UNIVERSIDAD METROPOLITANA DE EDUCACIÓN, CIENCIA Y TECNOLOGÍA
UMECIT

DIRECCIÓN DE INVESTIGACIÓN Y POSTGRADO

ANÁLISIS DEL PROTOCOLO DE APLICACIÓN DEL FORO DE DISCUSIÓN JERÁRQUICO

Tesis Para Optar Por El Título De Maestría En Sistemas Computacionales Con Énfasis En Redes Y Comunicaciones

Por:
Raúl Enrique Dutari Dutari

Profesor Asesor:
Ingeniero Ronald Mitre, M.Sc.

Panamá, República De Panamá

Noviembre, 2011

DEDICATORIA

Al supremo creador, porque sin su apoyo es imposible lograr algo en esta vida.

A mis padres, Juan B. y María E., quienes con sus esfuerzos, apoyo y amor me alentaron para alcanzar la realización de uno de mis más preciados sueños.

A Yodania del Carmen Quirós Aguilar "Yodita", porque su amor, su cariño, su respeto y su amistad, son el impulso, el "motorcito" que me animó en los momentos más difíciles de este proyecto.

AGRADECIMIENTO

Agradezco profundamente el apoyo brindado por todas aquellas personas que, de una u otra forma, han contribuido a que este trabajo de graduación sea una realidad.

UNIVERSIDAD METROPOLITANA DE EDUCACIÓN, CIENCIA Y TECNOLOGÍA
ANÁLISIS DEL PROTOCOLO DE APLICACIÓN DEL FORO DE DISCUSIÓN JERÁRQUICO

AUTOR: RAÚL DUTARI
NOVIEMBRE, 2011
NRO. PÁG. 149

RESUMEN

El presente trabajo titulado ***Análisis Del Protocolo De Aplicación Del Foro De Discusión Jerárquico***, se realizó con el objetivo de ***Estudiar el protocolo de aplicación del Foro De Discusión Jerárquico***, mediante la metodología descriptiva y documental. Su objeto de estudio fue el ***protocolo de comunicaciones*** que utiliza la aplicación informática denominada como ***Foro De Discusión Jerárquico*** –en adelante, ***FDJ***-, implementada por **[DUTA01]** y sus resultados indican que el protocolo de comunicaciones que utiliza el ***FDJ*** es un prototipo que debe ser mejorado sustancialmente, a fin de que optimice sus niveles de cumplimiento de los estándares de comunicaciones fundamentados en los modelos ***OSI*** o ***TCP/IP***.

DESCRIPTORES: Protocolo de comunicaciones, Arquitectura de protocolos, Modelo OSI, Modelo TCP/IP, Primitivas de servicio.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

INTRODUCCIÓN

La naturaleza intrínseca del ser humano lo ha llevado, desde tiempos inmemoriales, a buscar el apoyo de los grupos de sus semejantes. Esta tendencia gregaria-, aunada a su nivel de raciocinio superior, le han permitido ser el creador de un sinnúmero de herramientas que le facilitan sus labores diarias, que, a su vez, lo han llevado a evolucionar como el ser que domina al planeta.

Bajo este contexto, el computador se puede considerar como una de sus creaciones más versátiles; ya que, desde sus orígenes, esta valiosa herramienta se ha convertido en su auxiliar cotidiano de las tareas más disímiles que puede realizar. Adicionalmente, el poder de este instrumento maravilloso se potencia hasta niveles críticos cuando se asocia con otros equipos similares, formando las conocidas redes de computadoras.

Sin embargo, de manera similar a las personas -que requieren de un idioma y cultura comunes para poderse comunicar entre sí-, las computadoras deben comunicarse siguiendo una serie de normas y reglas preestablecidas; los llamados protocolos de comunicación, que resultan ser unos componentes de software bastante misteriosos, aún para los estudiantes de postgrado, cuando se analizan internamente en detalle.

En **[DUTA01]** se plantea una situación típica donde se utilizan varios de estos protocolos. En particular, se bosquejan parcialmente los componentes estructurales básicos, de un protocolo que se utiliza como parte de los sistemas de comunicación del llamado ***Foro de Discusión Jerárquico***, -conceptualizado por **[NÚÑE99]** y en adelante, ***FDJ***-.

Esta investigación tratará, precisamente, de ilustrar didácticamente los elementos descriptivos esenciales que se requieren para analizar, a nivel estructural y funcional, el protocolo de comunicaciones de la herramienta de software antes mencionada.

CAPÍTULO 1

CAPÍTULO 1: EL PROBLEMA.

A continuación, se analizarán las generalidades del problema de investigación que se enfrentará.

1.1 PLANTEAMIENTO.

El ***FDJ***, es una aplicación que ofrece un área de trabajo compartida donde sus usuarios pueden intercambiar opiniones o ***Enunciados***. Estas opiniones se estructuran de manera jerárquica, de acuerdo al modelo de hilos de discusión planteado por **[NÚÑE99]**.

A nivel funcional es, esencialmente, una herramienta que ofrece el servicio de mensajería instantánea, de manera similar al MS-Messenger, Yahoo Messenger u otras similares, con la diferencia que el diálogo se desarrolla de manera jerárquica; en tanto que, en las herramientas tradicionales de mensajería instantánea antes mencionadas, el diálogo se presenta de manera lineal.

Cada uno de los Enunciados registrados dentro del diálogo, estará categorizado bajo una de las naturalezas permitidas[1]. En consecuencia y con base en dichas naturalezas, el ***FDJ*** puede calcular el llamado ***Estado De Plausibilidad*** de cualquiera de los enunciados que lo integran.

---

[1] Las naturalezas que pueden asumir los enunciados dentro del foro son: *Aclaración (A), Evidencia A Favor (F), Evidencia En Contra (C), Desactivado (D) y Opinión Primaria (O)*.

En **[DUTA01]**, se establece que el ***FDJ***, a nivel estructural, está constituido esencialmente, como una aplicación que sigue el modelo cliente-servidor. Este modelo se describe en detalle en **[FORO07]**.

[DUTA01] establece, de manera básica los mecanismos de comunicación del ***FDJ***, como una serie de flujos de datos basados en sockets en implementados en Java. Adicionalmente, también se establece que el ***FDJ*** es, en sí mismo, un prototipo, sujeto a mejoras; en particular, el modelo de comunicación empleado -protocolo-.

Sin embargo, analizando en detalle la referencia antes señalada, se observa que el tratamiento realizado a la documentación del protocolo de comunicación que utiliza puede ser mejorada significativamente, a nivel de sintaxis, semántica, temporización, así como en otros aspectos, que serán objeto de estudio en esta investigación.

*El producto de esta investigación será el análisis del protocolo de comunicaciones del **FDJ**, a nivel de especificación de unidades de datos de protocolo (**PDU's**) y primitivas de servicio, donde se evidencien los elementos estructurales y funcionales característicos de este tipo de estándar.*

En tal sentido se debe recalcar que lo que se pretende es, fundamentalmente, ofrecer un estudio -que posteriormente se puede completar a nivel de diseño y sea implementado con relativamente pocas dificultades, en un lenguaje de programación que ofrezca soporte para establecer enlaces de comunicación vía redes de datos-, que se pueda llevar a la práctica en versiones futuras del ***FDJ***, implementado en **[DUTA01]**.

1.2 ANTECEDENTES.

En **[NÚÑE99]** se propone la fundamentación teórica del foro de discusión, que se utilizó como base de la implementación de **[DUTA01]**, pues está basado en un modelo de hilos de discusión -es jerárquico- y, al mismo tiempo, permite apoyar los procesos de toma de decisiones en grupo, basándose en la teoría de plausibilidad de **[AGÜE87].**

Sin embargo a la fecha -y previa investigación de posibles trabajos previos-, no se conocen implementaciones de foros de discusión en línea que se basen en estos conceptos simultáneamente, diferentes a la presentada por **[DUTA01]**.

Adicionalmente, dado que el ***FDJ*** expuesto en **[DUTA01]**, es un prototipo, se puede afirmar, sin temor a equivocarse, que a la fecha, no hay antecedentes de implementación de algún protocolo de aplicación, orientado a este tipo de herramientas.

1.3 JUSTIFICACIÓN.

Examinando la literatura existente relacionada a la arquitectura de protocolos de telecomunicaciones -en libros de texto de la especialidad, tales como **[FORO07]**, **[FOCH06]**, **[COME00]** así como **[STAL04]**-, se observa que el tratamiento de estos temas es teórico, sin ejemplificaciones claras, dejando interrogantes al momento en que se pretende implementar un protocolo de comunicaciones desde cero.

Por otro lado, si se analizan las ***Peticiones De Comentarios*** del grupo especial de ingeniería de internet -mejor conocido en inglés como ***Internet***

Engineering Task Force (IETF)-, se observa la situación extrema a la anterior, ya que son documentos altamente técnicos, donde los neófitos de la materia pueden resultar fácilmente abrumados por los detalles técnicos de implementación.

Finalmente, si se examinan referencias bibliográficas tales como **[DAYJ08]**, se observa un discurso de informática teórica basado en máquinas de estado finito, donde sólo aquellas personas que tienen ése tipo de formación, pueden comprenderlo a cabalidad.

En tal sentido, la importancia de este proyecto de investigación radica en que presenta un ejercicio ordenado -a la vez que didáctico- de la documentación inicial que se debe elaborar para responder a la necesidad de analizar un protocolo de comunicaciones funcional, con la meta a futuro de diseñarlo e implementarlo –es decir, la fase de análisis-.

1.4 OBJETIVOS.

Las metas que pretende alcanzar este proyecto se plasman en los objetivos que se expresan a continuación:

1.4.1 General.

1. Estudiar el protocolo de aplicación del Foro De Discusión Jerárquico.

1.4.2 Específicos.

1. Establecer un marco teórico, que fundamente el análisis de los protocolos de comunicaciones, a nivel de primitivas de servicio.

2. Comprender la arquitectura de comunicación que emplea el ***FDJ***.

3. Analizar la estructura interna de los mensajes que se intercambian en el ***FDJ***.

4. Contrastar la arquitectura de comunicaciones del ***FDJ***, ante los elementos estructurales y funcionales que fundamentan el estudio de los protocolos de comunicaciones, para resaltar las fortalezas y debilidades del modelo bajo análisis.

5. Establecer las funciones que utiliza el ***FDJ*** para intercambiar información, en términos de primitivas de servicio, resaltando sus fortalezas y debilidades.

1.5 ALCANCE.

Este proyecto de investigación tiene entre sus metas:

1. **La propuesta será planteada a nivel de análisis:** Es decir, la norma existente se estudiará, a nivel estructural y funcional, para establecer sus debilidades y fortalezas.

2. **La propuesta se ajustará, en lo posible, a los estándares existentes, con base en las normas del Modelo de Interconexión de sistemas abiertos -OSI, así como en el modelo *TCP/IP*-:** En tal sentido, el estudio planteado deberá ajustarse a las técnicas de análisis del diseño estructurado, planteando la existencia o no de capas, si se emplean o no paquetes de longitud fija, si se incluyen o no atributos para la detección y corrección de errores, o si se dispone o no de un sistema de secuenciación de paquetes, entre otras características.

1.6 DELIMITADORES.

Dada la complejidad del proyecto propuesto, el alcance de esta investigación estará circunscrito por los siguientes factores delimitadores:

1. **El estudio, en sí, se fundamentará en la documentación existente relacionada al *FDJ*, así como en el análisis de su código fuente, localizados en [DUTA01]:** dado que el ***FDJ*** es un prototipo de aplicación, la documentación existente relacionada con él, está limitada esencialmente a lo que aporta la referencia antes mencionada, que incluye todo el código fuente de la aplicación.

2. **La investigación no se enfocará en los aspectos de comunicación entre clases:** ya que el centro de interés del proyecto es el protocolo de comunicación que utilizan los clientes y el servidor del ***FDJ*** para comunicarse, los aspectos que se relacionen con la comunicación local entre sub-clases no serán mayormente considerados.

3. **El proyecto se llevará explícitamente hasta la etapa de análisis:** Es decir, esta investigación no pretende ofrecer una propuesta formal de diseño o implementación de un protocolo de comunicaciones para el ***FDJ***. En su lugar, la propuesta entregada abarca el análisis de fortalezas y debilidades que caracterizan al sistema de intercomunicación de la aplicación.

Algunos de los aspectos antes descritos, pueden ser planteados como trabajos de investigación, que se pueden derivar de los resultados de este proyecto.

CAPÍTULO 2

CAPÍTULO 2: MARCO TEÓRICO: PROTOCOLOS DE COMUNICACIONES.

En esta sección se presenta el marco teórico que sustenta esta investigación. En tal sentido:

1. Se establece el concepto de protocolo de comunicaciones.

2. Se plantean los elementos claves que se requieren para tipificar a los protocolos de comunicaciones.

3. Se analizan las funciones más relevantes cumplen los protocolos dentro de los procesos de comunicación en los sistemas distribuidos.

4. Se clasifican los protocolos de comunicaciones.

5. Se describen las funciones de las organizaciones que se encargan de normalizar los protocolos de comunicaciones.

6. Se establecen los principios que deben seguirse al momento de plantear un protocolo de comunicaciones, en términos de capas y primitivas de servicios.

2.1 CONCEPTOS FUNDAMENTALES.

La literatura especializada en el tema, enfoca este concepto de varias maneras.

Para **[STAL04]**, un protocolo de comunicaciones consiste en el conjunto de reglas que gobiernan el intercambio de datos entre dos entidades. Bajo este

enfoque, se entiende como ***entidad*** a cualquier ente que sea capaz de enviar y recibir información.

Por su parte **[FORO07]**, establece que en la transmisión de datos, un **protocolo** consiste en un conjunto de reglas o convenciones que gobiernan todos los aspectos del intercambio de información entre sistemas.

Los datos que se intercambian serán agrupados en bloques especiales denominados ***Unidades de Datos de Protocolos*** (***PDU***, del inglés Process Data Units) **[DAYJ08]**.

Por otro lado, las entidades adyacentes entre sí –verticalmente hablando-, que intercambian ***PDU's*** entre sí y poseen interfaces[2] de comunicación comunes entre sí, se denominan ***Capas*** o ***Niveles***. En cada capa se realizan una serie de funciones interrelacionadas, necesarias para comunicarse con los niveles superiores e inferiores.

Los conjuntos de protocolos y capas agrupados bajo un sistema de estándares intercompatibles se denominan ***Arquitectura de Protocolos*** **[FORO07].** Todas las funciones de comunicación, dentro de una arquitectura de protocolos dada, se deben realizar en función de las interfaces que se definen entre las capas jerárquicas **[TANE03]**.

---

[2] Interface se debe interpretar en el sentido de que es una serie operaciones y servicios encapsulados y preestablecidos **[TANE03]**.

2.2 UNIDAD DE DATOS DEL PROTOCOLO (PDU).

Como ya se ha mencionado, el intercambio de información entre las entidades se debe realizar utilizando unidades de datos finitas -las ***PDU's***, en general-. Dependiendo de la época y del protocolo particular que se maneja, han recibido diferentes nombres, tales como marcos, frames, paquetes, segmentos, celdas, mensajes, entre otros, aunque conceptualmente todas giran en torno al mismo concepto: unidad finita de intercambio de información entre entidades **[DAYJ08]**.

Esencialmente, las ***PDU's*** tienen como estructura a tres elementos básicos: ***Header***, ***Trailer*** y los ***datos del usuario***.

- **Header (Encabezado):** la mayoría de los protocolos lo contienen. Constan de campos que normalmente tienen longitud fija, dejando de último a los campos que pueden tener longitudes variables. Contienen identificadores tales como: identificadores de protocolos -es decir, el tipo de protocolo que se maneja-, así como campos se señalan los atributos o acciones que se espera que se manipulen con el paquete de datos enviado.

- **Trailer (Cola):** las ***PDU's*** de algunos protocolos suelen contener un elemento que se denomina ***trailer***, que puede ser interpretado como una cola que cierra el paquete de datos. Su uso más común se asocia al transporte de los códigos de redundancia cíclica **(CRC)**, de comprobación de errores, siempre y cuando el protocolo se implemente cerca de la capa física. En otros casos, su utilidad es muy marginal, por lo que no es frecuentemente implementada.

- **Datos Del Usuario:** consisten en la información que se desea compartir entre las entidades que intervienen en el proceso de comunicación.

El conjunto de ***header*** y ***trailer*** constituye la llamada información de control del protocolo (***PCI***). Generalmente, deben tener un tamaño reducido en comparación con la longitud de los datos del usuario, para asegurar que el transporte de datos entre las entidades involucradas se realiza de manera relativamente eficiente.

En la ilustración que se muestra a continuación, se puede observar la estructura de una ***PDU***, en general.

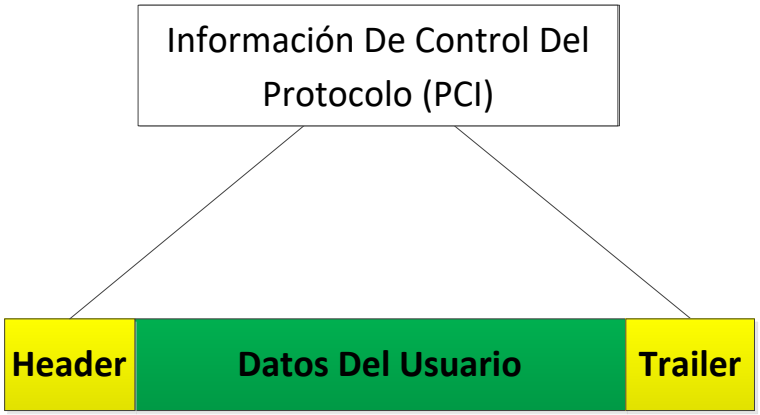

Ilustración 2.1: Estructura General De La PDU
Fuente: [DAYJ08]

2.3 ELEMENTOS ESTRUCTURALES.

Para **[FORO07]**, los elementos estructurales clave que tipifican a los protocolos de comunicaciones son: Su sintaxis, su semántica y su temporización.

2.3.1 Sintaxis.

La ***sintaxis del protocolo*** especifica la ***estructura*** del formato que deben tener los datos. Es decir, el orden en el cual se presentan.

Por ejemplo, un protocolo de comunicaciones sencillo podría esperar que los primeros ocho bits de datos representen a la dirección del emisor, en tanto que los segundos ocho bits, constituyen la dirección del receptor y el resto del flujo consiste en el mensaje propiamente dicho.

2.3.2 Semántica.

Por otro lado, la ***semántica del protocolo*** establece el significado de cada sección de bits que contiene el mensaje. Es decir, se consideran aspectos tales como la ***interpretación*** de un determinado patrón dentro del paquete de información, así como las ***acciones*** que se deben tomar en función de dicha representación.

Por ejemplo: Si dentro del mensaje aparece una dirección; ella identificará a la ruta que debe seguir el paquete de manera inmediata, o el destino final que tendrá el mensaje.

2.3.3 Temporización.

Dentro de un protocolo de comunicaciones, la ***temporización*** se caracteriza por dos atributos básicos:

- En qué ***momento*** de la transmisión se deben enviar los datos a través del medio;

- Que ***velocidad*** de transmisión se deben utilizar en su envío.

Por ejemplo, si un emisor envía datos a una velocidad de 100Mbps, pero el receptor correspondiente puede procesar datos únicamente a 1Mbps, el proceso de transmisión sobrecargará al receptor y se perderán gran cantidad de datos.

En tal caso, los procesos de temporización entre las entidades deben conducir a que se sincronicen a 1Mbps, para garantizar que ambas pueden comunicarse sin pérdidas de información potenciales.

2.4 FUNCIONES.

Según lo que establece **[STAL04]**, las funciones más relevantes que deben cumplir los protocolos de comunicaciones[3] se establecen a continuación:

2.4.1 Encapsulamiento.

La función de ***encapsulamiento*** de los protocolos consiste, esencialmente, en la adición de información de control, a los datos que se intercambian entre las

---

[3] Es relevante aclarar que algunas de estas funcionalidades estarán ausentes en ciertos protocolos, ya que de lo contrario representará una duplicación innecesaria de ellas. Sin embargo, otras funcionalidades deberán repetirse en ciertos protocolos que se ubican en diferentes niveles.

entidades que participan en el proceso de comunicación, vía las ***PDU's*** respectivas.

Esencialmente, la información de control se debe interpretar como ***atributos adicionales*** a la información propiamente compartida. La ***PDU*** no puede cumplir satisfactoriamente su objetivo de intercambio de datos sin ella. De hecho, en algunos tipos de protocolos, las ***PDU's*** contienen exclusivamente información de control.

La información de control se puede categorizar básicamente en tres tipos de clases:

- **Direccionamiento:** En este caso, contendrá los identificadores de origen -emisor- y destino -receptor- de la ***PDU***. Es decir, qué entidad originó el mensaje, así como la entidad que debe recibirlo.
- **Detección de errores:** Por otro lado, la ***PDU*** debe incluir información que ofrezca la posibilidad de identificar -y de ser posible corregir- errores en la transmisión, generalmente basados en esquemas de secuencias de verificación.
- **Control del protocolo:** En la ***PDU*** también se incluye información suplementaria que posibilita la realización de las funciones restantes que se mencionan en esta sección.

2.4.2 Segmentación Y Ensamblado.

Generalmente, la transferencia de datos entre entidades debe ser realizada utilizando bloques de datos de tamaños limitados, aunque las aplicaciones

manipulen sus **mensajes**[4] como cadenas de bytes continuas; sobre todo, cuando se consideran las funciones de los protocolos de capas inferiores.

El procedimiento mediante el cual, los protocolos de capas inferiores ***dividen*** los mensajes de las aplicaciones se denomina ***segmentación***. La ***PDU***, en consecuencia, consiste en cada uno de los bloques de datos que se intercambian entre entidades.

Por otro lado, el procedimiento inverso a la segmentación se denomina ***ensamblado***, y no es más que la ***reconstrucción*** del mensaje original que envió la entidad origen, a la entidad destino, y que puede ser manipulado directamente por la aplicación correspondiente.

En la ilustración que se muestra a continuación, pueden observarse las funciones de segmentación y ensamblado, conjuntamente con la de encapsulamiento.

---

[4] Se denomina ***mensaje*** a la unidad lógica de transferencia de información entre aplicaciones **[STAL04]**.

