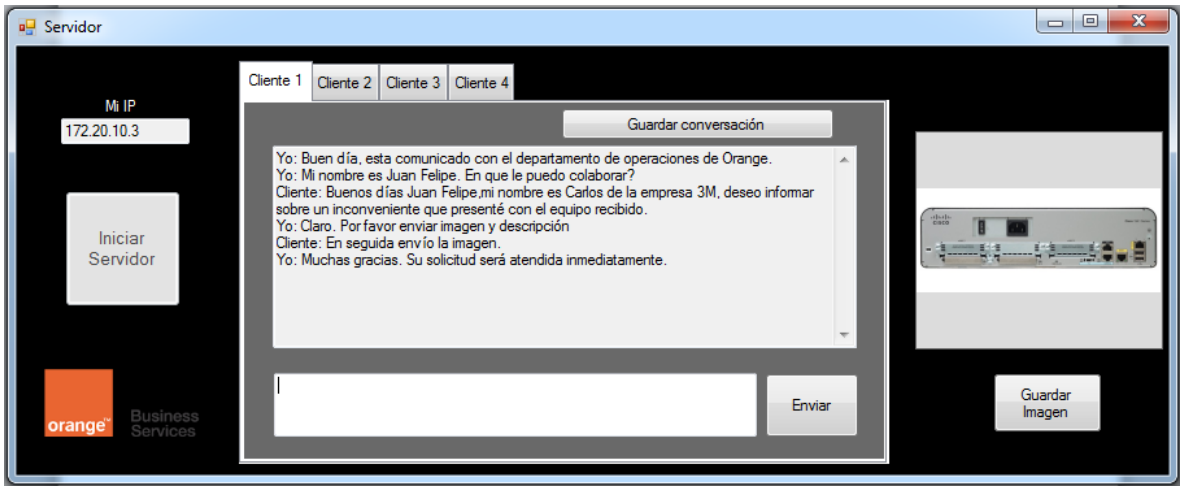

Figura 30. Interfaz de la aplicación servidor en ejecución.

En la figura 30 se observa la interfaz del servidor, con la cual da inicio a las conexiones entrantes y posteriormente establece la comunicación con los usuarios que acceden a la aplicación de manera inmediata (tiempo promedio de conexión: 1 segundo). Posee el número de pestañas según los clientes que solicitan el servicio, facilitando el acceso a cada uno de manera independiente.

El operario del servidor recibe las notificaciones de cada acción que suceda con los usuarios: si un cliente se conectó o se desconectó, si recibe un mensaje nuevo o una imagen. Así mismo, posee la opción de almacenar la información, tanto las imágenes recibidas como los mensajes transmitidos.

Por otro lado, el usuario de la aplicación cliente tiene la libertad de conectarse con el servidor, siempre y cuando éste se encuentre disponible.

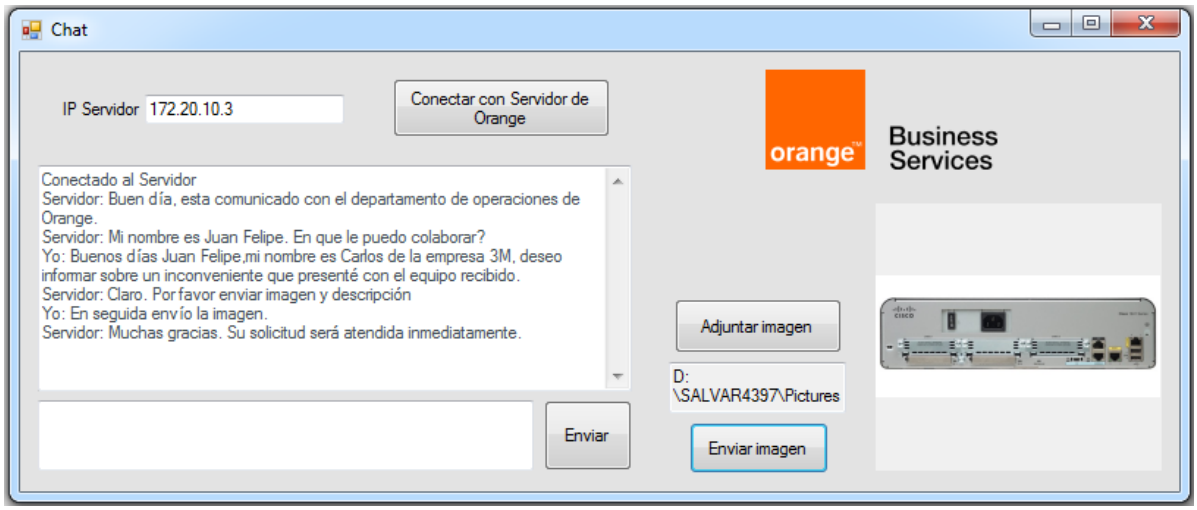

Figura 31. Interfaz de la aplicación cliente en ejecución.

En la aplicación cliente visualizada en la figura 31, el usuario realiza la conexión con solo un botón, posteriormente es informado que ha sido conectado con el servidor, de manera que puede iniciar la conversación y recibir la información de interés. A su derecha, posee la opción de adjuntar una imagen según la solicitud, para posteriormente enviarla al departamento de operaciones.

Como conclusión, se presenta esta nueva aplicación a disposición de la empresa, como una nueva alternativa para obtener una mejor interacción con los clientes, garantizando transparencia en los procesos actuales y futuros.