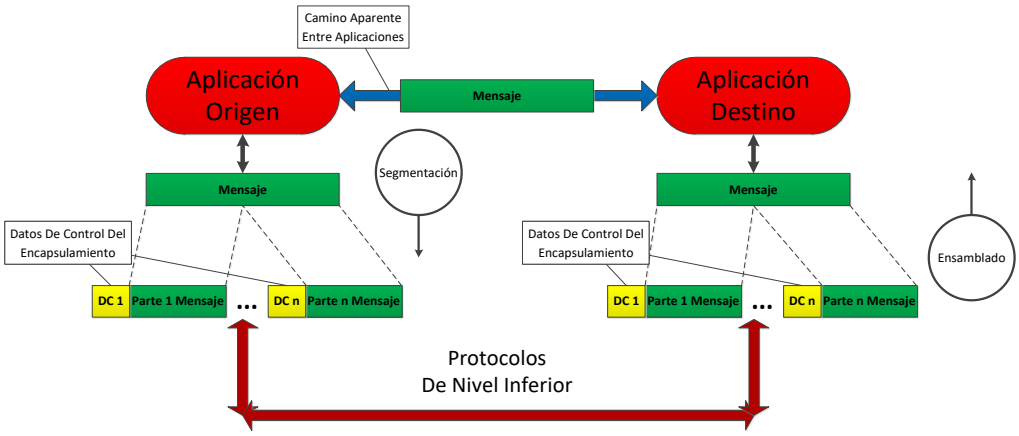

Ilustración 2.2: Procesos De Segmentación / Ensamblado y Encapsulamiento
Fuente: El Autor

2.4.2.1 Tamaño De Las *PDU's*.

El tamaño o cantidad de bytes que debe contener un bloque de texto son motivo de controversia. Así algunas de las ventajas de utilizar ***PDU*** de tamaño reducido son:

- En sus capas más inferiores, las redes de comunicaciones generalmente aceptan bloques de tamaño limitado (***ATM*** 53bytes, ***Ethernet*** 1526bytes).

- Mientras menor sea el tamaño de la ***PDU***, menor será la cantidad de datos a retransmitir en caso de errores.

- Las ***PDU's*** de tamaño reducido posibilitan que el acceso a los medios de transmisión compartidos se realice de manera más equitativa y con niveles de retardo inferiores -evitando que una entidad cualquiera monopolice el acceso al medio-, a los que se presentarían cuando se utilicen ***PDU's*** de mayor tamaño.

- Las entidades receptoras y emisoras necesitarán reservar cantidades menores de memoria temporal para buffers.

- Periódicamente, las entidades que intervienen en los procesos de comunicación de datos, requieren que su intercambio se detenga cada cierto tiempo, con el fin de completar tareas de tipo administrativo.

- El tamaño o cantidad de bytes que debe contener un bloque de texto son motivo de controversia. Así algunas de las ventajas de utilizar ***PDU*** de tamaño reducido son:

Por otro lado, el uso de bloques de transmisión de gran tamaño, se puede justificar -y en consecuencia, el uso de bloques de tamaño reducido será desventajoso-, por las razones que se enuncian a continuación **[KOZI05]**:

- Mientras más grande es la ***PDU*****,** menor es el porcentaje de información de control suplementaria que se debe incluir.

- La cantidad de interrupciones que deberá atender el procesador será menor, mientras menos ***PDU's*** se reciban.

- Así mismo, el tiempo relativo que debe dedicar el procesador a la atención de las ***PDU's*** pequeñas, será mayor a medida que aumenta su número.

Los diseñadores de protocolos deben considerar todos estos factores, al momento de establecer los tamaños máximos y mínimos de sus ***PDU's***.

2.4.3 Control De La Conexión.

Esta función define, esencialmente, la forma en que se ***administra*** la conexión entre las entidades involucradas en el proceso de comunicación.

Las conexiones entre entidades transmisoras se pueden realizar utilizando dos formas básicas de enlace:

2.4.3.1 Enlaces No Orientados A Conexión.

Bajo este contexto, la entidad que emite las ***PDU's*** las programa de tal manera que cada una será tratada por los canales de comunicación de manera ***independiente*** a las que se han recibido con anterioridad.

Este modo de conexión es viable en los casos en que no se prevé un intercambio de datos considerable entre las entidades. También es funcional cuando los protocolos involucrados en el proceso de comunicación no requieren controlar detalladamente el proceso de comunicación. Salvo ciertas situaciones particulares, este sistema de comunicación no es el más utilizado.

Un ejemplo de este comportamiento se puede observar en los datagramas del protocolo ***UDP*****.**

2.4.3.2 Enlaces Orientados A Conexión.

La característica fundamental de este modo de conexión, radica en que cada entidad involucrada en el proceso de comunicación, enumera de manera secuencial cada ***PDU*** que envía a la otra entidad.

Ambas entidades ***saben*** que están involucradas en un proceso de comunicación, de modo que controlan coordinadamente las recepciones y envíos de los paquetes de datos, a través de un canal lógico que se mantiene abierto durante todo el proceso.

En contraste con los enlaces no orientados a la conexión, en las situaciones en que se prevé un gran intercambio de información entre las entidades, o los protocolos involucrados requieren que se controlen algunos detalles especiales de la transferencia de información, de manera dinámica -tales como

la detección de errores-; se puede llegar a requerir, de manera obligatoria, la transferencia de datos orientada a las conexiones. Este es el sistema de comunicación que más se utiliza en las redes de datos.

En este tipo de transferencia se dan tres fases **[FORO07]**:

- **Establecimiento de la conexión:** Durante esta fase, las entidades acordarán el intercambio de datos. Generalmente, una de las entidades enviará una solicitud de conexión (usando un sistema de transferencia no orientada a conexión) a la otra.

  Dependiendo de la complejidad del sistema, simplemente se devolverá un mensaje indicando que la solicitud se acepta o rechaza; o seguirá una fase completa de negociación de sintaxis, semántica y temporización, parametrizando el protocolo de comunicación. Sin embargo, en general involucra tres acciones al menos– denominado diálogo de tres partes-, que se observan en la siguiente ilustración -las fechas amarillas denotan información de control-.

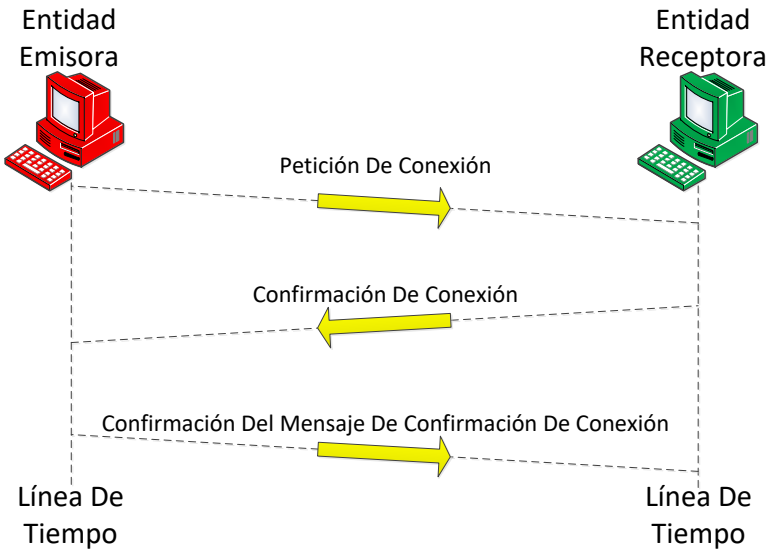

Ilustración 2.3: Diálogo De Tres Partes Para Establecer La Conexión
Fuente: [FORO07]

- **Transferencia de datos:** En esta fase, las entidades intercambian datos e información de control -para control de flujo o errores, por ejemplo-. El intercambio se puede dar en un único sentido, aunque lo más común es que se realice en doble sentido. Nuevamente, la discusión entre las entidades se lleva a cabo vía diálogo de tres partes, como se aprecia en la siguiente ilustración -las fechas amarillas denotan información de control, en tanto que la fecha azul señala datos-.

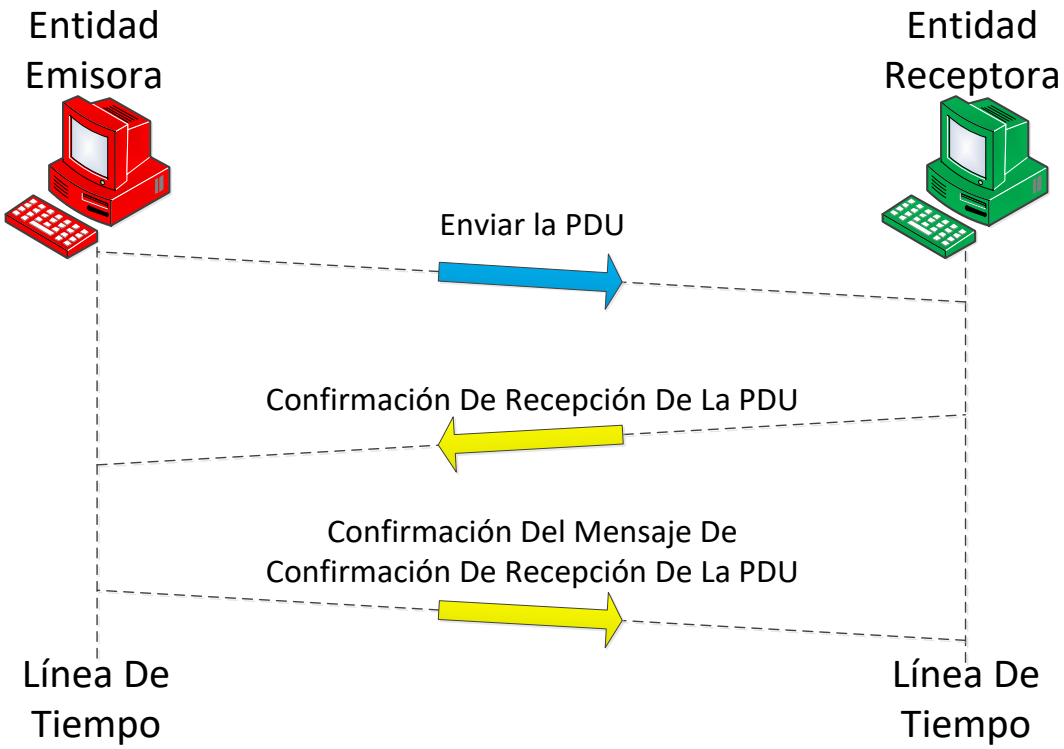

Ilustración 2.4: Diálogo De Tres Partes Para La Transferencia De Datos
Fuente: [FORO07]

- **Cierre de la conexión:** Si todo transcurre normalmente, finalizado el proceso de intercambio de información, cualquiera de las entidades puede solicitar a la otra la finalización de la conexión, vía solicitud de cierre de conexión. También puede ser ordenado por una entidad central. En este caso, nuevamente la conversación entre entidades se realiza empleando el diálogo de tres partes, como se aprecia en la

siguiente ilustración -las fechas amarillas denotan información de control-.

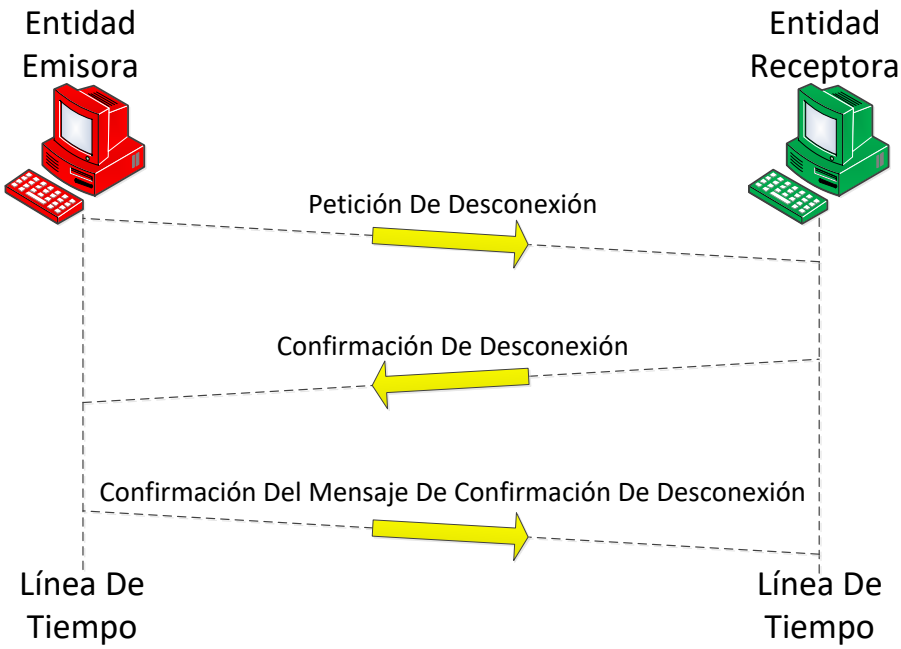

Ilustración 2.5: Diálogo De Tres Partes Para Finalizar La Conexión
Fuente: [FORO07]

Es relevante destacar que en protocolos más elaborados, pueden presentarse fases adicionales, tales como la interrupción del diálogo, con las correspondientes recuperaciones de conexión.

También es importante resaltar que la numeración secuencial de las ***PDU's***, asociada a este modo de comunicación, se relaciona fuertemente, con las tres funciones que se considerarán a continuación: La entrega en orden, el control del flujo y el control de errores.

2.4.4 Entrega En Orden.

Por otro lado, la entrega en orden establece fundamentalmente, el orden en que pueden ser recibidos los paquetes.

Esta función obedece a la posibilidad de que las entidades que intervienen en el proceso de comunicación, reciban las ***PDU's*** que se transmiten en un orden diferente al que se enviaron, por la diversidad de rutas que pueden haber para encaminarlas hacia su destino.

En principio, parece una tarea sencilla si cada una de las ***PDU's*** se numera secuencialmente; es cuestión de que se mantenga el orden señalado en la numeración. Sin embargo, al considerar que la longitud del campo en la ***PDU*** donde se almacena dicha información es limitado, se presenta el problema de que dichos números se podrían repetir, mientras más grande sea el tamaño de los mensajes intercambiados.

2.4.5 Control Del Flujo.

Consiste en una operación realizada por la entidad receptora con el fin de limitar la velocidad o cantidad de datos que envía la entidad emisora. Esto, con la finalidad de no se sobrecargue al receptor con una cantidad excesiva de datos, con respecto a su memoria temporal y su capacidad de procesamiento.

El control de flujo es una de las funciones típicas que se deben realizar en varios niveles de protocolos de comunicaciones. Para tal efecto, considere una arquitectura de protocolos simplificada tal como se observa en la siguiente ilustración:

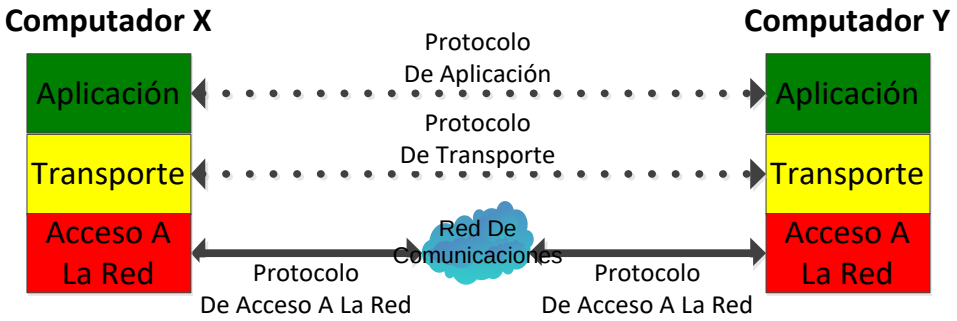

Ilustración 2.6: Arquitectura Simplificada De Protocolos
Fuente: [STAL04]

Es claro que los datos, al fluir entre las capas de la arquitectura, serán objeto de procesamiento por cada uno de los niveles de software y hardware transitados. Esto implica que, en todos esos puntos, se debe controlar la sobrecarga del sistema, vía la función de control de flujo.

Existen una serie de esquemas para control de flujo. El más sencillo, es conocido como el ***Procedimiento De Parada Y Espera***, en tanto que los más eficientes conllevan algoritmos más especializados, tales como el de ***Ventana Deslizante***.

2.4.5.1 Procedimiento De Parada Y Espera.

Bajo este esquema, el emisor debe esperar a que el receptor le confirme que ha recibido cada ***PDU***, antes de intentar enviar otra. De este modo, la entidad destino puede controlar el flujo de datos que le llega, simplemente retrasando la transmisión de las confirmaciones de acuse de recibo (acknowledgement, o ack).

Este esquema, en presencia de un mensaje que se envía utilizando un número reducido de bloques de gran tamaño, funciona bien y de hecho es difícil mejorar sus prestaciones.

Sin embargo, la situación más común consiste en que el emisor segmente el mensaje en una gran cantidad de bloques pequeños, para minimizar la retransmisión de tramas de gran tamaño en presencia de errores, así como para evitar la monopolización del medio de transmisión por parte de una única entidad.

En este último escenario, el problema más relevante radica en la serialización de las ***PDU's*** -sólo puede haber una en tránsito, no se pueden aprovechar las capacidades de intercambio en alta velocidad de los medios de transmisión actuales-. Cuando la longitud del enlace[5] es mayor que la longitud de la ***PDU***, se presentan claras situaciones de ineficiencia.

Si el tiempo de transmisión se normaliza a **1** y el retardo de propagación -el tiempo que le toma a un bit en llegar desde la entidad emisora a la entidad receptora- se expresa como la variable $\boldsymbol{a}$. Se obtienen las siguientes relaciones.

- **Si el tiempo de propagación es menor que el tiempo de transmisión $\boldsymbol{a < 1}$:** En este caso, la ***PDU*** es lo suficientemente larga para que sus primeros bits lleguen a la entidad receptora antes de que la entidad emisora termine su transmisión.

---

[5] La longitud de un enlace en bits se define como el número de bits presentes en el enlace cuando este se ocupa completamente por una secuencia de bits. Matemáticamente, la longitud del enlace $LE = \frac{(R \times d)}{V}$ donde R velocidad de transmisión en bps, d es la distancia del enlace en metros, y V es la velocidad de propagación en el medio, en m/s.

- **Si el tiempo de propagación es mayor que el tiempo de transmisión $a > 1$:** Bajo este escenario, la entidad emisora termina la transmisión de toda la ***PDU*** antes de que el primer bit de la misma llegue a la entidad receptora.

La ilustración que se muestra a continuación, permite observar el comportamiento de este procedimiento de control de flujo.

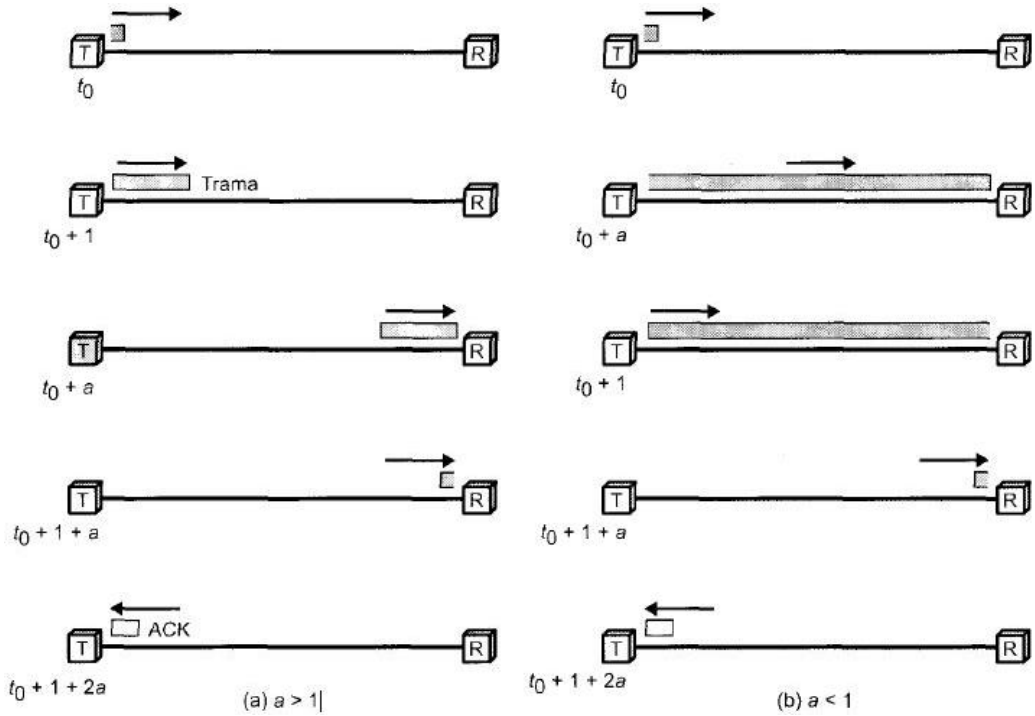

Ilustración 2.7: Control De Flujo Mediante Parada Y Espera
Fuente: [STAL04]

La ilustración anterior presenta una secuencia de transferencias de mensajes, considerada a lo largo del tiempo, para cada valor posible de a. En ambos casos, las cuatro primeras instantáneas muestran el proceso de la transmisión de una trama que contienen datos, y la última muestra la devolución de una trama pequeña de confirmación.

Se debe observar que para $\boldsymbol{a} > \mathbf{1}$, el medio de comunicación está subutilizado siempre, en tanto que para el caso en que $\boldsymbol{a} < \mathbf{1}$, la línea se utiliza ineficientemente.

En resumen, el control de flujo mediante parada y espera conlleva a la utilización ineficiente del medio, en el caso que se presenten velocidades de transmisión elevadas y que los emisores y receptores estén separados por grandes distancias.

2.4.5.2 Procedimiento De Ventanas Deslizantes.

En general, los esquemas más avanzados de control de flujo involucran la concesión de una especie de crédito a la entidad emisora, en el sentido de que se le concede una cantidad de datos pre-establecida que puede transmitir sin esperar confirmación. Tal es el principio fundamental de las ventanas deslizantes.

A continuación, se analizará el comportamiento de dos entidades **A** y **B -** emisor y receptor, respectivamente- conectadas mediante un enlace full dúplex. Posee una reserva de memoria temporal que le permite almacenar hasta **W *PDU's*** sin necesidad de confirmación. Sin embargo, el control sobre las tramas particulares a las que se ha confirmado su recepción, se realizará con base en el número de secuencia que se asignará a cada una de ellas.

Cada vez que **B** confirma la recepción de las tramas previas, le envía un acuse de recibo (ack) a la entidad **A**, señalando:

- El número de secuencia de la siguiente ***PDU*** que se espera recibir.

- Que el receptor **B** está preparado para recibir las **W** tramas siguientes, a partir de la que se recibió previamente.

Bajo este procedimiento, se pueden confirmar simultáneamente la recepción de varias ***PDU's*.** Así por ejemplo, el receptor **B** podría haber recibido las tramas 1, 2 y 3, pero debe tener la confirmación hasta que la trama 4 llegara. En este momento, de le enviaría al emisor **A** él acuse de recibo con número de secuencias 5. De este modo, **B** confirma simultáneamente las tramas 1, 2, 3 y 4.

Este procedimiento obliga a que el emisor **A** mantenga un listado de los números de secuencia de tramas que se le permite transmitir, en tanto que el receptor **B** debe controlar otro listado con los números de secuencia de tramas que espera recibir. Cada una de estas listas se denomina como ***ventana de tramas*.** De allí que este procedimiento se nombre como ***Control De Flujo Mediante Ventana Deslizante*** (sliding-window flow control).

Este procedimiento se plantea asumiendo que dentro de cada una de las ***PDU's*** se encuentra un campo que contiene la numeración secuencial previamente señalada, y que evidentemente tendrá un tamaño limitado a cierta cantidad de bits. Así por ejemplo, si dicho campo tuviera una amplitud de 4 bits, los números de secuencia podrían variar entre 0 y 15. En consecuencia, las tramas se deberán enumerar basándose en un esquema módulo 16, es decir, después del número 15 seguiría el 0.

En el caso más general, cuando el campo de secuencia posea k bits, el rango de números de secuencia se extenderá entre 0 y 2^{k-1}, con las tramas numeradas en módulo 2^k. La ilustración que se muestra a continuación, permite observar el comportamiento general de este procedimiento.

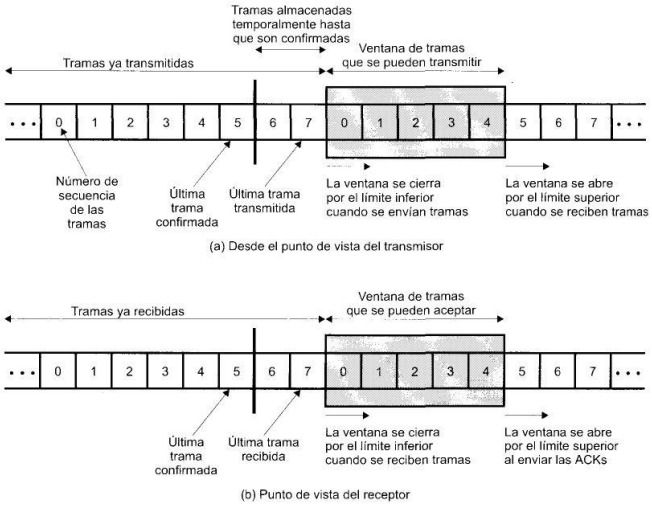

Ilustración 2.8: Control De Flujo Mediante Ventanas Deslizantes
Fuente: [STAL04]

Por otro lado, la siguiente ilustración presenta la aplicación del procedimiento de ventanas deslizantes al comunicarse dos entidades **A** y **B**, empleando un campo de secuenciación de 3 bits y un tamaño máximo de ventana igual a siete tramas.

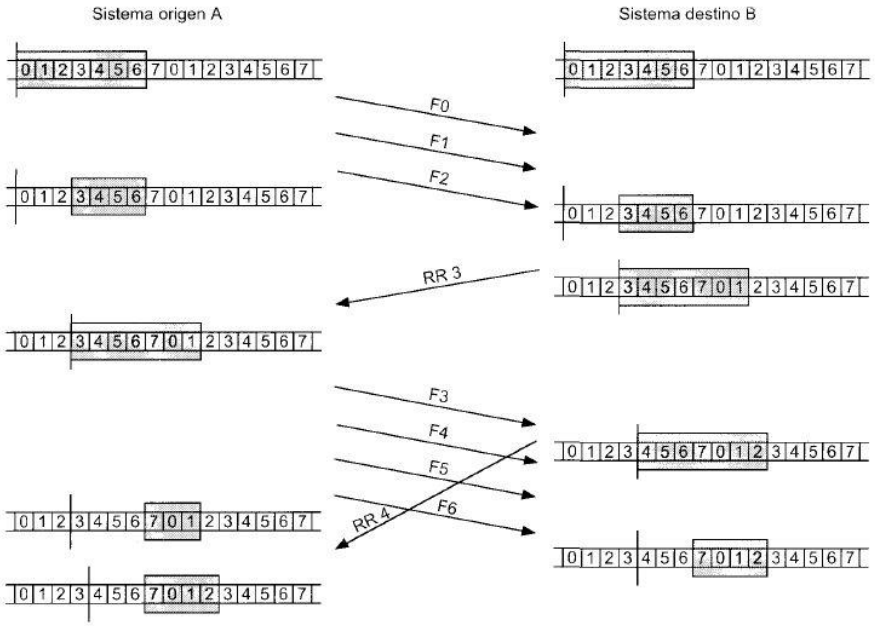

Ilustración 2.9: Ejemplo De Control De Flujo Mediante Ventanas Deslizantes
Fuente: [STAL04]

En este ejemplo, los mensajes señalados como $\mathbf{F}_i$[6] (i = 0,.., 6) señala las tramas enviadas, en tanto que los mensajes identificados como $\mathbf{RR}_j$[7] (j = 3,4), identifican los momentos en que el receptor está preparado para recibir más mensajes al mismo tiempo que confirma las tramas previamente recibidas.

2.4.6 Control De Errores.

Consiste en una función que permite esencialmente, identificar y corregir ***PDU's*** que contengan información incorrecta. Generalmente se implementa mediante dos funciones separadas:

- **La detección de errores:** Para implementar esta función, la entidad emisora del mensaje inserta en cada ***PDU*** un código -función estricta de los bits que constituyen a la ***PDU***- que permita identificar la presencia de errores en la transmisión. Por otro lado, la entidad receptora calculará nuevamente dicho valor y lo comparará contra el recibido. Si ambos valores coinciden, el paquete se considera válido y la entidad receptora informa a la entidad emisora su recepción exitosa. Por otro lado, en caso que se detecte un paquete dañado, el receptor simplemente lo descartará.

- **La retransmisión de paquetes dañados:** cuando la entidad receptora recibe un mensaje erróneo, al descartarlo, no devolverá un mensaje de validación de recepción. Si la entidad emisora del ***PDU*** no recibe este

---

6 **Frame**, entendido como **marco** o **trama**.

7 **Receive Ready**, se debe interpretar como **preparado para recepciones**.

aviso de confirmación dentro de un intervalo de tiempo razonable, retransmitirá nuevamente el paquete, asumiendo que no llegó o llegó dañado.

- **Corrección de errores:** Algunos protocolos muy especializados, con base en los códigos de comprobación de ***PDU's*** pueden, dentro de ciertas limitaciones, corregir los errores que contiene el paquete dañado, evitando en, algunos casos, la necesidad de retransmitirlo. Esta no es una característica común dentro de los protocolos de comunicaciones.

De la misma forma que en la función de control de flujo, el control de errores es otra función que debe ser realizada en varios niveles de las arquitecturas de protocolos. Por ejemplo, al considerar la Ilustración 2.6, se puede observar que, en los niveles de acceso a la red y transporte, es importante que exista la posibilidad de identificar y retransmitir paquetes erróneos.

2.4.7 Multiplexación.

Esta función consiste en un conjunto de técnicas que permiten la transmisión simultánea de múltiples señales a través de un único enlace de datos. Se asemeja a los sistemas de acueducto actuales, donde una gran tubería principal alimentadora de agua, suministra el servicio, al mismo tiempo, a varios locales.

Esta situación se origina por el incremento en el uso de los recursos de comunicaciones, que incrementan proporcionalmente el tráfico sobre los mismos en la medida en que crece el número de usuarios conectados a los servicios.

La solución de este problema utilizando líneas individuales es bajo todo punto de vista, ineficiente, excesivamente costosa, además de que complica innecesariamente los sistemas de telecomunicaciones.

En tal sentido, la multiplexación ofrece un sistema que permite maximizar la eficiencia de los medios de transmisión, de manera que con una infraestructura mínima, se puede maximizar uso.

Cuando se implementa la multiplexación, generalmente se requiere que se aplique conjuntamente con técnicas de control de congestión de tráfico, a fin de evitar que los servicios colapsen por el exceso de demanda.

La multiplexación admite dos formas básicas de implementación, así como una trivial:

- **Multiplexación uno a uno:** En este caso simplemente, una conexión de nivel superior se corresponde exactamente a una conexión del nivel inferior y la multiplexación como tal simplemente no se presenta.
- **Multiplexación ascendente -o hacia adentro-:** Bajo este esquema, varias conexiones de nivel superior comparten o se multiplexan han sobre una única conexión de nivel inferior. Esta técnica puede ser útil cuando se desea utilizar eficazmente los servicios de nivel inferior o para proporcionar varias conexiones del nivel superior en un entorno donde existe una única conexión de nivel inferior. En lo que se conoce como medio de transmisión con acceso compartido.
- **Multiplexación descendente -o de división-:** En este caso, se establece una única conexión de nivel superior, utilizando varias conexiones de nivel inferior. Es decir, todo el tráfico de la conexión superior será dividido entre varias conexiones de nivel inferior. Esta

técnica se suele utilizar para agregar los niveles de seguridad a la conexión -ya que ofrece redundancia en el canal-, así como eficiencia en la transferencia de grandes cantidades de información.

La ilustración que se muestra a continuación, permite apreciar gráficamente las tres formas de multiplexación antes mencionadas:

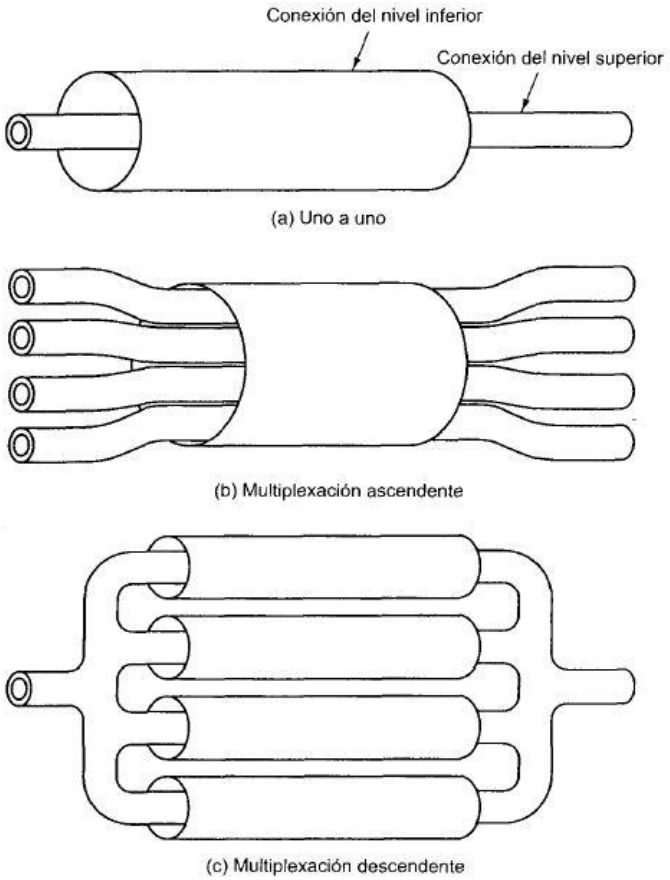

Ilustración 2.10: Esquemas De Multiplexación
Fuente: [STAL04]

La multiplexación se puede aplicar en otros contextos distintos, en combinación con las funciones de direccionamiento y encaminamiento que serán consideradas a continuación.

2.4.8 Direccionamiento.

El direccionamiento es una de las funciones más complejas y abarcadoras dentro de la funciones de los protocolos de comunicaciones. Está relacionada a las funciones de encaminamiento y multiplexación.

La función de direccionamiento consiste, esencialmente, en identificar, de manera inequívoca, a las entidades emisora y receptora de los mensajes; de manera tal que las ***PDU's*** enviadas lleguen a la entidad destino y sólo ella.

Las explicaciones subsiguientes de esta sección se realizarán con base en la ilustración que se muestra a continuación[8].

---

[8] En dicha ilustración, se expone un modelo que esencialmente representa a la arquitectura ***TCP/IP***, pero que sin embargo, es aplicable a prácticamente cualquier otra arquitectura similar.

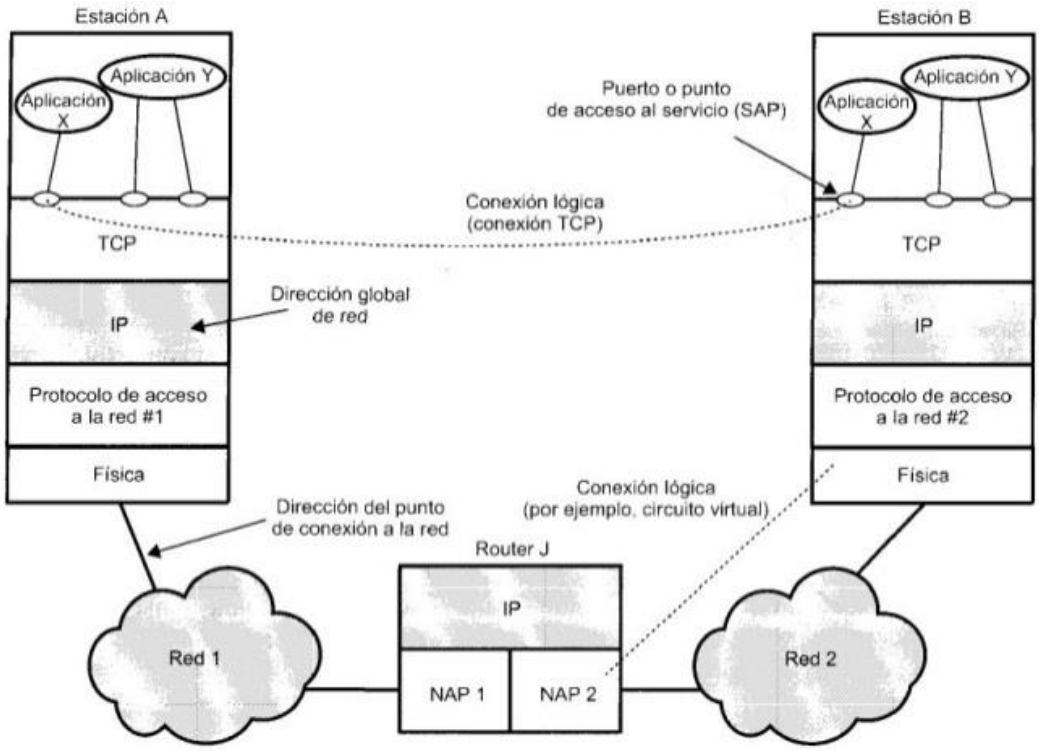

Ilustración 2.11: Conceptos De Direccionamiento
Fuente: [STAL04]

Dentro de las arquitecturas de protocolos, el concepto de direccionamiento es complicado, ya que abarca una serie de aspectos tales como:

2.4.8.1 El Nivel Del Direccionamiento.

Cuando se analiza el nivel de direccionamiento, se está haciendo referencia al nivel que identifica a la entidad, dentro de la arquitectura de comunicaciones bajo estudio.

Es así como por ejemplo bajo la arquitectura ***TCP/IP***, los equipos -ya sean computadoras o equipos de intercomunicación- se identifican por las llamadas direcciones ***IP***, en tanto que bajo la arquitectura ***OSI*** esta misma direcciones se denominan puntos de acceso al servicio de red (***NSAP***, Network Service Access Point), en la llamada capa de red. Estas direcciones se utilizarán para

encaminar las ***PDU's*** a través de la red o redes que comunican a los sistemas emisor y receptor.

Por otro lado, en la capa de aplicación, cada entidad que participa en el proceso de comunicación deberá también tener un identificador único, entendiendo entidad como:

- Aplicaciones individuales que usan los servicios.
- Múltiples usuarios que utilizan una misma aplicación.
- Múltiples procesos de un usuario dentro de una aplicación.

Para este nivel, ***TCP/IP*** define estos identificadores como puertos, en tanto que ***OSI*** los llama puntos de acceso al servicio (***SAP***, Service Access Point). En la Ilustración 2.11, se aprecian dos niveles direccionamiento dentro del sistema modelado, sin embargo puede darse el caso en que cada capa de la arquitectura tenga un único ***SAP***.

2.4.8.2 El Alcance Del Direccionamiento.

Otro tema relacionado es del alcance de direccionamiento, entendiendo este concepto como el ámbito donde la dirección establecida tiene vigencia.

En tal sentido, se conocen direcciones tanto locales como globales. Las direcciones locales, como su propio nombre lo dicen, tienen vigencia únicamente en un entorno limitado, constituido por algunas entidades. Su característica fundamental es que entre los equipos que integran su entorno, dichas direcciones son visibles; en tanto que desde ambientes externos son invisibles.

Por otro lado, las direcciones globales, tienen como características principales:

- **No ambigüedad global:** Es decir, que una dirección global identifica a un solo sistema. Sin embargo para un sistema dado, puede haber más de una dirección global.

- **Aplicabilidad global:** Establece que desde cualquier sistema se podrá identificar a cualquier otro sistema, utilizando su dirección global.

Estas dos características son las que permiten que, bajo Internet, se puedan intercambiar datos entre cualesquiera equipos que tengan direcciones globales -las llamadas ***IP's*** públicas-.

Es relevante señalar que las direcciones que se han comentado hasta este momento, pueden ser categorizadas como de tipo ***lógico***, porque se establecen en los niveles de software, por encima de las capas físicas.

Adicionalmente a ellas, también se requieren direcciones de tipo ***físico***, para identificar a nivel de hardware, a cada uno de los interfaces que contiene cada dispositivo de la red. Como ejemplo de este tipo de direccionamiento se encuentran las direcciones ***MAC*** de las redes ***IEEE 802***, así como las direcciones de las estaciones de red ***X.25***, que permiten que las capas de red encaminen unidades de datos tales como tramas ***MAC*** o paquetes ***X.25***. Estos tipos de direcciones se conocen como direcciones de punto de conexión en la red.

Ahora, el alcance del direccionamiento es una conceptualización que sólo tiene relevancia en las capas de arquitectura de red o inferiores -bajo el modelo ***OSI***-. Por encima de este nivel, un puerto o dirección ***SAP*** debe ser único dentro de la entidad local, sin que se requiera que lo sea a nivel global. Por ejemplo, en la Ilustración 2.11, tanto en la estación **A** como en la **B**, puede

existir perfectamente un puerto 1, que pueden ser diferenciados de manera única como **A.1** y **B.1**, respectivamente.

2.4.8.3 Los Identificadores De La Conexión.

Consisten en la asignación de un nombre único que identifica a la conexión durante todo el proceso de transmisión de datos.

Este concepto tiene sentido exclusivamente cuando se plantea sobre la base de que las transferencias de datos se realizan en un entorno orientado a conexiones -tal como los circuitos virtuales, por ejemplo-. Por otro lado, en las transferencias no orientadas a conexión -como los datagramas-, se debe utilizar un nombre global para cada transmisión.

El empleo de los identificadores de conexión ofrece algunas ventajas, tales como:

- **Reducción en el tamaño de las cabeceras:** Por lo general, los identificadores de conexión son más cortos que los identificadores globales.

- **Encaminamiento:** Ya que los identificadores de conexión, al operar sobre circuitos virtuales, actúan sobre una ruta fija, que permite identificar los caminos que seguirán las ***PDU's*** futuras -el concepto de encaminamiento será ampliado posteriormente-.

- **Multiplexación:** Los identificadores de conexión permitirán que una entidad pueda utilizar, simultáneamente, más de una conexión -al aplicar multiplexación descendente-, por lo que las ***PDU's*** se deben reconocer vía el identificador de conexión. Esto permitirá explotar en todo su potencial a la capacidad de transmisión de los medios actuales.

- **Uso de información de estado:** Establecida la conexión, vía los identificadores de conexión, los sistemas finales de usuario podrán tener acceso a la información de estado relativa a la conexión. Esta facilidad posibilita que se implementen funciones tales como el control de flujo o de errores, mediante la numeración secuencial de las ***PDU's***.

2.4.8.4 El Modo De Direccionamiento.

Consiste en la forma en que se en que se identifica a las entidades. Como relación puede ser planteada de las siguientes formas:

- **Unidestino (Unicast):** Se presenta cuando el direccionamiento y las entidades se relacionan de ***uno en uno***. Es la que se utiliza en el direccionamiento convencional dentro de las redes de datos, para identificar a cada sistema con una única dirección o puerto.

- **Multidestino (Multicast):** Se observa cuando el direccionamiento y las entidades de relacionan de ***uno a varios***. Este es el caso, en el que, por ejemplo, para clonar toda la información que se encuentra en un computador, hacia un grupo de equipos que se desea, sean clones idénticos de la máquina base, mientras que no se altera a otros equipos dentro de la red.

- **Difusión (Broadcast):** Se evidencia cuando el direccionamiento y las entidades se relacionan de ***uno a todos***. Es la que se utiliza, por ejemplo, cuando un administrador, dentro de un dominio, desea enviar un mensaje a todos los sistemas del dominio, para informarles que el servidor central será detenido en cierto tiempo.

En la tabla que se presenta a continuación, se ilustran todas las posibilidades de modos de direccionamiento.

Destino	Dirección De Red	Dirección Del Sistema	Dirección Del Puerto / SAP
Unidestino	Individual	Individual	Individual
Multidestino	Individual Individual Todos	Individual Todos Todos	Grupo Grupo Grupo
Difusión	Individual Individual Todos	Individual Todos Todos	Todos Todos Todos

Tabla 2.1: Modos De Direccionamiento
Fuente: [STAL04]

2.4.9 Encaminamiento.

Por otro lado, la función de encaminamiento esencialmente consiste en que bajo el supuesto que exista un sistema de rutas alternas al comunicar a la entidad receptora con la emisora, siempre se ***escoja un recorrido*** que permita alcanzar efectivamente al destinatario del mensaje.