9. CONCLUSIONES

i. La aplicación de mensajería instantánea para la empresa Orange Business Services Colombia S.A. lograría establecer un canal de comunicación alternativo para el acceso de los clientes de la compañía con el departamento de operaciones, de esta manera, el cliente podría adquirir la información concerniente a sus pedidos de una manera más efectiva y confiable.

ii. Al ser proporcionado el archivo ejecutable para acceder a este servicio, el cliente no demandaría la necesidad de instalar algún tipo de programa, es así como la aplicación garantizaría su ejecución desde cualquier ordenador con sistema operativo Windows y que cuente con conexión a internet.

iii. La instrucción de ejecución en segundo plano (*BackgroundWorker*) fue un eje fundamental para el desarrollo de las interfaces cliente y servidor, ya que permite realizar múltiples tareas de manera simultánea. De esta manera, denotó una gran ventaja en escenarios ocurridos durante la ejecución del chat, un claro ejemplo fueron las instrucciones de escucha de sockets para cada cliente de manera independiente.

iv. Las funcionalidades que ofrece la aplicación para el envío de imágenes y almacenamiento de ellas, al igual que las conversaciones realizadas, establecen una garantía para la seguridad de las partes (cliente y servidor), ya que son soportes de la conversación en tiempo real y que pueden ser de utilidad para sucesos que se puedan derivar en el futuro.

v. El software Visual Studio utilizado para el desarrollo del programa ofrece una interfaz de diseño sencilla a través de la creación de *Windows Form*, de esta manera, permite la ejecución de la aplicación por medio de archivos ejecutables desde el sistema operativo *Windows*.

vi. Con el uso del software Wireshark fue posible detallar los protocolos existentes en la transmisión entre el cliente y el servidor, además de garantizar el envío de los paquetes y su respectiva recepción a través del indicador de acuse de recibo (ACK).

10. BIBLIOGRAFIA

[1] "La Satisfacción del Cliente". Julio 2015. Disponible en: http://www.promonegocios.net/clientes/satisfaccion-cliente.html

[2] "Correo Electrónico vs Teléfono, ¿cuál es mejor?". Octubre 2013. Disponible en: http://budowebs.net/correo-electronico-vs-telefono-cual-es-mejor/

[3] "Chat – Tipos de chat". Octubre 2005. Disponible en: http://www.uv.es/avirtual/internet/t_6_2.htm

[4] "Skype for Business". Disponible en: https://products.office.com/en-in/skype-for-business/online-meeting-solutions

[5] Live Person Chat: Disponible en: https://register.liveperson.com/pricing?_ga=1.191581290.1183461264.1474476837

[6] Volano Chat. Disponible en: http://www.volano.com/download.html

[7] CLIFFORD, Catherine."Top 10: Apps de mensajería instantánea". Diciembre 2013. Disponible en: https://www.entrepreneur.com/article/266543

[8] Aplicaciones – ABC Tecnología. "Facebook Messenger ya tiene más de mil millones de usuarios como WhatsApp". Disponible en: http://www.abc.es/tecnologia/moviles/aplicaciones/abci-facebook-messenger-tiene-mas-millones-usuarios-201607201706_noticia.html

[9] MUTTON, Paul. "IRC Hacks". O´Reilly Media Inc. 2004.

[10] Data-chat. Disponible en: http://donweb.com/es-co/datta-chat-live

[11] Zopim Chat. Disponible en: https://es.zopim.com/pricing

[12] Olark. Disponible en: https://www.olark.com/pricing

[13] ClickDesk Chat. Disponible en: https://www.clickdesk.com/pricing

[14] FALL, Kevin R. – STEVENS, W. Richard. "TCP/IP Illustrated, Volume 1: The Protocols". Addison-Wesley, 2011. Página 16.

[15] “El modelo OSI”. Disponible en: http://www.exa.unicen.edu.ar/catedras/comdat1/material/ElmodeloOSI.pdf. Página 1.

[16] “IPv4 Adressing”. Network Basic. Página 3. Editor NOITE S.C. Disponible en: https://books.google.es/books?id=UNxyCwAAQBAJ&dq=ipv4&hl=es&source=gbs_navlinks_s

[17] BLANK, Andrew G. “TCP/IP JumpStart: Internet Protocol Basics”. Jhon Wiley & Sons, 2006. Página 68.

[18] ATELIN, Philippe / DORDOIGNE, José “TCP/ IP y protocolos de internet”. Editorial ENI, (2007). Barcelona. Páginas 71-73.

[19] AMAYA, Carlos. “Seguridad Avanzada en Redes de Datos – Puertos”. Medellín, Colombia. UNAD. 2013. Disponible en: http://datateca.unad.edu.co/contenidos/233015/233015Exe/leccin_24_puertos.html

[20] “Dynamic Vs. Static IP Addressing In A TCP/IP Environment”. Disponible en: http://www.dpstele.com/dps/protocol/2001/may-jun/ip-addressing-tcp-dynamic-static.php

[21] “Wireshark / Windump. Analisis capturas tráfico red. Interpretación Datagrama IP”. Noviembre 2009. Disponible en: https://seguridadyredes.wordpress.com/2009/11/05/wireshark-windump-analisis-capturas-trafico-red-interpretacian-datagrama-ip-actualizacian/

[22] HUGON, Jérome. “C# 6 Desarrolle aplicaciones Windows con Visual Studio 2015”. Ediciones ENI. Diciembre 2015. Barcelona. Página 19.

[23] “Lenguaje Visual C#“. Disponible en: https://msdn.microsoft.com/es-es/library/aa287558%28v=vs.71%29.aspx

[24] BISHOP, David. “A complete guide to C#”. Sudburry, Massachussets. Jones and Bartlett learning, 2004