El problema surge al momento en que las redes crecen de manera significativa, con la adición de rutas redundantes entre los distintos destinos, algunas más eficientes que otras; algunas seguras y otras con riesgos significativos de pérdida de mensajes.

En este escenario, es importante que los sistemas de comunicaciones posean cierto nivel de independencia al momento de establecer la ruta que seguirán los mensajes. Para tal efecto, existen equipos especiales en las redes que se conocen como encaminadores, enrutadores o routers, que se encargan de esta tarea **[FORO07]**.

Generalmente, estos dispositivos permiten que las redes operen con base a dos criterios básicos, que no siempre se complementan:

- **Eficiencia:** Desde el punto de vista de las empresas que administran las redes de datos, es deseable que se minimice la cantidad de equipos que integran a la red, bajo el supuesto que ella sea capaz de gestionar toda la carga de trabajo que se espera que soporte. Estas necesidades se expresan generalmente en términos del ***tráfico en horas punta***. Es decir, la carga promedio que se espera durante los períodos de mayor actividad a lo largo del día. Esto, con el fin de minimizar los costos de operación de la red.

- **Flexibilidad:** Por otro lado, es posible que en cualquier momento -no necesariamente en las horas punta-, la red sea sometida a una carga que exceda sus capacidades reales- por ejemplo en medio de una tormenta eléctrica-. Bajo estas condiciones, sería deseable que la red tuviera un buen nivel de tolerancia a fallos y redundancia, que garanticen un nivel de servicio mínimo, que obviamente, incrementará sus costos de operación.

El encaminamiento, es precisamente, la funcionalidad que permite lograr el compromiso entre la eficiencia y la flexibilidad. Aunque la infraestructura de comunicaciones -entiéndase medios de transmisión-, es más o menos estática y jerárquica, se complementa con un sistema dinámico de encaminamiento. Bajo este esquema, la ruta que toman los paquetes de datos se establece con base en el análisis de las condiciones instantáneas de tráfico de la red.

En todo caso, la función de encaminamiento es una de las más complicadas y dentro del entorno de la arquitectura de protocolos, dadas las múltiples variaciones de tráfico y capacidad de medios de transmisión que se pueden presentar y variar rápidamente de condición, con la correspondiente respuesta del sistema de encaminadores **[KURO07]**. Por esta razón, generalmente se implementa esta funcionalidad en los niveles inferiores de las pilas de

protocolos, a fin de maximizar la capacidad de respuesta de los sistemas **[FORO07]**.

2.4.10 Servicios De Transmisión.

Además de todas las prestaciones antes descritas, los protocolos también pueden ofrecer una serie de servicios adicionales a las entidades que los requieran, tales como **[KURO07]**:

- **Prioridad en el envío de los paquetes:** los protocolos deben poder ofrecer la posibilidad de incrementar o disminuir el tiempo de espera que deben enfrentar los paquetes al momento de su entrega, de acuerdo al tipo de servicio o de conexión. Es decir, ciertos tipos de o paquetes, como el correo electrónico, pueden admitir un retardo relativamente elevado sin que eso represente un desmejoramiento en la calidad del servicio -es difícil que alguna persona se moleste por recibir un correo un minuto después que se envió, cuando esperaba recibirlo un segundo después de enviado-. Por otro lado, paquetes que involucren vídeo en tiempo real no pueden admitir ningún tipo de retardo en su entrega.

- **Calidad de servicio:** se relaciona con el aspecto anterior, ya que dependiendo del tipo de paquete o servicio, puede darse el caso que se requiera garantizar ciertos niveles mínimos de la eficiencia en la entrega del paquete, tales como minimizar o eliminar la cantidad de paquetes que requieran ser retransmitidos por dañarse durante el envío.

- **Seguridad:** bajo este aspecto, se considera la posibilidad de controlar el acceso a la información que contienen los paquetes, por parte de entidades no autorizadas.

Estos y otros servicios que ofrezcan las redes de comunicaciones, dependerán de los sistemas de transmisión subyacentes, así como de las facilidades que se brinden en los niveles inferiores de la arquitectura de protocolos.

Si los niveles inferiores ofrecen estos servicios, las entidades superiores pueden utilizarlos con sólo invocar al protocolo correspondiente.

Por otro, lado, si los niveles inferiores no ofrecen estos servicios, será más difícil implementarlos en los niveles superiores de arquitectura, ya que se deben implementar desde cero.

2.5 CLASIFICACIÓN DE LOS ESTÁNDARES DE COMUNICACIONES DE DATOS.

En el área de las comunicaciones de datos, se requieren de grandes esfuerzos para lograr la coordinación fluida y eficiente de los sistemas que las componen, sobre todo cuando son creados por fabricantes diferentes.

De allí nace la necesidad de los estándares, que son esenciales para crear y mantener un sistema abierto y competitivo, de manera que se garantice la interoperatividad en los datos, tecnologías y procesos de telecomunicaciones.

La ilustración que se muestra a continuación, muestra la clasificación general de los estándares.

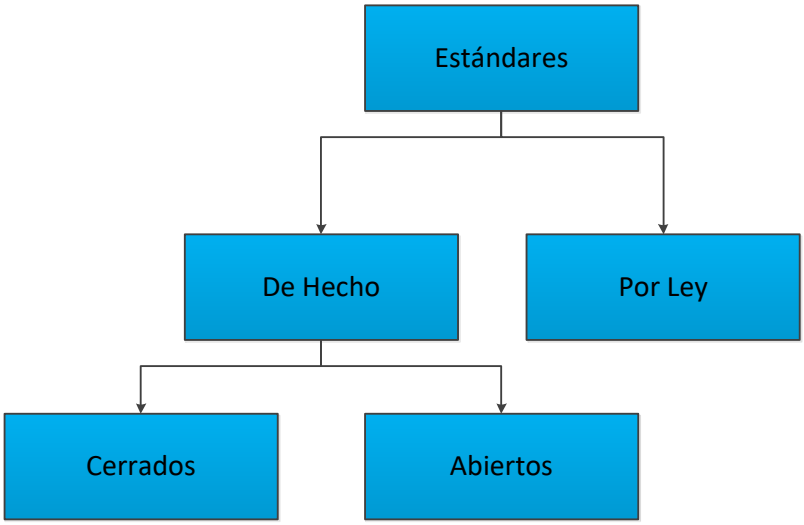

Ilustración 2.12: Clasificación De Los Estándares
Fuente: [FORO07]

Los estándares de hecho, son aquellos que han sido adoptados por su amplia difusión dentro de la comunidad tecnológica, científica e industrial. En tanto, los estándares por ley son aquellos legislados por organizaciones de estandarización mundialmente reconocidas.

Por otro lado, los estándares cerrados, son aquellos que adoptan las empresas y los que cobran regalías cuando otros fabricantes los tratan de aplicar, en tanto que los estándares abiertos, son aquellos que se divulgan al dominio público y cualquiera los puede aplicar en sus desarrollos.

El pragmatismo actual y la presión de los consumidores ha forzado a la industria tecnológica a reconocer la necesidad de modelos generales y hay un acuerdo global de lo que deben ser estos modelos.

La inteligencia y la previsión de los diseñadores parecen ser tales que los estándares que están siendo adoptados actualmente facilitarán más que retrasarán el desarrollo técnico. No obstante, de vez en cuando algunas empresas tratan de imponer modelos cerrados que les permitan maximizar sus ganancias.

2.6 ORGANIZACIONES QUE ADMINISTRAN LAS NORMALIZACIONES EN LOS PROTOCOLOS DE COMUNICACIONES.

Los estándares en el área de comunicaciones son desarrollados mediante la cooperación de comités foros y agencias reguladoras de los gobiernos. A nivel norteamericano, y de hecho mundial, las organizaciones que son reconocidas al momento de publicar estas normas son **[FORO07]**:

2.6.1 ISO.

La ***International Standards Organization*** (***ISO***; también denominado como Organización Internacional para la Estandarización) es un organismo multinacional cuyos miembros provienen fundamentalmente de los comités de creación de estándares de varios gobiernos a lo largo del mundo.

Creado en 1947, ***ISO*** es una organización totalmente voluntaria dedicada a establecer acuerdos mundiales sobre estándares internacionales.

Con un número de miembros que actualmente incluye cuerpos representativos de 82 naciones industrializadas, su objetivo es facilitar el intercambio internacional de productos y servicios, proporcionando modelos de compatibilidad, mejoras de calidad, mejoras de productividad y precios más baratos.

En el área de las telecomunicaciones, sus esfuerzos han resultado en la creación del modelo de Interconexión de Sistemas Abiertos (***OSI***).

2.6.2 ITU-T.

A principios de la década de los 70, un cierto número de países estaba definiendo estándares nacionales para telecomunicaciones, pero a pesar de ello seguía habiendo muy poca compatibilidad internacional.

Las Naciones Unidas respondieron a este problema formando, como parle de su Unión Internacional de Telecomunicaciones (***ITU***), un comité, denominado Comité Consultivo para la Telefonía y la Telegrafía Internacional (***CCITT***).

Este comité estaba dedicado al desarrollo y establecimiento de estándares para telecomunicaciones en general y para la telefonía y los sistemas de datos en particular.

El 1 de marzo de 1993, el nombre de este comité se cambió a Unión Internacional de Telecomunicaciones-Sector de Estándares de Telecomunicaciones (***ITU-T***).

Los estándares mejor conocidos de la ***ITU-T*** son las series ***V (V.32, V.33, V.42)*** que definen la transmisión de datos a través de líneas telefónicas; la serie ***X (X.25, X.400, X.500)*** que define la transmisión de datos a través de redes digitales públicas; correo electrónico, servicios de directorios y la Red Digital de Servicios Integrados (***RDSI***), que incluyen parte de las otras series y definen la emergente red digital internacional.

Los productos actuales incluyen una ampliación de ***RDSI*** llamada ***RDSI de banda ancha***, conocida popularmente como la Autopista de la Información.

2.6.3 ANSI.

A pesar de su nombre, el Instituto Nacional Americano para la Estandarización (***ANSI***) es una corporación completamente privada sin ánimo de lucro que no tiene ninguna relación con el gobierno federal de los Estados Unidos. Sin embargo, todas las actividades de ***ANSI*** están orientadas hacia el desarrollo de los Estados Unidos, y sus ciudadanos tienen una importancia primordial.

Los objetivos expresados por ***ANSI*** incluyen servir como una institución de coordinación nacional para la estandarización voluntaria dentro de los Estados Unidos, persiguiendo que la adopción de los estándares permita hacer avanzar la economía de los Estados Unidos y asegurar la participación y la protección del interés público.

Los miembros de ***ANSI*** son sociedades profesionales, asociaciones de la industria, agencias gubernamentales y reguladoras y grupos de consumidores.

Sus temas actuales de discusión incluyen: planificación e ingeniería de interconexión de redes; servicios, señalización y arquitecturas ***RDSI***; y jerarquía óptica (***SONET***).

2.6.4 IEEE.

El Instituto de Ingenieros Eléctricos y Electrónicos (***IEEE***, Institute of Electrical and Electronics Engineers) es la mayor sociedad profesional de ingeniería del mundo. De ámbito internacional, sus objetivos son el desarrollo de la teoría, la creatividad y la calidad de los productos en el campo de la ingeniería eléctrica, la electrónica y la radio, así como otras ramas relacionadas de la ingeniería.

Como uno de sus objetivos principales, el ***IEEE*** prevé el desarrollo y adopción de estándares internacionales para computación y comunicación. El ***IEEE*** tiene un comité especial para las redes de área local (***LAN***), del cual ha surgido el ***Proyecto 802*** -por ejemplo, los estándares ***802.3*** (***Ethernet***), ***802.4*** (***Token Bus***) y ***802.5*** (***Token Ring***)-.

2.6.5 EIA.

En la línea de ***ANSI***, la Asociación de Industrias Electrónicas (***EIA***) es una organización sin ánimo de lucro dedicada a la promoción de aspectos de la fabricación electrónica. Sus objetivos incluyen despertar el interés de la educación pública y hacer esfuerzos para el desarrollo de los estándares.

En el campo de la tecnología de la información, la ***EIA*** ha hecho contribuciones significativas mediante la definición de interfaces de conexión física y de especificaciones de señalización eléctrica para la comunicación de datos. En particular, el ***EIA-232-D, EIA-449 y EIA-530***, que definen la transmisión serie entre dos dispositivos digitales (por ejemplo, computadora a módem).

2.6.6 FCC.

La Comisión Federal de Comunicaciones (***FCC***) es una agencia federal del gobierno de los Estados Unidos De América, que tiene autoridad sobre el comercio interestatal e internacional en lo que se refiere a las comunicaciones.

Cada producto de las tecnologías de las telecomunicaciones debe tener una aprobación del ***FCC*** antes de que pueda ser vendido en los ***EUA*** (se puede comprobar en la parte de abajo de un teléfono y se verá que hay un código de

aprobación de este organismo regulador). Las responsabilidades específicas del ***FCC*** incluyen:

- Comprobación de las revisiones y las aplicaciones de las tarifas hechas por los proveedores de telegrafía y telefonía.
- Revisión de las especificaciones técnicas del hardware de telecomunicaciones.
- Establecimiento de tasas de retorno razonables para portadores comunes.
- División y asignación de las frecuencias de radio.
- Asignación ele las frecuencias portadoras para las emisiones de radio y televisión.

2.7 PRINCIPIOS QUE DEBEN SEGUIRSE AL MOMENTO DE PLANTEAR UN PROTOCOLO DE COMUNICACIONES.

En primer plano, la discusión que se planteará en adelante, se enfoca en el comportamiento y estructura del modelo ***OSI***, pero es, en general, válida para cualquier diseño de protocolo de comunicación moderno.

La ***ISO*** utiliza la jerarquización por capas para definir a los protocolos de comunicaciones.

La relevancia de la ***ISO*** como organización internacional dedicada a la normalización, conlleva que sus recomendaciones tengan un carácter mandatorio **[STAL04]**.

En tal sentido la especificación ***ISO*** consistió, simplemente, en establecer un grupo de capas conceptualmente próximas, cada una de complejidad relativamente reducida; así como los servicios que debe ofrecer cada una de ellas.

Este conjunto de normas es lo que se conoce universalmente en la telemática como el Modelo ***OSI*** **[TANE03]**, y que se asume conocido dentro del marco teórico de esta investigación.

Bajo estos principios, todas las funciones de comunicación de las arquitecturas de protocolos de comunicaciones se deben establecer en función de capas jerárquicas **[FORO07]**.

Cada capa realiza un conjunto limitado de funciones interrelacionadas, necesarias para comunicarse con otros sistemas[9].

Así mismo, cada capa ofrece a su par superior, los servicios que ella requiera, ocultándole los detalles de implementación; en tanto que las ubicadas en los niveles inferiores, le ofrecen el acceso a funciones más básicas y primitivas.

Este modelo de diseño tiene la intención de que los cambios que se pueden dar a lo interno de una capa no impliquen cambios en las otras capas (modularidad y encapsulamiento).

---

[9] Estos mecanismos de comunicación entre capas se conocen como interfaces **[FORO07]**.

A continuación, se establecen los principios generales que se siguieron al momento de establecer las capas del modelo ***OSI***, y que de hecho, norman el diseño de protocolos en general:

#	**Principio**
1	No crear demasiadas capas de forma que la descripción e integración de las capas sea más difícil de lo estrictamente necesario.
2	Definir separaciones entre capas tal que la descripción de servicios sea pequeña y el número de interacciones entre capas sea mínimo.
3	Definir capas separadas para funciones que sean claramente diferentes, en lo que respecta al servicio ofrecido como a la tecnología implicada.
4	Definir funciones similares en la misma capa.
5	Seleccionar los límites o separación entre capas de acuerdo con lo que la experiencia previa aconseje.
6	Definir las capas tal que las funciones se puedan localizar fácilmente de forma que la capa se pueda rediseñar completamente y tal que sus protocolos se puedan modificar para adaptarse a las innovaciones en la arquitectura, la tecnología hardware o en el software sin necesidad de cambiar los servicios que .se usan o proporcionan en las capas adyacentes.
7	Definir una separación entre capas. allí donde pueda ser útil tener la interfaz correspondiente normalizada.
8	Crear una capa donde exista la necesidad de un nivel diferente de abstracción en el procesamiento de los datos (por ejemplo, morfológico, sintáctico, semántico).
9	Permitir modificaciones de funciones o protocolos dentro de una capa, siempre que no afecten a otras capas.
10	Crear para cada capa límites o separaciones sólo con su capa superior e inferior. Principios similares han sido aplicados para la creación de subcapas.
11	Crear subgrupos y organizaciones adicionales de funciones en subcapas dentro de una capa sólo en los casos donde se necesiten servicios distintos de comunicación.
12	Crear, donde sea necesario, dos o más subcapas con una funcionalidad común y por lo tanto mínima para permitir la operación de la interfaz con capas adyacentes.
13	Permitir la no utilización de todas las subcapas.

Tabla 2.2: Principios Utilizados En La Definición De Las Capas OSI (ISO 7498)
Fuente: [STAL04]

2.7.1 Funcionamiento De Las PDU's Dentro De Las Arquitecturas De Protocolos.

Desde un punto de vista meramente conceptual, el funcionamiento general de las ***PDU's*** dentro del modelo ***OSI***, es bastante simple. El mismo, puede ser analizado utilizando como base a la ilustración que se puede observar a continuación **[STAL04]**:

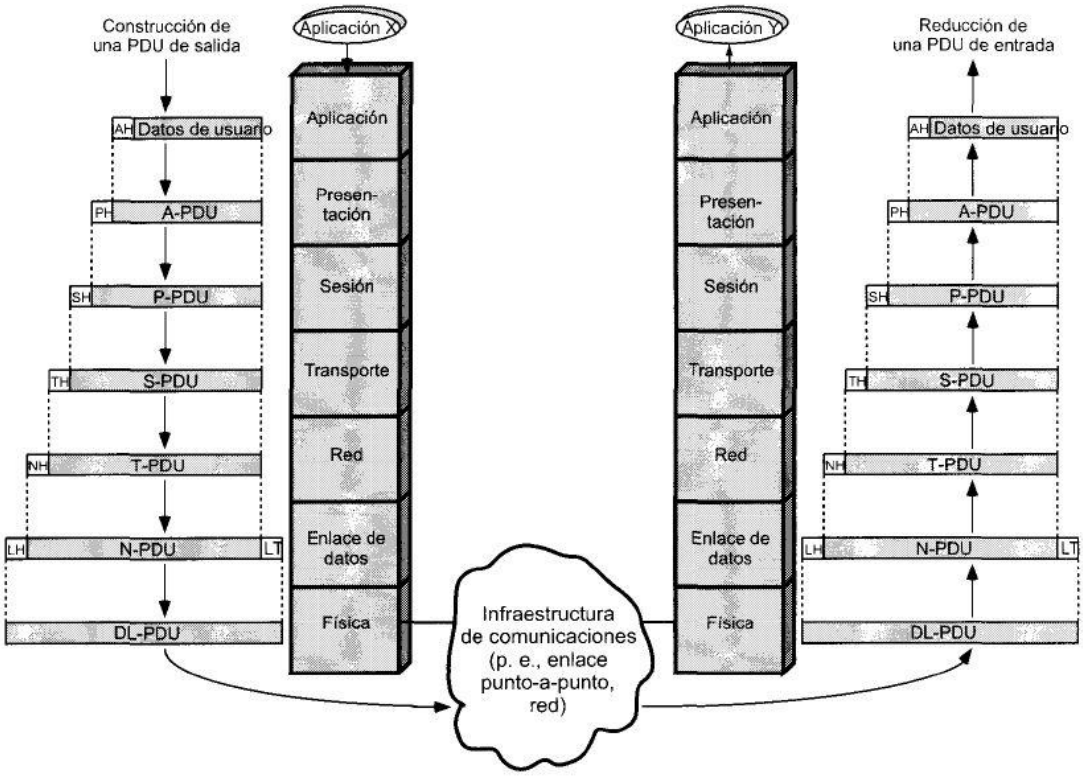

Ilustración 2.13: Administración De PDU's En El Modelo OSI
Fuente: [STAL04]

Esencialmente, se observa que al tratar de comunicar a la aplicación **X** con la aplicación **Y**, vía la red de computadoras, se establece una relación de igual a igual entre dichas aplicaciones.

Luego, los datos del usuario en la aplicación **X** van descendiendo paulatinamente a través de todas las capas del modelo. En cada descenso, se toman los datos que ofreció la capa anterior, se encapsulan y segmentan de acuerdo al protocolo(s) que se manejan en cada capa, y se le agregan ***encabezados*** sucesivos -identificados como **{A, P, S, T, R, L[10]}H**-, en tanto que en la capa física se agrega un ***trailer*** -identificados com **LT**-, de manera que se crea un nuevo ***PDU*** bajo las reglas del correspondiente protocolo.

Es al llegar a la capa física que se transmiten los bits vía el medio de transmisión. Posteriormente, al completarse la transferencia bajo el nivel físico, las ***PDU's*** son sucesivamente desempaquetadas y ensambladas, hasta que el mensaje transmitido alcanza a la aplicación **Y**.

Es relevante señalar que, salvo en el nivel físico, las capas equivalentes generalmente no se comunican directamente. Dicho de otra forma, en cada entidad de protocolo diferente de la física, se realiza la transferencia de los paquetes de datos, de manera descendente primero, luego cambian de entidad a nivel de la capa física, para luego ascender hasta alcanzar a su entidad par, tal como se puede apreciar en la Ilustración 2.13.

---

[10] (A)plicación, (P)resentación, (S)esión, (T)ransporte, (R)ed, (L)ink – o enlace de datos-.

2.7.2 El Modelo OSI Como Referencia Para La Normalización De Protocolos De Comunicación.

La arquitectura modular -es decir, la descomposición del problema global en problemas más pequeños- que se observa en el modelo ***OSI*****,** permite que el diseño e implementación de protocolos, dentro de él, resulte bastante eficiente, ya que **[STAL04]**:

- Dado que las capas que integran al modelo se encuentran claramente delimitadas y no son muy abarcadoras, permiten que se puedan diseñar e implementar estándares dentro de ellas de manera paralela e independiente, lo que permite acelerar los procesos de actualización del modelo.

- Por otro lado, como las fronteras entre las capas se encuentran bien definidas, se pueden realizar cambios en los estándares de una capa, sin afectar a las adyacentes, permitiendo así que sea sencilla la inclusión de normalizaciones adicionales.

- Adicionalmente, la modularidad permite que se implemente el principio de ocultamiento de información, de manera tal que las capas superiores son no requieren -y de hecho, no deben- conocer los detalles de implementación que requieren las capas inferiores.

- A lo interno de cada capa, se ofrecen una serie de servicios a la capa superior adyacente; además que implementa el protocolo con la capa par en el sistema remoto.

En la siguiente ilustración, se puede observar de manera más clara la forma en que se debe establecer la normalización dentro de cada capa.

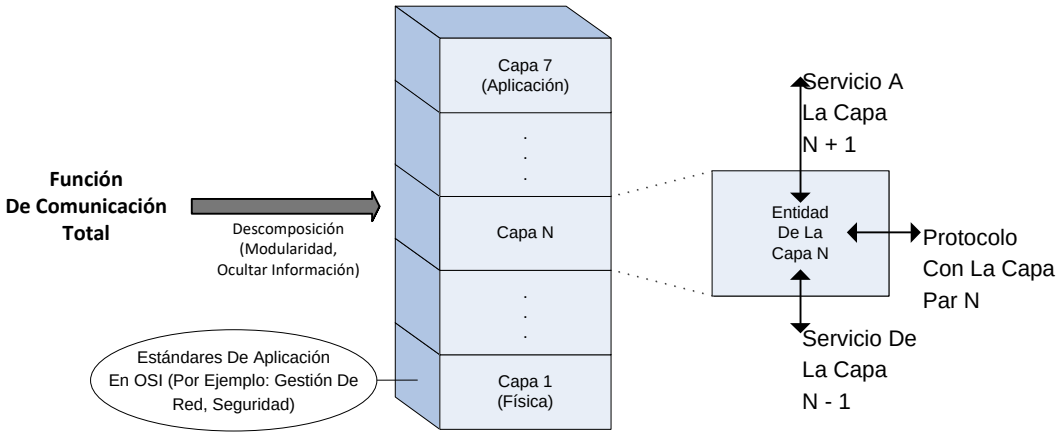

Ilustración 2.14: Normalización Entre Las Capas Del Modelo OSI
Fuente: [STAL04]

Analizando la situación con más cuidado, se puede observar que se presentan tres elementos clave dentro la normalización de cada capa:

- **Especificación del protocolo:** cada pareja de entidades ubicadas dentro de la misma capa pero en sistemas distintos[11] deben coordinar sus acciones a través del protocolo. Bajo este escenario, el protocolo debe establecer explícitamente tanto el formado de las ***PDU's***, como su semántica y secuenciación, ya que se están comunicando dos sistemas completamente abiertos[12].

- **Definición del servicio:** Adicionalmente a contar con un conjunto de protocolos que funcionan dentro de una capa dada de la arquitectura, son necesarias normalizaciones adicionales que puedan suministrar los

---

11 Se les denomina procesos paritarios **[FORO07]**.

12 Por ejemplo, podría ser un sistema PC basado en el sistema operativo Windows y un procesador Intel, que establece comunicación con un sistema Oracle-Sun, basado en Solaris y SPARC, respectivamente.

servicios que requieren la capa superior inmediatamente adyacente. Generalmente, la definición de servicios consiste en una definición de tipo funcional, que establece una declaración formal del servicio que se ofrece -en términos de tipos de datos requeridos como parámetros, y los tipos de datos retornados-, pero sin aclarar sus detalles de implementación -es decir, no se establecen los procedimientos algorítmicos requeridos para ofrecer el servicio-. Este enfoque favorece la implementación local de sistemas eficientes, sin comprometer la interoperatividad de los sistemas abiertos.

- **Direccionamiento:** Bajo el modelo ***OSI***, cada una de las capas que integran el modelo, debe ofrecer servicios a las entidades que se encuentran en la capa superior inmediatamente adyacente. Estos servicios se identifican mediante puntos de acceso al servicio (***SAP***, Service Access Point), dentro del sistema. Este enfoque permite que existan varias entidades en la capa superior que utilicen los servicios de la capa inferior vía multiplexación del servicio. De la misma forma, en cada capa también se pueden definir puntos de acceso al servicio de red (***NSAP***, Network ***SAP***), lo que identifica a una entidad de transporte que utiliza dicho servicio vía la red. Esta multiplexación de servicios no se debe llevar a cabo obligatoriamente en todos los niveles del modelo, pero está permitida.

La ilustración que se presenta a continuación, muestra, de manera más clara, la forma en que interactúan los elementos antes señalados.

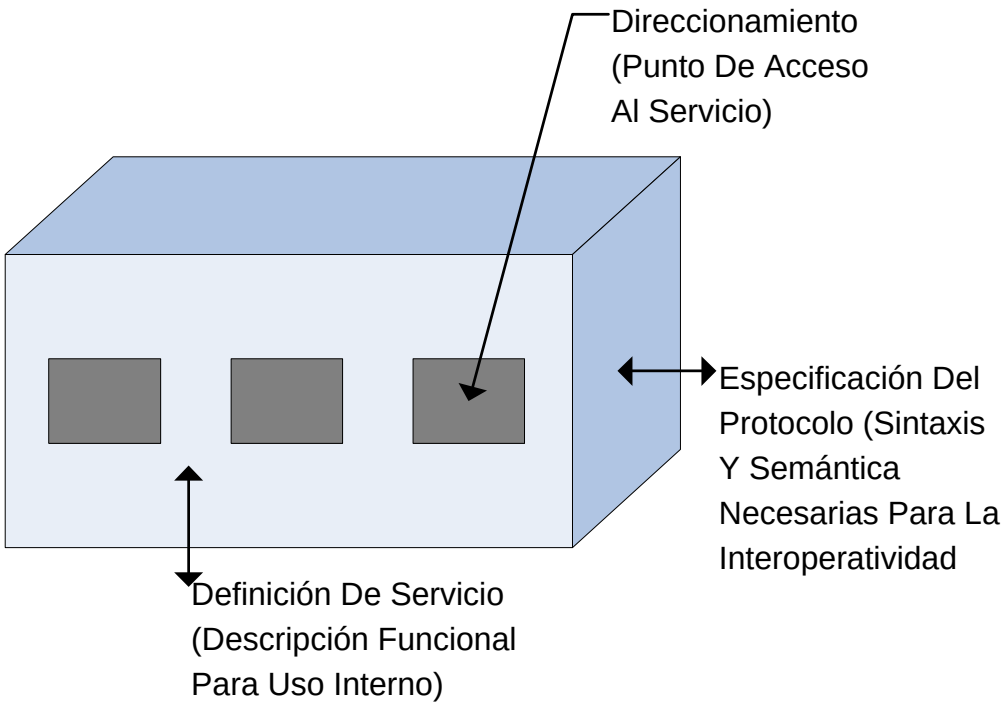

Ilustración 2.15: Elementos Clave En La Normalización De Capas Del Modelo OSI
Fuente: [STAL04]

2.7.3 Primitivas De Servicio Dentro Del Modelo OSI.

Bajo el modelo ***OSI***, los servicios que ofrecen sus capas adyacentes entre sí, se denominan como ***Primitivas de Servicio*** **[UITT93]**.

Una ***Primitiva de Servicio*** no es más que una operación o función -desde el punto de vista de la programación modular-, donde se especifica una acción explícita que se debe concretar; involucra parámetros y valores retornados, que se utilizan para intercambiar datos e información de control, entre las capas de la arquitectura.

La ***Primitiva*** es invocada por la capa N del modelo -definida como ***Usuario del Servicio***-, cuando trata de comunicarse con la capa $N-1$ -denotada como el ***Proveedor del Servicio***-.

La forma explícita que adoptará una ***Primitiva*** dada, dependerá de los detalles de implementación que se involucran en el protocolo particular donde se define **[KOZI05]**.

Dentro del modelo ***OSI***, se establecen cuatro primitivas fundamentales, que se enumeran en la tabla que se presenta a continuación:

Nombre	Significado / Interpretación
Solicitud	Esta primitiva la emite el usuario del servicio para invocarlo o enviar los parámetros requeridos para definir explícitamente el servicio solicitado.
Indicación	El proveedor del servicio emite esta primitiva bajo dos situaciones: 1. Para señalar que se ha invocado un procedimiento por parte del usuario del servicio par; así como para ofrecer los parámetros asociados a dicho procedimiento. 2. Cuando se requiere informar al usuario del servicio de que el proveedor ha iniciado una acción.
Respuesta	El usuario del servicio emite esta primitiva cuando se requiere confirmar o completar algún procedimiento previamente invocado por una **Indicación**, para ese usuario.
Confirmación	El proveedor del servicio publica esta primitiva cuando requiere confirmar o completar algún procedimiento previamente invocado mediante una **Solicitud** del Usuario del Servicio.

Tabla 2.3: Primitivas De Servicio Del Modelo OSI
Fuente: [STAL04]

Por otro lado, las interacciones entre las entidades usuario y proveedor del servicio se pueden sintetizar esencialmente en la secuencia de pasos que se describe a continuación, cuando, una entidad N intercambia datos con su entidad par en otro sistema:

1. La entidad origen (N) invoca a su entidad $(N-1)$ utilizando una primitiva de ***Solicitud***. Implícitamente, asociados a esta primitiva van los parámetros tales como los datos que serán transmitidos y la dirección del destinatario.

2. La entidad origen $(N-1)$ prepara una ***PDU*** $(N-1)$ para enviárselos a su entidad par $(N-1)$.

3. La entidad destino $(N-1)$ entrega los datos al destinatario adecuado (N) a través de una primitiva de ***Indicación***, que incluye los datos y la dirección de origen como parámetros.

4. En caso que se requiera una confirmación, la entidad destinataria (N) emite una primitiva de ***Respuesta*** a su entidad $(N-1)$.

5. La entidad $(N-1)$ convierte la confirmación en una ***PDU*** $(N-1)$.

6. La confirmación se entrega a la entidad (N) como una primitiva de ***Confirmación***.

Cuando esta secuencia de eventos se completa, se dice que se realizó un ***Servicio Confirmado***, ya que la entidad que inicia la transferencia, al final recibe una confirmación que le informa del éxito de la transacción en el otro extremo.

Por otro lado, si únicamente se invocan las primitivas de ***Solicitud*** *e* ***Indicación*** -que corresponden a los pasos 1, 2, y 3-, entonces dice que se realizó un ***Servicio No Confirmado***, ya que la entidad que inicia la transferencia no recibe confirmación de que la acción que solicitó se concretó.

La ilustración que se muestra a continuación, permite observar gráficamente ambos tipos de servicio.

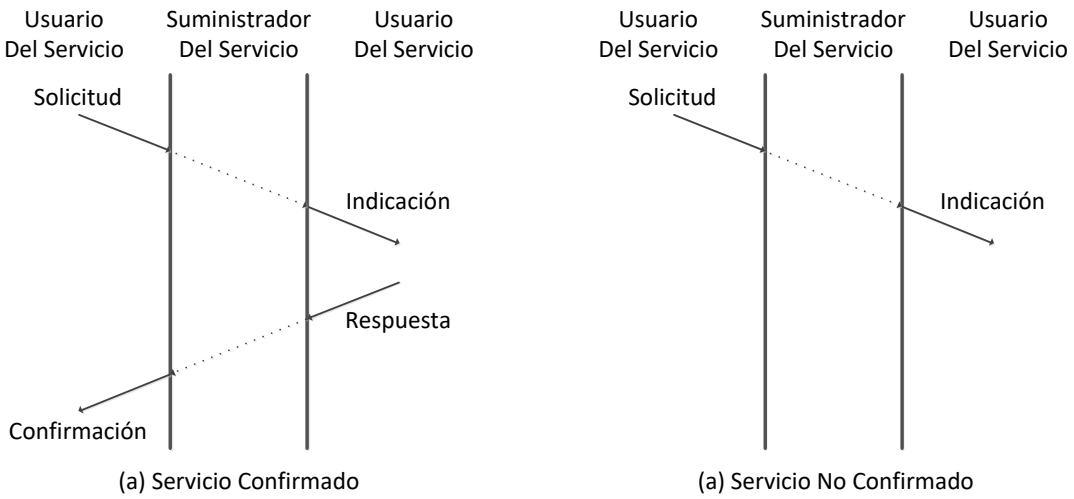

Ilustración 2.16: Secuencia Temporal De Las Primitivas De Servicio
Fuente: [STAL04]

2.7.4 Relación Entre Servicios Y Protocolos.

Para **[TANE03]**, los conceptos de servicio y protocolo son diferentes.

Un servicio es un conjunto de operaciones o primitivas que le ofrece una capa cualquiera de la pila de protocolos, a las que están por encima de ella. Se refiere a una interface entre capas, siendo la capa inferior la que ofrece el servicio, en tanto que la capa superior hace uso de él.

El servicio establece o define las operaciones que la capa está preparada para que sus usuarios utilicen, sin embargo, no establece nada acerca de la forma en que se instrumentarán estas facilidades.

Por otro lado, un protocolo de comunicaciones es un conjunto de reglas que gobiernan el formato y el significado de los paquetes o mensajes que se intercambian entre entidades pares dentro de una misma capa. Los protocolos

son utilizados por las entidades con el fin de implementar sus definiciones de servicios, además de tener la libertad de cambiar los protocolos a voluntad, en tanto y cuanto no se cambien los servicios visibles a sus usuarios. En consecuencia, tanto los servicios como los protocolos están totalmente desacoplados.

Comparándolos con los lenguajes de programación, un servicio equivale a un tipo de dato abstracto o a un objeto en un lenguaje orientado a objetos, ya que define las operaciones que se pueden realizar con él, sin especificar la forma en que está implementado internamente. En tanto que un protocolo de comunicaciones se refiere a la implementación del servicio, por lo que como tal, no es visible a los usuarios del mismo.

2.8 GLOSARIO DE TÉRMINOS TÉCNICOS.

A continuación, se expondrá un glosario con los términos técnicos más empleados en este informe de investigación:

Arquitectura de protocolos conjunto de protocolos y capas agrupados bajo un sistema de estándares intercompatibles.

ATM Abreviatura de Modo de Transferencia Asíncrona (Asynchronous Transfer Mode).

Capas También conocidas como niveles. Son entidades adyacentes entre sí, que intercambian ***PDU's*** y poseen interfaces de comunicación comunes.

Cola También conocida como Trailer. Son los campos que cierran las ***PDU's***.

CRC Abreviatura de Chequeo Redundante Cíclico.

DNS Acrónimo de Servicio de Nombres de Dominio (Dominion Name Service).

Encabezado También conocidos como Header. Son los campos iniciales en las PDU's que intercambian las capas.

Entidad cualquier ente que es capaz de enviar y recibir información.

IEEE Acrónimo de Instituto de Ingenieros Eléctricos y Electrónicos.

IETF Acrónimo de grupo especial de ingeniería de internet, mejor conocido en inglés como Internet Engineering Task Force.

Interface operaciones y servicios encapsulados y preestablecidos.

IP Abreviatura de Protocolo de Internet (Internet Protocol).

LAN Abreviatura de Red de Área Local Cableada (Local Area Network).

OSI Abreviatura de Interconexión de Sistemas Abiertos (Open Systems Interconnection).

PCI Acrónimo de protocol control information, o información de control del protocolo.

PDU Acrónimo de unidad de datos de protocolo, o protocol data unit, en inglés.

Protocolo Conjunto de reglas o convenciones que gobiernan todos los aspectos del intercambio de información entre sistemas.

TCP/IP Abreviatura de Protocolo de Control de Transmisión/Protocolo de Internet (Transmission Control Protocol/Internet Protocol).

CAPÍTULO 3

CAPÍTULO 3: MARCO METODOLÓGICO.

En esta etapa del proyecto, se utilizarán las herramientas que brinda la metodología de la investigación, para que los resultados obtenidos al culminarla, tengan validez científica. Para alcanzar este fin, se procede de acuerdo a los siguientes pasos:

1. Se establece el tipo de investigación que se realizará.

2. Se estructura el diseño de investigación que se utilizará.

3. Se plantea el problema de investigación.

4. Se sistematiza el problema de estudio.

5. Se describe, de manera ordenada, los pasos metodológicos que se van a seguir durante la investigación.

3.1 TIPO DE INVESTIGACIÓN.

El proyecto que se está describiendo se ubica en el tipo de investigación descriptiva, ya que se desarrolla alrededor de la realidad de un hecho, recolectando información, así como localizando atributos y cualidades para analizarlas y concluir con los productos que brinde el proceso **[HEFE06].**

Esta categoría de investigación tiene como meta la descripción de situaciones, eventos, propiedades, características o hechos, tal como son y cómo se manifiestan. Es decir, se trabaja por establecer el perfil del objeto de estudio.

Por otro lado, esta investigación, además de ser considerada como de tipo descriptivo, también se le ubicar como de tipo bibliográfico exploratorio, porque trata de examinar, con cierto nivel de detalle, un tema que generalmente es poco estudiado en los cursos de la especialidad **[HEFE06]**, situación que se observa en este proyecto**.**

3.2 DISEÑO DE LA INVESTIGACIÓN.

Por otro lado, en cuanto al diseño de este proyecto de investigación, se puede establecer que involucra un estudio no experimental, ya que no se realiza ningún tipo de manipulación intencional de variables independientes **[HEFE06]**.

A su vez la investigación pertenece al diseño transeccional exploratoria **[HEFE06]**, pues se recolectan los datos en un solo momento, en un tiempo único, además de enfocarse en el estudio de un problema poco conocido, como lo es la arquitectura del protocolo de comunicaciones que utiliza el ***FDJ***, que no se encuentra documentada en ningún libro, dadas sus características de prototipo y de ser el producto de un trabajo de grado **[DUTA01]**.

3.3 PLANTEAMIENTO DEL PROBLEMA.

El ***FDJ*** es un prototipo de herramienta informática producto del trabajo de grado de **[DUTA01]**. En consecuencia, es una implementación incompleta y casi no documentada, lo que permite plantear como problema o sujeto de estudio al ***desconocimiento de las características estructurales y funcionales del protocolo de comunicación que se utiliza en dicho sistema.***

3.3.1 Formulación Del Problema.

El problema antes planteado se puede formular formalmente con la siguiente pregunta:

¿Cuáles son las características estructurales y funcionales que presenta el protocolo de comunicaciones que utiliza el ***FDJ***?

3.3.2 Sistematización Del Problema.

El problema antes planteado, puede sistematizarse a través de las preguntas que se plantean a continuación:

1. ¿Cómo está organizada la arquitectura del ***FDJ***, a nivel estructural y funcional?

2. ¿De qué manera se estructuran los mensajes que se intercambian entre los clientes y el servidor del ***FDJ***.

3. ¿Cómo se confronta la arquitectura de comunicaciones del ***FDJ***, frente a los elementos estructurales y funcionales que deben estar presentes en este tipo de software?

4. ¿Cuáles son las capas de software que constituyen al ***FDJ***?

5. ¿Cuáles son las interfaces funcionales del protocolo de comunicación del ***FDJ***, en términos de primitivas de servicio?

6. ¿Cómo se confrontan las interfaces funcionales del protocolo de comunicación del ***FDJ***, en términos de primitivas de servicio, ante los elementos estructurales y funcionales que se espera que evidencie.

3.4 SUJETO DE INVESTIGACIÓN.

En esta sección se describe el perfil del sujeto de investigación, que será tomado en cuenta para el estudio.

En tal sentido, el sujeto de investigación está constituido por la documentación y el código fuente del ***FDJ***.

En cuanto a la documentación escrita que aporta **[DUTA01]**, ella es bastante clara y consistente. Sin embargo, hay momentos en que resulta insuficiente su profundidad, por el nivel de detalle que se requiere manejar para comprender el funcionamiento interno del protocolo. En tal caso, se debe recurrir al análisis directo del código fuente de la aplicación. Dicho código fuente tiene niveles de detalle variables.

Por un lado, algunas de las clases fueron programadas de acuerdo a lo que se conoce como ***"código autodocumentado"***, que se caracteriza por ofrecer un código fuente bien estructurado, con abundantes comentarios que explican y aclaran el funcionamiento de cada una de los identificadores involucrados en el código, así como la asignación de nombres a los identificadores siguiendo un principio de claridad semántica –es decir, el nombre de la variable se identifica directamente con su contenido-. En tanto que otras clases sencillamente carecen de todas las ayudas previamente descritas.

3.5 PROCEDIMIENTO METODOLÓGICO QUE SE RESPETARÁ DURANTE LA INVESTIGACIÓN.

La ejecución de la investigación se realizará de acuerdo a los pasos que se describen a continuación, que tienen como eje central, al sujeto de estudio de este proyecto:

1. Se inicia con la caracterización de la arquitectura del ***FDJ***, desde el punto de vista de la ingeniería de software inicialmente, y posteriormente, desde el enfoque de la arquitectura de protocolos. Esta caracterización será enfocada desde dos enfoques: el enfoque estructural –que procura establecer ***cuáles son los componentes*** que integran al sistema-, así como desde el enfoque funcional –que trata de establecer la ***forma en que se relacionan los componentes*** estructurales de la aplicación-.

2. Posteriormente, se clasifican y estudian los mensajes que se intercambian en el ***FDJ***. Este análisis tiene como meta, establecer relaciones de paralelismo de modo que se puedan identificar dentro de los mensajes, a los elementos estructurales y funcionales que los deben integrar, de acuerdo a los principios que se establecen en el marco teórico.

3. Luego se organizan las clases que integran al ***FDJ***, a manera de pila, con el objetivo de establecer la secuencia de intercambio de mensajes que se sigue entre ellas, así como equiparar sus interfaces.

4. Finalmente, se establecen cuáles son las funciones –en términos de primitivas de servicio- que se utilizan dentro de las clases del ***FDJ***, para intercambio de mensajes entre los clientes y el servidor de esta

aplicación. Esta identificación, además, va acompañada de otro proceso de comparación y contraste contra el marco teórico de este proyecto de investigación, para establecer las fortalezas y debilidad que caracterizan a estas interfaces.

CAPÍTULO 4

CAPÍTULO 4: ANÁLISIS DE RESULTADOS.

En esta sección se analiza la estructura interna de los mensajes que intercambian los clientes y el servidor del ***FDJ***. Para lograr tal meta, se procede de acuerdo a los siguientes pasos:

1. Se analiza la arquitectura del ***FDJ***, enfocándose en los componentes que intervienen en el proceso de comunicación.

2. Se estudia la arquitectura de comunicación que emplea el ***FDJ***.

3. Se caracterizan los tipos de tráfico de mensajes que intercambian el servidor y los clientes del ***FDJ***.

4. Se investiga la estructura de los mensajes que utiliza el ***FDJ***, en términos de los elementos de información que se intercambian.

5. Se contrasta la arquitectura de comunicaciones del ***FDJ***, frente a los elementos estructurales y funcionales de los protocolos de comunicación, contemplados en el marco teórico de esta investigación, para resaltar sus fortalezas y debilidades.

6. Se establecen las funciones que utiliza el ***FDJ*** para intercambiar información, en términos de primitivas de servicio, resaltando sus fortalezas y debilidades.

4.1 ANÁLISIS DE LA ARQUITECTURA DEL *FDJ*: CLASES RELEVANTES PARA LA INVESTIGACIÓN.

De acuerdo a **[PRES05]**, la **Arquitectura de Software** consiste en la organización jerárquica de los componentes estructurales de un programa, en términos de módulos así como la forma en que ellos interactúan y las estructuras de datos que utilizan.

En consecuencia, según **[DUTA01]**, la arquitectura del ***FDJ*** se modela de acuerdo al diagrama de emplazamiento ***UML***, que se presenta a continuación:

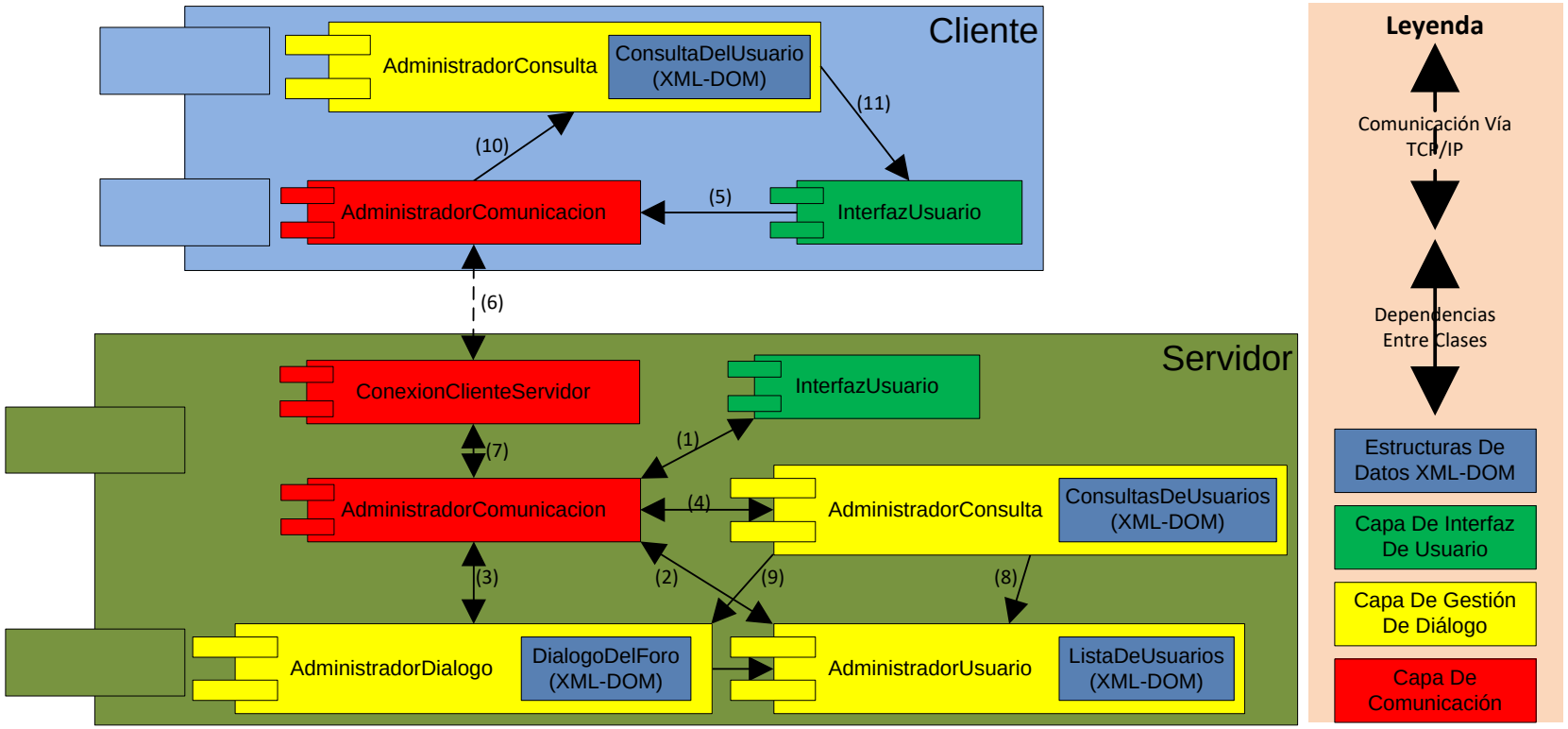

Ilustración 4.1: Arquitectura Del *FDJ*
Fuente: [DUTA01] Modificado Por El Autor

Dentro de este diagrama, aparecen representadas todas las clases que integran al ***FDJ***, así como las estructuras de datos que administran –representadas con los recuadros de color azul-. Los números que se colocan entre paréntesis referencian los procesos de intercambio de mensajes y datos entre dichas clases.

Las cabezas de flecha en las líneas de dependencia / comunicación señalan el sentido de intercambio de los mensajes entre las clases, en el caso de cabezas de fecha dobles se da a entender que los mensajes se pueden intercambiar de manera bidireccional, en tanto que las cabezas de flecha sencillas indican un único sentido de intercambio de mensajes. La cabeza de la fecha indica la clase que se instancia, en tanto que la cola de la flecha, señala la clase donde se declara la instanciación **[STPO02]**.

Adicionalmente, los colores que se han asignado a cada una de las clases, representan a las capas dentro de una arquitectura cliente/servidor de tres niveles bien marcada **[PRES05]**. Estos niveles se identifican como:

- **Interfaz de Usuario:** contiene las clases que administran las caras visibles de la aplicación para sus usuarios finales, tanto a nivel de los clientes como del servidor del ***FDJ***. Está identificada en la arquitectura de la aplicación con las clases de color verde. Para los efectos de esta investigación, no tiene mayor relevancia y no será analizada, ya que este proyecto se concentra en el protocolo de comunicaciones entre el cliente y el servidor de la aplicación, más que en la comunicación interna entre sub-clases locales, de conformidad con lo planteado en la sección Delimitadores.1.6.

- **Capa de Gestión De Diálogo:** en este nivel, se encuentran todas las clases que se encargan de administrar las transacciones de dialogo, consultas y usuarios, tanto a nivel de los clientes como del servidor. Se

identifica en la arquitectura del ***FDJ*** con las clases de color amarillo. Esta capa tiene relevancia en este proyecto, en términos de su interacción con la capa de comunicaciones.

- **Capa de Comunicaciones:** en ella se encuentran los administradores de comunicación de los clientes y el servidor, así como la clase de administración de comunicaciones cliente – servidor. Se encarga de concretizar los procesos de comunicación entre los sistemas clientes y el servidor, vía los enlaces de datos disponibles en el medio, establecidos a nivel de sockets ***TCP/IP***. Se identifica en la arquitectura del ***FDJ*** con las clases de color rojo. Para los fines de esta investigación, esta capa es el eje central de la investigación.

En consecuencia, se encuentran siete clases dentro del diagrama de emplazamiento antes presentado, relevantes para el proyecto, a saber:

- AdministradorComunicacion (Servidor).
- ConexiónClienteServidor.
- AdministradorComunicación (Cliente).
- AdministradorConsultaServidor (Servidor).
- AdministradorConsultaCliente (Cliente).
- AdministradorUsuario.
- AdministradorDialogo.

A continuación, se detalla la funcionalidad de estas clases, en términos del problema en estudio -procesos de comunicación-:

4.1.1 AdministradorComunicacion (Servidor).

Esta clase se encuentra localizada en el servidor del ***FDJ***. Tiene una clase homónima ubicada en el cliente de la aplicación, que será analizada posteriormente, en la sección 4.1.3. Las funciones que desempeña, a lo interno del sistema son:

- Controla la transmisión y recepción de los paquetes de información que se intercambian entre el servidor y cada uno de los clientes, de manera directa con la clase ***ConexiónClienteServidor*** (7).

- Recibe y desensambla los paquetes de los clientes, y envía -tanto la instrucción como el usuario-, al administrador correspondiente -Diálogos, Consultas o Usuarios-, para que se realicen las transacciones que ameriten (4), (2), (3), así como al interfaz del usuario del servidor (1).

- Ensambla y envía los paquetes a los clientes del foro, vía la clase ***ConexiónClienteServidor*** (7), en función a las actualizaciones de datos y estado que recibe de los administradores de Diálogos, Consultas o Usuarios (4), (2), (3).

- Controla la cantidad máxima de usuarios que pueden ser admitidos en el foro.

- Controla cuando se debe actualizar la información que se exhibe en la interfaz de Usuario del Servidor (1), (2), así como en las interfaces de los clientes (2), (3), (4), (7).

- La estructura interna de los paquetes que son intercambiados vía esta clase, será detallada posteriormente, en la sección 4.2.2.

4.1.2 ConexiónClienteServidor.

Esta clase también se encuentra ubicada en el servidor del ***FDJ***. En ella, es donde se maneja el componente concurrente del servidor, ya que su única función consiste en la administración de la conexión ***TCP/IP*** entre las clases ***AdministradorComunicacion*** del servidor y de cada uno de los clientes de la aplicación en general –vía la clase ***AdministradorComunicación*** que será analizada a continuación-.

4.1.3 AdministradorComunicación (Cliente).

Esta clase se encuentra localizada en el cliente del ***FDJ***. Como se ha mencionado previamente, posee una clase homónima en el servidor de la aplicación. En consecuencia, tiene algunas funciones en común con el ***AdministradorComunicación*** del servidor analizado en la sección 4.1.1. Sin embargo, posee a su vez, algunas particularidades propias. En tal sentido, las funciones que desempeña esta clase, a lo interno del sistema son:

- Controla la transmisión y recepción de los paquetes de información que se intercambian entre el cliente y el servidor (6).
- Ensambla y envía los paquetes que llegarán al servidor del foro (6), con base en la información recibida desde la clase ***InterfazUsuario*** del cliente (5).
- Recibe y desensambla los paquetes que llegan del servidor, y envía las instrucciones al ***AdministradorConsulta***, para que actualice la consulta del cliente (10).

- Cuando el cliente se conecta por primera vez, recibe el identificador interno de usuario que utilizará, o un mensaje de error (6), (5).

- La estructura interna de los paquetes intercambiados con el servidor del ***FDJ***, será detallada posteriormente, en la sección 4.2.2.

4.1.4 AdministradorConsultaServidor (Servidor).

Un análisis más detallado de la estructura y el funcionamiento interno del código que constituye al ***FDJ,*** establece que la clase ***AdministradorConsultaServidor*** desempeña un papel relevante en el proceso de comunicación del sistema.

Ella se encuentra localizada en el servidor de la aplicación y contiene varios métodos que se encargan de: Ensamblar los grupos de enunciados que enviará la clase ***AdministradorComunicacion*** del servidor, a cada uno de los clientes conectados -de manera individual-, que corresponden a mensajes agregados o modificados por alguno de los clientes conectados.

Es importante recalcar que la clase antes mencionada desempeña otras tareas significativas dentro del ***FDJ***. Sin embargo, para los efectos de la investigación en curso, únicamente resultan relevantes las funciones antes señaladas.

4.1.5 AdministradorConsultaCliente (Cliente).

Al igual que su clase simétrica en el servidor del foro, la clase ***AdministradorConsultaCliente*** juega un papel relevante en los procesos de comunicación, aunque a primera vista, no lo tiene.

Esta clase se encuentra localizada en el cliente de la aplicación y contiene varios métodos que realizan, entre otras, una tarea relevante en el proceso de comunicación de datos entre el cliente y el servidor del ***FDJ***. Específicamente, se encargan de: Desensamblar los mensajes recibidos del servidor, que corresponden a grupos de mensajes agregados o modificados por alguno de los clientes conectados -incluyéndolo a él mismo-.

Nuevamente es relevante destacar que esta clase también desempeña otras tareas significativas dentro del ***FDJ***, pero que no son significativas para los efectos de la investigación en curso.

4.1.6 AdministradorDialogo.

Esta clase también, a primera vista, parece que no tiene relación directa con los procesos de comunicación del ***FDJ.*** Sin embargo, también desempeña un papel relevante en el proceso de comunicación del sistema. Explícitamente, se encarga de: desensamblar los mensajes que se reciben en el servidor, desde un cliente específico, para integrarlas al diálogo que se administra en la sesión. Dichos mensajes, según el caso, serán agregados, o modificando al diálogo – (3), (7), (6), (10), (11)-, o serán rechazados –(3), (7), (6), (10)-.

Reiteradamente, es relevante destacar que esta clase también desempeña otras tareas significativas dentro del ***FDJ***, pero que no son significativas para los efectos de la investigación en curso.

4.1.7 AdministradorUsuario.

Esta clase también se encuentra ubicada en el servidor del ***FDJ***. En ella, es donde se administra el ingreso y la salida de los usuarios en el foro –(2), (7),

(6)-. Además, informa a los usuarios acerca del éxito o fracaso de sus intentos de ingreso a la sesión –(7), (6), (5)-.

4.2 ANÁLISIS DE LA ARQUITECTURA DE COMUNICACIÓN DEL *FDJ*.

De acuerdo lo planteado en la sección 2.1, las **arquitecturas de protocolos** estan constituidas por conjunto de normas agrupadas bajo un sistema de estándares intercompatibles, que lógicamente, permiten completar algún tipo de proceso de comunicación.

En **[DUTA01]** no se plantea explícitamente una construcción como tal para el ***FDJ****;* únicamente se modela una aplicación que realiza estas tareas, y se plantea explícitamente, la estructura de tramas que emplea dicho sistema. Es decir, no se dispone de un protocolo de comunicaciones formalmente definido.

Sin embargo, el ***FDJ*** es un prototipo funcional lo que implica que su sistema de comunicaciones tiene cierto nivel de eficiencia, ya que puede administrar el diálogo para el que fue creado **[DUTA01]**.

El eje principal de esta investigación consiste en analizar detalladamente las descripciones de las clases de la sección anterior (4.1), en conjunto con el estudio de los diagramas de clases estáticas del cliente y del servidor del ***FDJ***, que se encuentran en **[DUTA01]**, a nivel de las descripciones de sus atributos y métodos públicos, así como de sus trazas de ejecución -ver secciones desde la 0 hasta la 0 de esta investigación-, con la finalidad de examinarlos con cierto nivel de profundidad.

Este análisis tiene como meta la caracterización de los procesos de comunicación que están involucrados en el funcionamiento de la aplicación,

conjuntamente con las tramas que están involucradas en dichos procedimientos.

Posteriormente, este estudio se podrá utilizar como punto de partida en la propuesta de diseño de un protocolo de comunicación formal para dicha aplicación.

4.2.1 Caracterización De Los Procesos De Comunicación Del *FDJ*.

En primer plano, cuando se inicia la aplicación servidor del ***FDJ***, se establecen los parámetros del diálogo -en particular de interés, el puerto de comunicaciones que se utilizará-. Posteriormente, al levantar el servidor, se ***inicializa una conexión TCP/IP de servicio***, denominada ***ServerSocket*** que es, esencialmente, el mecanismo que utilizará cada cliente de la aplicación para conectarse al servidor. Esta conexión queda a la espera de solicitudes de los clientes.

Posteriormente, cada vez que la aplicación cliente se inicia, se le debe proporcionar la dirección ***IP*** o el nombre de ***equipo*** del servidor, así como el **puerto** de conexión que se utiliza, para levantar su sesión de trabajo en el servidor, esto asumiendo que el servidor del foro se levantó en un equipo cuya dirección ***IP*** es visible desde el sistema cliente.

En el momento que el cliente se intenta conectar con el servidor, inicialmente, se produce un intercambio de información, orientado a la ***administración de los usuarios***, ya que cada usuario dentro de la sesión del servidor, debe poseer un nombre único, para que se acepte al usuario. En caso de que se proponga un nombre duplicado, el servidor rechaza la conexión del cliente, pero él puede insistir nuevamente, utilizando un nombre no registrado.

Una vez se establece la conexión entre un cliente cualquiera y el servidor, los intercambios de información entre estos sistemas se orientan a la ***administración de enunciados***.

En este momento, el sistema cliente recibe una ***vista inicial del diálogo*** que se desarrolla hasta ese momento e inmediatamente, queda en posibilidad de utilizar toda la funcionalidad del foro, que en sí, no es de interés dentro de esta investigación.

En general, cada vez que un cliente incluye enunciados dentro del diálogo, ellos se reflejan en los otros clientes -así como en su propia sesión-, a través del siguiente procedimiento:

- Cuando el cliente incluye un enunciado, éste se envía al servidor, únicamente.
- Posteriormente, el servidor lo incluye en el diálogo general que mantiene, y actualiza todas las vistas de los clientes, que lo requieran.
- Luego, para cada uno de los clientes -uno por uno, inclusive el que incluyó el enunciado-, se procede de la siguiente manera:
  - Se envía la vista completa que debe observarse en su interface de trabajo, en el caso que haya solicitado un cambio de vista.
  - Sólo recibe los enunciados necesarios para sincronizar su vista con la existente en el servidor, en el caso que requiera la actualización de la que tiene.

Cuando un usuario se retira del diálogo, es desactivado dentro del servidor, y todos los enunciados que él ha incluido quedan bloqueados, ya que ningún

otro cliente está facultado para alterarlos. En esta etapa, nuevamente el intercambio de información, se orienta a la ***administración de los usuarios.***

Del análisis anterior, se desprende que, en alto nivel, hay dos categorías de procesos bien marcados de intercambio de información dentro del ***FDJ***: por un lado hay procesos de administración de usuarios, en tanto que por otro se presentan procesos de administración de enunciados.

4.2.2 Estructura De Los Mensajes Del *FDJ*.

En esta sección se analizará la estructura de mensajes que utiliza actualmente el ***FDJ***, de acuerdo a **[DUTA01]**.

Es relevante resaltar que se caracterizan claramente las ***PDU's*** que emplea la aplicación, al momento de intercambiar información, de conformidad con lo que se establece en la sección 2.2 de esta investigación, con la singularidad de que se establecen con longitud variable, en contraposición de lo que establece el concepto de segmentación.

Adicionalmente, se identifican dos tipos de mensajes, bien definidos que se procederá a analizar: Mensajes simples y mensajes complejos.

4.2.2.1 Mensajes Simples.

En **[DUTA01]**, se denominan **mensajes simples** a los elementos básicos de intercambio de información entre clientes y servidores. A nivel interno, son simplemente cadenas de texto plano delimitado. Por otro lado, semánticamente, son equivalentes a las ***PDU's*** de cualquier protocolo estándar, ya que poseen la estructura, que se observa en la siguiente ilustración:

$\{tipoMensaje\}\{cabecera\}\{datos\}$

Ilustración 4.2: Estructura De Los Mensajes Simples Del *FDJ*
Fuente: [DUTA01]

En dicha estructura, se pueden identificar tres campos claramente definidos, -con base en el análisis del proyecto de investigación, así como del código fuente que lo acompaña- que son:

4.2.2.1.1 Tipo De Mensaje.

Definido en la trama como $\{tipoMensaje\}$. Consiste en un único carácter alfabético escrito en mayúscula que identificará a la ***PDU*** como un mensaje de una categoría pre-establecida -este diseño limita la cantidad de mensajes que se pueden intercambiar a únicamente 26, los caracteres del alfabeto inglés, aunque hasta el momento, parece que han sido suficientes-.

4.2.2.1.2 Cabecera.

Definido en la trama como $\{cabecera\}$ está integrado por una secuencia de caracteres numéricos, delimitados por "/", que definen la longitud de cada uno de los sub campos que componen al mensaje. La presencia o ausencia de un campo en particular estará predispuesta por el valor particular del componente $\{tipoMensaje\}$.

La afirmación antes planteada implica que se debe entender que a cada tipo de mensaje se le asocia una cabecera particular, lo que representa un nivel de complicación adicional para el proceso de comunicación entre clientes y servidor, ya que no se sigue un patrón uniforme dentro de todas las tramas. Sin embargo, esto le ofrece una enorme flexibilidad al momento de organizar los datos dentro del paquete.

Adicionalmente, al analizar la documentación existente se puede observar que no todos los tipos de mensaje tienen involucrada una cabecera. Siendo reemplazada por el campo $\{tipoMensaje\}$.

4.2.2.1.3 Datos.

Definido en la trama como $\{datos\}$ está constituido por la cadena de caracteres delimitados por "/", que conforma al mensaje en sí mismo. Su estructura está determinada directamente por el tipo de mensaje y la cabecera particular que la precede.

4.2.2.2 Análisis De La Estructura De Los Mensajes Simples.

Examinando con más detalle la información disponible acerca de los mensajes simples, se pueden destacar los aspectos que se detallan a continuación:

La siguiente tabla, detalla los tipos de mensaje que gestiona el ***FDJ***, desde el servidor, hacia los clientes y se organiza con la información contenida en **[DUTA01]**.

Tipo De Mensaje	Carácter
Tema de Discusión	T
Inicio / Fin de Mensaje Complejo	I
Reemplazar Enunciados	R
Mensajes de la Línea de Estado	M
Anexar Enunciados	A
Sobrescribir Enunciados	S
Enunciado	E

Tabla 4.1: Tipos De Mensaje Del *FDJ Desde El Servidor Hacia Los Clientes*
Fuente: El Autor

Por otro lado, en la siguiente tabla continuación, detalla los tipos de mensaje que gestiona el ***FDJ***, desde los clientes, hacia el servidor, nuevamente se organiza con la información contenida en **[DUTA01]**.

Tipo De Mensaje	Carácter
Evidencia Asociada de un Enunciado	E
Cambiar de Consulta	V
Salir de la sesión	S
Agregar Enunciado	A
Modificar Enunciado	M

Tabla 4.2: Tipos De Mensaje Del *FDJ Desde Los Clientes Hacia El Servidor*
Fuente: El Autor

Por otro lado, en los anexos de este documento, se expone el detalle de los campos que integran a cada uno de los tipos de mensaje simple que gestiona el ***FDJ***, analizado por **[DUTA01]**.

En la ilustración que se muestra a continuación, se pueden examinar los campos básicos de una ***PDU*** cualquiera, emparejados con las estructuras equivalentes en la ***PDU*** del ***FDJ***, a través de los colores que se les asignan.

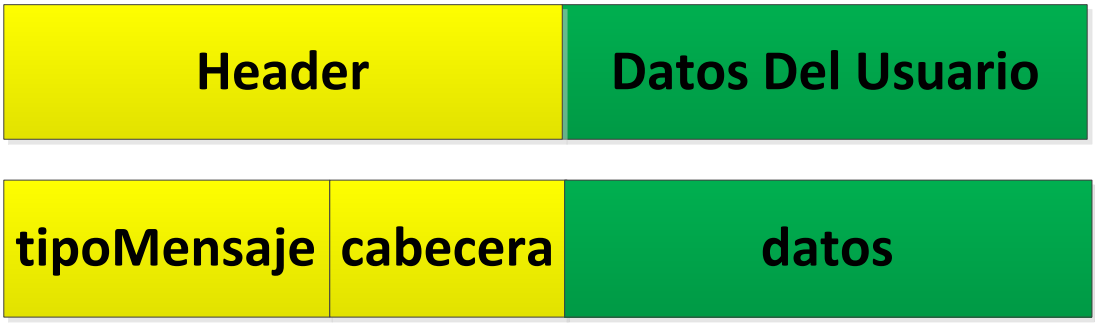

Ilustración 4.3: PDU's Estándar Y Del *FDJ*
Fuente: [DUTA01]

Es relevante observar que de acuerdo a la forma en que está organizada esta ***PDU***, se ha concebido como una cadena de bytes que tiene una longitud variable, en contraposición de lo que se maneja usualmente en

telecomunicaciones, relativo a que los campos de una ***PDU*** generalmente tendrán longitud fija y pequeña **[DAYJ08]**.

Además, dentro de sus campos, no se ha contemplado información para la detección de errores, lo que involucra problemas potenciales de consistencia e integridad de información.

Por otro lado, el modelo de ***PDU*** planteado resulta bastante flexible, ya que administra la información como vectores de longitud variable, por cuanto que:

- Como por definición $\{tipoMensaje\}$ es un campo que sólo posee un carácter, es fácilmente manipulable vía algún tipo de estructura de selección “case”.

- Establecido el campo $\{tipoMensaje\}$, se puede establecer cuales son sub-campos que integran al registro en mención. Además, la longitud de cada sub campo estará determinada en el campo $\{cabecera\}$.

- En el caso especial que el mensaje que se recibiera constara únicamente de una única expresión, se podrían eliminar los campos $\{cabecera\}$ o $\{datos\}$, sin pérdida de generalidad, de modo que el mensaje se reduciría a $\{tipoMensaje\}\{datos\}$ o $\{tipoMensaje\}$.

- Conociendo los sub campos recibidos, así como sus longitudes, se puede almacenar la cadena recibida en alguna estructura de datos temporal como un vector o un registro.

- Al final de todo este proceso, el cliente o el servidor del ***FDJ*** ha recibido la ***PDU*** y tiene almacenado en memoria el mensaje recibido, conjuntamente con todos sus parámetros, por lo que pueden realizar el procesamiento que corresponda al mensaje recibido.

Para concluir el análisis de los mensajes simples, se puede establecer que en sí, el modelo de ***PDU*** que utiliza el ***FDJ***, es funcional a nivel de la aplicación, ya que ofrece un nivel de flexibilidad significativo en cuanto a expresividad de intercambio de información.

Sin embargo, al momento de considerar el problema de comunicación vía un medio compartido, se presenta un serio problema potencial de acaparamiento de medios, establecido por la longitud de los mensajes intercambios -ya que tienen medidas variables-.

Este tipo de problema, generalmente se controla en las capas inferiores de las pilas de protocolos, con tecnologías tales como el modo de transferencia Asíncrona o ***ATM*** -Asynchronous Transfer Mode en inglés-, de modo que la red de comunicaciones como tal, no se verá afectada en su rendimiento **[FORO07]**.

No obstante, la situación cambia a nivel del ***FDJ***, ya que puede conducir a situaciones de bloqueo tanto en los clientes como los servidores. Esto, por cuanto que al momento en que la aplicación envía o recibe un paquete de longitud significativa, ella se bloquea durante la lectura o escritura al medio compartido, lo que puede conducir a problemas de retardo en el rendimiento global del sistema. El problema se agravará en la medida en que la longitud de los mensajes se incremente, por lo que se le debe buscar algún tipo de solución a nivel de la aplicación.

4.2.2.3 Mensajes Complejos.

Por otro lado, en **[DUTA01]**, se denominan **mensajes complejos** a los grupos de mensajes simples que se enmarcan entre dos mensajes especiales, denominados como ***InicioTrama/FinTrama***. Ellos, delimitan a un conjunto

arbitrario de mensajes simples -de longitud no establecida-, tal como se ha descrito en la sección 4.2.2.1.

Es decir, estos mensajes responden a la estructura que se presenta a continuación:

$$\{tramaInicial\}$$
$$\{trama1\}$$
$$\{trama2\}$$
$$...$$
$$\{trama(n)\}$$
$$\{tramaFinal\}$$

Ilustración 4.4: Formato De Mensaje Complejo Del *FDJ*
Fuente: [DUTA01]

En consecuencia, de acuerdo a esta conceptualización, cada una de las tramas indicadas -inclusive las tramas inicial y final-, están conformadas por mensajes simples de longitud arbitraria, por lo que ***FDJ*** puede intercambiar mensajes complejos de dimensiones arbitrarias.

Estos mensajes, se enmarcan en tres tipos especiales, que se enuncian a continuación:

Reemplazar Enunciados	Sobrescribir Enunciados	Anexar Enunciados
I/InicioTrama R/Reemplazar E/1/2/3/4/5/6/7/8/9/10/11/DATA E/1/2/3/4/5/6/7/8/9/10/11/DATA …. E/1/2/3/4/5/6/7/8/9/10/11/DATA I/FinTrama	I/InicioTrama S/Sobrescribir E/1/2/3/4/5/6/7/8/9/10/11/DATA E/1/2/3/4/5/6/7/8/9/10/11/DATA …. E/1/2/3/4/5/6/7/8/9/10/11/DATA I/FinTrama	I/InicioTrama A/Anexar E/1/2/3/4/5/6/7/8/9/10/11/DATA E/1/2/3/4/5/6/7/8/9/10/11/DATA …. E/1/2/3/4/5/6/7/8/9/10/11/DATA I/FinTrama

Tabla 4.3: Mensajes Complejos que Envía el Servidor a los Clientes del *FDJ*
Fuente: [DUTA01]

4.2.2.4 Análisis De La Estructura De Los Mensajes Complejos.

Examinando con más detalle la información disponible acerca de los mensajes complejos, se pueden destacar los aspectos que se detallan a continuación:

Un aspecto curioso resulta del hecho que el sentido común sugiere que tanto los clientes como el servidor del ***FDJ*** deben utilizar los mensajes complejos como forma regular de intercambio de información. Sin embargo, de acuerdo a **[DUTA01]**, se establece que los clientes de la aplicación sólo requieren enviar mensajes simples al servidor. En consecuencia, este tipo de mensaje es empleado exclusivamente por el servidor, para enviar mensajes a los clientes de del ***FDJ***.

Por otro lado, por analogía con la sección 4.2.2.2, se puede afirmar que los clientes del foro también son capaces de procesar los mensajes que reciben del servidor de este tipo.

Para concluir el análisis de los mensajes complejos, al igual que en el caso de los mensajes simples, se puede establecer que en sí, este modelo de ***PDU*** compleja -si se le puede llamar así-, que utiliza el ***FDJ***, sigue siendo funcional a nivel de la aplicación, ya que amplía significativamente el nivel de flexibilidad en cuanto a expresividad de intercambio de información, que ofrecen los mensajes simples.

Adicionalmente, incrementa la magnitud de problema de comunicación potencial de acaparamiento de medios, previamente planteado en la sección 4.2.2.2, a nivel de los mensajes simples.

4.3 LA ARQUITECTURA DE COMUNICACIONES DEL *FDJ*, ANTE LOS ELEMENTOS ESTRUCTURALES Y FUNCIONALES DE LOS PROTOCOLOS DE COMUNICACIÓN.

En esta sección se procederá a contrastar y comparar detalladamente a la arquitectura de comunicaciones del ***FDJ,*** planteada en términos del sistema de mensajes simples y complejos que utiliza la aplicación -planteados en las secciones 4.1 y 4.1.5-; frente a los elementos estructurales y funcionales previamente establecidos en el marco teórico de esta investigación, en las secciones 2.3 y 2.4.

Este análisis tiene como objetivo, el resaltar las debilidades y fortalezas que presenta el protocolo en mención, con el objetivo de plantear, en un proyecto posterior, el diseño de un protocolo de comunicaciones más robusto, que mejore el funcionamiento del ***FDJ***.

Queda entendido que en cuanto a la sección 2.2 -la presencia o ausencia de ***PDU's***- el ***FDJ*** cumple plenamente con este elemento estructural de los protocolos de comunicación, salvo en el aspecto de que utilizan tramas de longitud variable.

4.3.1 Características Estructurales Del Protocolo Del *FDJ*.

A nivel estructural, el protocolo de comunicaciones del ***FDJ*** presenta los siguientes atributos:

4.3.1.1 Sintaxis.

En este aspecto, con base en los análisis previamente planteados en la sección 4.2.2, se puede afirmar categóricamente, que el protocolo del ***FDJ*** cumple a cabalidad con este aspecto. Sigue una sintaxis bien definida, para cada uno de los campos que integran a sus ***PDU's***, tanto a nivel de los mensajes simples como de los complejos, por cuanto que cada tipo de mensaje tiene bien definida su estructura interna, en términos de las secuencias de campos y la forma en que se organizan e identifican.

4.3.1.2 Semántica.

Por otro lado, este atributo también se considera satisfecho por el protocolo del ***FDJ***, ya que, de acuerdo a la sección 4.2.2, cada uno de los campos definidos de sus ***PDU's*** -simples y complejas-, tiene una representación semántica bien definida y explícita, en términos del problema de comunicación que se modela.

4.3.1.3 Temporización.

En contraste con los dos elementos estructurales anteriores, el protocolo del ***FDJ*** adolece de serios problemas de temporización, atribuibles esencialmente, al empleo de tramas de longitud variable.

El modelo de comunicaciones del ***FDJ*** confía ciegamente en la propiedad del protocolo ***TCP/IP*** para administrar sockets.

No se toma en cuenta que: al momento de enviar una cadena de bytes extensa vía Internet; de al menos varios cientos de bytes, tal como puede ser un

enunciado que envía un cliente al servidor, conjuntamente con toda la meta data que involucra; se pueden presentar problemas de bloqueo prolongados. Esto para el caso de los mensajes simples.

El problema se mitiga parcialmente, ya que los sockets, por definición, con canales lógicos full-dúplex **[ECKE07]**, pero es sólo un paliativo.

Sin embargo, lo ideal es que la aplicación controle el flujo de datos, ya que en el momento en que el sistema envía o recibe un conjunto significativo de datos, éste entrará en bloqueo, por lo que el usuario del mismo no lo podrá utilizar hasta el momento en que se complete la transacción. No obstante, el sistema operativo no se bloqueará, así como las restantes comunicaciones vía la red, ya que el protocolo ***TCP/IP*** controlará el despacho y recepción de los paquetes de todas las aplicaciones que lo requieran, y él si implementa mecanismos de control de flujo y temporización, que controlarán el problema a nivel de las aplicaciones restantes, asumiendo que están correctamente programadas.

En el caso de los mensajes complejos, la situación simplemente es significativamente más grave. En este caso, el servidor debe enviar, a cada uno de los clientes, al menos un mensaje complejo -que consta de uno o más enunciados-, cada vez que un cliente realiza un aporte en el foro, por lo que el problema se magnifica proporcionalmente a la cantidad de clientes que tengan una sesión abierta en la aplicación.

Adicionalmente, como se analizará más tarde, también carece de mecanismos de control de flujo, que podrían ayudar a controlar el problema que representa el problema de la temporización.

En consecuencia, para el aspecto particular de temporización, se puede concluir que representa una debilidad grave que adolece el protocolo de comunicaciones del ***FDJ*** y que debe ser corregida.

4.3.2 Características Funcionales Del Protocolo Del *FDJ*.

A nivel funcional, el protocolo de comunicaciones del ***FDJ*** presenta los siguientes atributos:

4.3.2.1 Encapsulamiento.

El análisis correspondiente evidencia las siguientes situaciones, en términos de información de control –sección 2.4.1-:

- **Direccionamiento:** en las ***PDU's*** que intercambian, tanto el servidor como los clientes, no se observa información de direccionamiento propiamente dicha. Esta función es controlada vía gestión de procesos, a través de primitivas de servicio, donde:

  - En el servidor, se crea un socket ***TCP/IP*** de servicio -serversocket-, con su dirección ***IP*** y un puerto de conexión preestablecido.

  - Posteriormente, cuando cada cliente se autentica, se crean sockets en ellos que apuntan a dirección ***IP*** y al puerto preestablecido del servidor. Cada uno de estos sockets es administrado en el servidor vía la creación de un proceso que se encarga de gestionar la comunicación entre un cliente particular y el servidor -un cliente, un proceso-. Es a través de la interacción vía estos puertos, que se administra la comunicación entre el emisor y los destinatarios de los mensajes.

En conclusión, esta característica es administrada a través de la creación de parejas de procesos entre el cliente y el servidor implementadas a través de primitivas de servicio, utilizando la información de direccionamiento ***IP*** y puerto de conexión disponibles entre ambos sistemas.

- **Detección de errores:** no se ha detectado información de manejo de errores a nivel de las ***PDU's*** intercambiadas entre los clientes y el servidor, así como tampoco a nivel de gestión de procesos en las aplicaciones antes mencionadas.

- **Control del protocolo:** En contraste, las ***PDU's*** evidencian una amplia gama de información de control del protocolo en sí, dada a través de los tipos de mensaje y cabeceras que acompañan a los datos intercambiados –ver sección 4.2.2-.

Concluyendo, el análisis de la característica de encapsulamiento, se puede establecer que el protocolo de comunicación del ***FDJ*** la implementa en gran parte, vía ***PDU's*** y primitivas de servicio. Esta implementación parcial implica que este sistema de comunicación depende exclusivamente de la forma en que la aplicación se programó, al momento de establecer el origen y destino de los paquetes de datos, lo que debe ser interpretado como una debilidad del modelo de gravedad media, que se debe remediar, especialmente por la carencia del control de errores.

4.3.2.2 Segmentación Y Ensamblado.

En función al modelo de tramas de longitud variable, el protocolo del ***FDJ*** no incluye esta característica, ni a nivel de sus ***PDU's***, ni a nivel de los procesos que integran a la aplicación, ya que de acuerdo a la sección 2.4.2.1, generalmente las ***PDU's*** deben tener una longitud fija y pre-establecida. A la

fecha, la segmentación y ensamblado de los mensajes la está gestionando internamente el socket ***TCP/IP***.

En conclusión, la inexistencia de la característica de segmentación y ensamblado, se debe interpretar como otra debilidad grave de la norma, en el sentido de que no está implementada y se debe considerar su inclusión, por la importancia que tiene al momento de controlar el acaparamiento de enlaces de transmisión.

4.3.2.3 Control De La Conexión.

Dado que el ***FDJ*** está implementado utilizando sockets ***TCP/IP*** que se encargan de gestionar el proceso de comunicación, en principio, se puede afirmar que esta aplicación implementa el control de la conexión vía un enlace orientada a conexión, de acuerdo a lo planteado en la sección 2.4.3.2, ya que se pueden observar las fases de[13].

- **Establecimiento de conexión:** implementada a través de los constructores:
  - ***AdministradorComunicacionCliente (dirección_servidor, puerto, nombre_usuario)*** de la clase ***AdministradorComunicacionCliente*** en el cliente.

---

[13] Los nombres de los parámetros que reciben los constructores y métodos han sido modificados a fin de facilitar a los lectores, su comprensión semántica.

- ***ConexionClienteServidor (identificador_proceso, socket_servidor, direccion_usuario, activo)*** de la clase ***ConexionClienteServidor*** del servidor.

- **Transferencia de datos:** se gestiona a través de los métodos:
  - ***enviar(mensaje)*** y ***repintarArbol(dialogo)*** de la clase ***AdministradorComunicacionCliente*** del cliente.
  - ***enviar_cad(mensaje)*** *y* ***refresh(mensaje)*** de la clase ***ConexionClienteServidor*** del servidor.

- **Cierre de la conexión:** que se administra a través de los métodos:
  - ***desconectar()*** de la clase ***AdministradorComunicacionCliente*** del cliente.
  - ***cerrarSocket()*** y ***cerrarThread()*** de la clase ***ConexionClienteServidor*** del servidor.

Sin embargo, contrastando con mayor detalle la forma en que se implementa la aplicación -a nivel del análisis del código fuente más las trazas de depuración que incluye el ***FJD***, ver sección 0-, se puede observar que la puesta en práctica de la característica es incompleta, ya que no se están cumpliendo los diálogos de tres partes descritos posteriormente en la sección 2.4.3.2, por cuanto que los métodos y constructores descritos envían y reciben los mensajes respectivos, más no se confirman las recepciones -que es el paso que falta para completar el proceso cabalmente-.

Concluyendo el análisis de la característica de control de conexión, se puede establecer que el protocolo de comunicación del ***FDJ*** no la implementa de manera completa. Esta carencia implica que este sistema de comunicación no

ofrece un mecanismo fiable para corroborar la recepción de los paquetes enviados, lo que debe ser interpretado como otra debilidad – de nivel medio-, del modelo, que requiere que se mejore.

4.3.2.4 Entrega En Orden.

Nuevamente, analizando el funcionamiento interno de las clases del ***FDJ***, conjuntamente con las trazas descritas en la sección 0, se puede observar que el intercambio de enunciados entre el servidor y los clientes, vía el protocolo de comunicaciones respectivo, se da con base en una disciplina estricta de gestión de colas ***FIFO*** -First In, First Out, primero que entra, primero que sale-. Este esquema de gestión asegura que todas las tramas serán entregadas en el orden correcto.

Sin embargo, el contexto de la entrega en orden se debe ubicar en el caso de que se cumple la función de segmentación y ensamblado. Es decir, que para cada mensaje que se envía, se crean muchos paquetes de tamaño relativamente pequeño, que se envían entre el emisor y el receptor de mensajes, con el objeto de no saturar el canal de comunicación con un único mensaje. Este no es el caso de los paquetes del ***FDJ***, ya que como se ha mencionado antes, ellos no son segmentados.

En consecuencia, del análisis de la característica de entrega en orden, se puede establecer que el protocolo de comunicación del ***FDJ*** no la implementa de manera completa. Esta carencia implica nuevamente que este sistema de comunicación no ofrece un mecanismo fiable para corroborar la recepción de los paquetes enviados, lo que debe ser interpretado como otra debilidad –de nivel medio-, del modelo, que requiere corrección.

4.3.2.5 Control De Flujo.

Nuevamente, del análisis del funcionamiento interno del ***FDJ***, conjuntamente con las trazas descritas en la sección 0, se puede observar que el intercambio de enunciados entre el servidor y los clientes, vía el protocolo de comunicaciones respectivo, no posee ningún mecanismo de control de flujo, tal como los descritos en la sección 2.4.5 –parada y espera o ventanas deslizantes, entre otros métodos-, donde se involucre algún esquema que regule el tráfico de tramas entre las entidades involucradas en el diálogo.

Esta carencia implica reiteradamente que este sistema de comunicación no ofrece un mecanismo fiable que controle la sobrecarga en los sistemas clientes o servidor del ***FDJ***, lo que debe ser interpretado como otra debilidad del modelo –de nivel grave-, que necesita corrección.

4.3.2.6 Control De Errores.

Revisando cuidadosamente la información relacionada con las ***PDU's*** que maneja el ***FDJ***, conjuntamente con las trazas descritas en la sección 0, no se puede identificar ningún campo que contenga información relacionada con esta característica, en ninguna de las formas en que se establece en la sección 2.4.6, donde se involucre un modelo para la detección o corrección de errores, así como de retransmisión de paquetes de datos dañados.

La ausencia de esta característica, en el protocolo de comunicación del ***FDJ*** involucra una debilidad notable en el sistema, ya que debe confiar en que los protocolos de niveles inferiores -tales como ***TCP/IP***-, realizan esta tarea en forma exclusiva, lo que no es cónsono con lo que se plantea en la sección

2.4.6, del marco teórico de este proyecto. Esta falencia –de nivel grave-, debe ser corregida y mejorada sustancialmente.

4.3.2.7 Multiplexación.

En función a la documentación de la arquitectura del ***FDJ*** que se encuentra en **[DUTA01]**, se puede establecer que el esquema de multiplexación que implementa esta aplicación es el trivial, **uno a uno**, tal como se establece en la sección 2.4.7, de este documento.

Este esquema de comunicación se constituye a través de un único enlace de comunicación definido por un puerto de comunicación -el 1220 por defecto, pero que puede ser modificado a voluntad por los administradores del servidor-, fundamentado en la dirección IP del sistema donde se ejecuta el sistema.

Tomando en cuenta la disponibilidad actual de las conexiones de banda ancha que ofrecen los proveedores de acceso a Internet, tanto del medio local como internacional-, este esquema de comunicación resulta a todas luces ineficiente, ya que no explota plenamente el recurso ancho de banda disponible, así como tampoco considera la carga sobre el enlace que representa la cantidad de usuarios conectados en el sistema. Es decir, que se implementen las posibilidades para utilizar multiplexación ascendente o descendente.

En consecuencia, esta característica no es llevada a la práctica de la mejor manera, en el ***FDJ*** –tanto a nivel de aplicación o del protocolo-, por lo que se considera que es ineficientemente el uso que se da a los recursos de banda ancha actuales, por parte de este sistema, lo que debe ser interpretado como una debilidad del protocolo – de nivel bajo-, que necesita corrección.

4.3.2.8 Direccionamiento.

Luego de un análisis cuidadoso, realizado al protocolo del ***FDJ***, a nivel de sus ***PDU's***, así como en la arquitectura de la aplicación antes mencionada, y de sus trazas descritas en la sección 0, se puede establecer que:

- El direccionamiento que utiliza este sistema –en los aspectos de nivel, alcance e identificadores de conexión, de acuerdo a los conceptos establecidos en las secciones 2.4.8.1, 2.4.8.2 y 2.4.8.3-, se fundamenta en el protocolo ***IP***, que es norma para la comunicación entre sistemas en Internet –en particular es el que utilizan los equipos clientes y servidor del ***FDJ***-, conjuntamente a un puerto de conexión preestablecido –por defecto 1220, modificable-, aunado al nombre que utiliza el usuario cliente para registrarse en la sesión.

- Por otro lado, en cuanto al modo de direccionamiento, el sistema emplea el esquema de comunicación ***uno a uno, unidestino*** o ***unicast*** –desarrollado en la sección 2.4.8.4-. Este hecho implica que la comunicación entre el servidor y los clientes de la aplicación no aprovecha los modos de transmisión multidestino y de disfusión –multicast y broadcast, respectivamente- lo que conduce nuevamente a desaprovechar el ancho de banda disponible en la actualidad.

- Adicionalmente, a este modelo sería deseable que se le agregara un identificador único para cada usuario, de modo que la duplicidad de entradas en el campo nombre, se pudiera permitir.

En consecuencia, en principio se puede afirmar que la característica de direccionamiento se encuentra implementada en el sistema. Sin embargo, un estudio más detallado revela que en realidad, se encuentra implementada a

nivel de la aplicación –en términos de primitivas de servicio-, en lugar de encontrarse en el protocolo.

Luego, al analizar la característica de direccionamiento, se puede establecer que el protocolo de comunicación del ***FDJ*** no la implementa. Sin embargo, a nivel de la aplicación en sí, se ofrece dicha funcionalidad de manera casi completa. Esta carencia involucra que el protocolo posee una debilidad menor, que debe ser corregida.

4.3.2.9 Encaminamiento.

Dado que el ***FDJ*** implementa su sistema de comunicación con base en sockets ***TCP/IP*** –con base en lo establecido por **[DUTA01]**-, y dicho protocolo –***TCP/IP***-, se caracteriza por ser un protocolo enrutable **[FORO07]**, se podría deducir que la aplicación está fundamentada en un protocolo enrutable, ya que utiliza esta funcionalidad vía protocolos de nivel inferior.

Por otro lado, dado que el enrutamiento es una tarea que deben cumplir los protocolos de bajo nivel, en tanto que la finalidad del ***FDJ*** es de alto nivel –mensajería instantánea donde el diálogo se maneja de manera jerárquica-.

En consecuencia, se tiene que, la característica en discusión, no necesariamente debe estar presente en todos los protocolos de comunicaciones de manera directa **[KURO07]**, por lo que su inexistencia a nivel de implementación directa en el ***FDJ*** no debe ser interpretada como una debilidad del protocolo, que necesita corrección.

4.3.2.10 Servicios De Transmisión.

En cuanto a la característica de los servicios de transmisión, el análisis realizado al protocolo del ***FDJ***, a nivel de sus ***PDU's***, así como en la arquitectura de la aplicación antes mencionada, y de sus trazas descritas en la sección 0, permite establecer que: tanto a nivel del protocolo en sí, como a nivel de las primitivas de servicio que ofrece, la característica en mención no se encuentra presente.

No se dispone de facilidades para alterar la prioridad del envío o recepción de los paquetes. Tampoco se disponen de mecanismos para controlar la calidad del servicio empleado, en términos de velocidades de transmisión mínimas o tiempos máximos de retardo. Finalmente, no existe ningún tipo de funcionalidad que ofrezca algún nivel de seguridad o privacidad en el intercambio de información.

En consecuencia, luego, al analizar la característica de servicios de transmisión, se puede establecer que el protocolo de comunicación del ***FDJ*** no la implementa. Esta carencia involucra que el protocolo posee una debilidad grave –por el aspecto de la seguridad-, que debe ser corregida.

4.3.3 Síntesis De Las Características Estructurales y Funcionales Del Protocolo Del *FDJ*.

A continuación, se presenta una tabla donde se sintetizan las características estructurales y funcionales, así como las debilidades y fortalezas, que evidencia el protocolo del ***FDJ***, de acuerdo al análisis antes planteado.

Característica	Tipo	Implementación	Presente En		Problemas	
			PDU	Primitivas De Servicio	Descripción	Gravedad
PDU	Estructural	Gran Parte	Si	No	Poseen Longitud y Composición Variable	Media
Sintaxis	Estructural	Total	Si	Si	No	Nula
Semántica	Estructural	Total	Si	Si	No	Nula
Temporización	Estructural	Nula	No	No	Característica inexistente	Grave
Encapsulamiento	Funcional	Gran Parte	Si	Si	Carece de detección de errores	Media
Segmentación y ensamblado	Funcional	Nula	No	No	Característica inexistente	Grave
Control de la conexión	Funcional	Gran Parte	No	Si	Diálogos de tres partes sin confirmación	Media
Entrega en orden	Funcional	Gran Parte	No	Si	Opera sin segmentación	Media
Control de flujo	Funcional	Nula	No	No	Característica inexistente	Grave
Control de errores	Funcional	Nula	No	No	Característica inexistente	Grave
Multiplexación	Funcional	Mínima	Si	Si	Implementación básica	Bajo
Direccionamiento	Funcional	Mínima	No	Si	Controlada por primitivas de servicio	Bajo
Encaminamiento	Funcional	Nula	No	No	No relevante	Nula
Servicios De Transmisión	Funcional	Nula	No	No	Carece de detección de niveles de seguridad	Grave

Tabla 4.4: Síntesis De Las Características Estructurales y Funcionales Del Protocolo Del *FDJ*
Fuente: El Autor

Tal como se puede observar en la tabla previamente expuesta, el protocolo de comunicaciones del ***FDJ*** tiene variables debilidades significativas, que requieren de una mejora significativa del estándar.

Por otro lado, en cuanto a sus fortalezas, sólo se puede destacar su capacidad de gestión de tramas de longitud y composición variables, pero con los problemas colaterales que ya se han discutido, relacionados a la longitud variable de las tramas, que dificulta la intercomunicación entre clientes y servidores, ya que disminuye la eficiencia del canal de comunicación (multiplexación del medio de transmisión).

4.3.4 Estructuración De Las Clases Del *FDJ* En Capas.

De acuerdo a lo planteado en las secciones 2.7.3, 4.1, 4.1.5 y 4.3, así como en la Ilustración 4.1, es posible organizar las clases del ***FDJ***, a manera de pila, en capas. Esta distribución tiene la finalidad de identificar los métodos que gestionan las funciones de intercambio de información dentro la aplicación –y parearlas entre sí como primitivas de servicio, así como para caracterizar al protocolo de comunicación en términos funcionales-. Esta estructuración se puede observar en la siguiente ilustración:

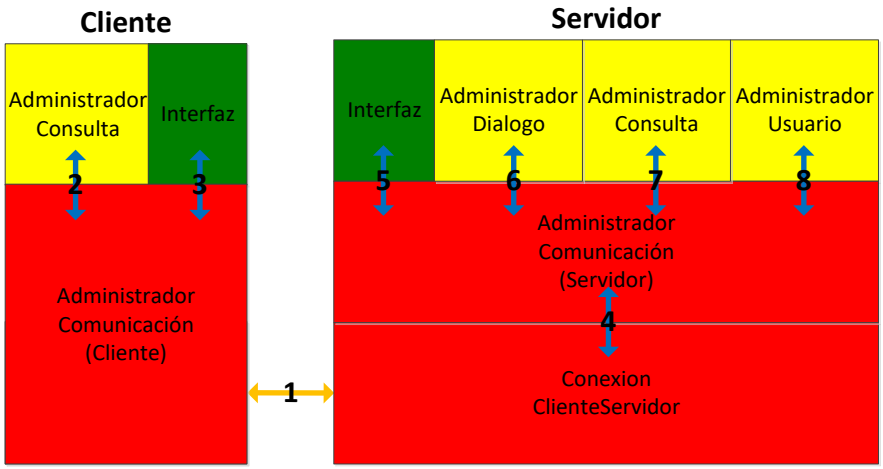

Ilustración 4.5: Clases Del *FDJ* Organizadas En Capas
Fuente: El Autor

En esta estructuración, se puede observar claramente las capas que se caracterizaron en la sección 4.1. Es decir: las clases de color ***verde*** corresponden a la capa de ***interfaces de usuario*** –que no será considerada en este análisis, de conformidad con lo planeado en la sección 4.1-, las clases de color ***amarillo*** se asocian a las de ***gestión del diálogo***, en tanto que las de color ***rojo*** se relacionan a las ***comunicaciones***. Como antes, las últimas dos clases son el objeto de estudio de este proyecto.

Por otro lado, la disposición de los objetos es cónsona con los conceptos desarrollados en la sección 2.7.2, relacionada con la normalización de protocolos.

En otro orden de cosas, las coloraciones que se han dado a las flechas tienen una connotación importante. El color naranja representa comunicación a nivel de protocolo, en tanto que el color azul representa comunicación vía interface de servicios. Se recuerda que, de acuerdo a lo planteado en la sección 2.7.4, estos conceptos no son equivalentes.

Además, es relevante recordar que, de acuerdo a todo lo planteado previamente en esta investigación, se tiene por sentado que, a nivel interno, el

diálogo entre clientes siempre se almacena primero en el servidor, y es esta aplicación la que se encarga de enviar las actualizaciones de vista a cada cliente.

En consecuencia, se procederá a analizar el intercambio de mensajes entre las capas de comunicación de la aplicación –las que se encuentran identificadas con el color rojo-, planteando sus interfaces funcionales, que definen a las primitivas de servicio que interrelacionan a los clientes y el servidor.

4.3.5 Interfaces Funcionales De Intercambio De Información Del *FDJ*, En Términos De Primitivas De Servicio.

Para facilitar su comprensión, se han organizado seis interfaces de comunicación entre las clases o capas que integran al ***FDJ***, de acuerdo a lo analizado en la sección 4.3.4. en función a las capas o clases que intervienen en el intercambio de mensajes, de donde se establecen sus nombres. Los números que las acompañan entre paréntesis al final, corresponden a la denominación establecida en la Ilustración 4.5. Ellas son:

- Administrador De Comunicaciones (Cliente) – Conexión Cliente/Servidor (1).
- Administrador De Comunicaciones (Cliente) – Administrador De Consultas (Cliente) (2).
- Conexión Cliente/Servidor – Administrador De Comunicaciones (Servidor) (4).

- Administrador De Comunicaciones (Servidor) – Administrador De Diálogo (6).
- Administrador De Comunicaciones (Servidor) – Administrador De Consultas (Servidor) (7).
- Administrador De Comunicaciones (Servidor) – Administrador De Usuarios (8).

De todas las capas antes mencionadas, de interés para los fines de esta investigación, resulta la que relaciona al administrador de comunicaciones del cliente con la conexión cliente servidor del servidor. En consecuencia, en las secciones siguientes de esta investigación, se procederá a establecer los grupos de métodos que corresponden a las primitivas de servicio presentes en dicha categoría, de acuerdo a lo señalado en la sección 2.7.3.

Esta organización se planteará con el auxilio de tablas donde se indicará, para cada método pertinente[14], su tipo – ya sea como solicitud, indicación, respuesta, o confirmación-, así como si corresponden a servicios confirmados o no confirmados, entre otros aspectos. Además, se seguirá la nomenclatura establecida en **[TANE03]**, para identificar a las primitivas.

---

[14] Se debe recalcar que en los análisis de esta investigación, se consideran únicamente los métodos que son relevantes para el estudio.

4.3.5.1 Administrador De Comunicaciones (Cliente) – Conexión Cliente/Servidor (1).

La tabla que se presenta a continuación, corresponde a la interfaz existente entre las clases ***AdministradorComunicacionCliente*** y ***ConexionClienteServidor***:

Clase	Método	Nombre Del Servicio / Primitiva	Tipo De Primitiva				Tipo De Servicio		Justificación
			Solicitud	Indicación	Respuesta	Confirmación	Confirmado	No Confirmado	
ConexionClienteServidor	ConexionClienteServidor (int indice, Socket socket, String dir_conex, int act)	Iniciar_sesion.solicitud	X					X	**Constructor:** solicita el inicio de sesión de algún cliente en el foro.
AdministradorComunicacionCliente	AdministradorComunicacionCliente(String dir_server, int port_server, String nombre)	Iniciar_sesion.indicacion		X				X	**Constructor**: acepta el inicio de sesión que ofrece un servidor del foro.
AdministradorComunicacionCliente	enviar(String cad)	Enviar_mensaje_servidor.solicitud	X					X	**Método:** envía mensajes al servidor del foro, desde el cliente
ConexionClienteServidor	refresh(String S)	Enviar_mensaje_servidor.indicacion		X				X	**Método:** recibe los mensajes que envían los clientes, en el servidor
ConexionClienteServidor	enviar_cad(String cad_usu)	Enviar_mensaje_cliente.solicitud	X					X	**Método:** envía mensajes al cliente del foro, desde el servidor
AdministradorComunicacionCliente	refresh(String S)	Enviar_mensaje_cliente.indicación		X				X	**Método:** recibe los mensajes que envía el servidor, en el cliente
AdministradorComunicacionCliente	desconectar()	Finalizar_sesion.solicitud	X					X	**Método:** solicita la finalización de la sesión del cliente, en el servidor
ConexionClienteServidor	cerrarSocket()	Finalizar_sesion.indicacion		X				X	**Método:** cierra la comunicación vía socket entre el cliente y el servidor

Tabla 4.5: Interface De Comunicación Administrador De Comunicaciones (Cliente) – Conexión Cliente/Servidor
Fuente: El Autor

4.3.6 Fortalezas Y debilidades Observadas En la Interface Analizada.

Del análisis de la Tabla 4.5, se puede observar que surgen ocho primitivas de servicio bien definidas. A saber:

- Iniciar_sesion.solicitud
- Iniciar_sesion.indicacion
- Enviar_mensaje_servidor.solicitud
- Enviar_mensaje_servidor.indicacion
- Enviar_mensaje_cliente.solicitud
- Enviar_mensaje_cliente.indicación
- Finalizar_sesion.solicitud
- Finalizar_sesion.indicacion

En todos los casos, se puede observar que las primitivas planteadas responden a los casos de solicitud e indicación. Es decir, corresponden a servicios no confirmados, de acuerdo a la sección 2.7.3.

Sin embargo, de acuerdo a lo que se establece en **[TANE03]**, los servicios del tipo inicio de sesión siempre deben ser confirmados. Dicho de otro modo, se deben identificar dos métodos adicionales –uno en la clase ***AdministradorComunicacionCliente*** y otro en la clase ***ConexionClienteServidor***-, que se encarguen de completar el ciclo de conexión.

Sin embargo, analizando el código fuente de la aplicación, se puede observar, por un lado, la ausencia de los métodos antes mencionados, en tanto que por otro lado, se detecta una debilidad muy seria en la aplicación, específicamente en las clases ***AdministradorComunicacionCliente*** y ***ConexionClienteServidor***, ya que entre ellas, se presenta un manejo deficiente de las solicitudes de conexión que deben ser rechazadas.

De hecho, al realizar pruebas donde se trataba de ingresar con nombres de usuario previamente registrados en la sesión, la aplicación rechazaba la solicitud con el nombre duplicado, pero a su vez, finaliza la sesión del usuario que tenía el nombre asignado inicialmente –situación que no se debe presentar-.

Ambos comportamientos evidencian debilidades dentro del ***FDJ***, que deben ser corregidas. Se considera que la ausencia de los métodos antes mencionados que completarían el ciclo de inicio de sesión, son el origen del problema antes mencionado.

CAPÍTULO 5

CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES.

Este capítulo presenta la culminación de esta investigación. En tal sentido:

1. Se formulan las conclusiones que sintetizan a este proyecto.

2. Se plantean las recomendaciones que surgen de esta investigación.

3. Se sugieren algunos proyectos adicionales que pueden realizarse como productos derivados de esta investigación.

4. Se exponen las contribuciones realizadas por este estudio.

5.1 CONCLUSIONES.

Al completar este proyecto de investigación, se obtienen las conclusiones que se plantean a continuación, establecidas en función al análisis de la documentación del ***FDJ***, y de acuerdo a lo planteado por **[DUTA01]**:

1. Desde el punto de vista de su arquitectura interna, el ***FDJ*** se plantea como una aplicación cliente/servidor bien marcada. Esto en función a la diferenciación marcada de funciones que se observa entre el cliente y la aplicación, que se puede observar en la sección 4.1 de este documento.

2. De conformidad con las secciones 4.2.2.2 y 4.2.2.4, el ***FDJ***, posee dos sistemas de mensajes simples y complejos, ambos basados en tramas de longitud y composición variables, que en sí, son funcionales a nivel

de la aplicación, ya que le ofrecen un nivel de flexibilidad significativo en cuanto a expresividad de intercambio de información. Sin embargo, a su vez, también plantean una serie de peligros potenciales en la estabilidad del sistema, por el riesgo de bloqueo de la aplicación cuando la longitud de los mensajes crece indiscriminadamente.

3. Analizando funcionamiento interno del ***FDJ***, en términos de su arquitectura de comunicaciones –ver sección 4.3-, se puede observar que hay varios aspectos estructurales y funcionales que presentan situaciones controversiales a nivel de sintaxis, semántica, temporización, servicios e intercambio de información, a saber:

   - **Posee una sintaxis bien definida:** cada uno de los campos que integran a sus ***PDU's***, tanto a nivel de los mensajes simples como de los complejos, por cuanto que cada tipo de mensaje tiene bien definida su estructura interna, en términos de las secuencias de campos y la forma en que se organizan e identifican.

   - **Posee una semántica clara en sus campos:** en función al problema de comunicación que se modela, este criterio se satisface.

   - **Tiene serios problemas de temporización:** que se atribuyen, esencialmente, al empleo de tramas de longitud variable y representan una debilidad grave en su diseño.

   - **El encapsulamiento tiene una implementación incompleta:** ya que no posee mecanismos para controlar los errores en la transmisión de los mensajes, además de que el direccionamiento

se implementa a nivel de gestión de procesos dentro de las aplicaciones.

- **Emplea paquetes de datos de longitud variable:** Por lo que los medios de comunicación compartidos pueden ser acaparados fácilmente por dicha aplicación. Es decir, la longitud variable de las tramas dificulta la intercomunicación entre clientes y servidores, ya que disminuye la eficiencia del canal de comunicación, por la multiplexación deficiente del medio de transmisión, por parte de la aplicación, lo que pude conducir a que la aplicación se cuelgue con facilidad.

- **El control de la conexión no cumple los diálogos de tres partes:** por lo que, a nivel de la aplicación, no se puede corroborar la recepción de paquetes enviados.

- **Sus mensajes no contienen datos relativos a secuenciación:** Es decir, al orden en que deben ser interpretadas por el equipo que los recibe y decodifica, ya que dichos paquetes, no se envían segmentados. No obstante, por diseño, siguen una disciplina estricta ***FIFO***, lo que puede mitigar parcialmente el problema.

- **No posee mecanismos de control de flujo:** de modo que el sistema de comunicaciones no ofrece un mecanismo fiable para controlar la sobrecarga en el tráfico entre los sistemas clientes y el servidor.

- **Dichos paquetes no contienen información relacionada a la corrección o detección de errores:** Lo que significa que no hay mecanismos que permitan establecer la presencia o ausencia de errores en el proceso de transmisión y se debe confiar en que

los protocolos de nivel inferior realizarán esta tarea de manera exclusiva. Adicionalmente, lo que es peor, ante cualquier tipo de falla de comunicación, los paquetes erróneos no serán retransmitidas.

- **Esquema de multiplexación deficiente:** ya que el protocolo se implementa sobre la base de multiplexación básica o uno a uno, lo que representa un uso ineficiente de los enlaces de transmisión de datos de banda ancha.

- **El direccionamiento se implementa a nivel de la aplicación:** el protocolo como tal, no posee este tipo de información.

- **El protocolo confía el encaminamiento de los paquetes a protocolos de nivel inferior:** lo que según **[KURO07]**, no representa una debilidad en sí.

- **No se implementan características que permitan alterar la prioridad en el envío de paquetes:** por lo que nuevamente, esta característica la deben ofrecer los protocolos de nivel inferior, si está disponible.

- **El intercambio de información no está cifrado:** De modo que cualquier intruso puede intersectar los paquetes de datos que intercambia el servidor y los clientes, lo que implica que es una aplicación no segura para el canje de datos confidenciales. Esta es una de las debilidades más significativa del protocolo.

4. El análisis del ***FDJ***, basado en capas, evidenció que la aplicación no tiene una organización eficiente, ya que: los mensajes que se intercambian entre los clientes y el servidor se estructuran en las clases

de administración –consultas, dialogo o usuarios-, en tanto que los administradores de comunicaciones se encargan sólo de iniciar y finalizar las sesiones, así como de intercambiar dichos mensajes.

5. Adicionalmente, se puede establecer que la implementación incompleta de las primitivas de servicio, a nivel de inicio de sesión, provoca que los mecanismos que se utilizan en la aplicación para controlar el inicio o finalización de la sesión de los usuarios no sean completamente fiables.

6. En consecuencia, es evidente que el protocolo de comunicaciones del ***FDJ*** presenta debilidades en cuanto a su conceptualización, diseño e implementación que se deben corregir.

5.2 RECOMENDACIONES.

Al completar este proyecto de investigación, se plantean las recomendaciones que se detallan a continuación:

1. Se debe profundizar y ampliar la documentación disponible relacionada a la arquitectura y funcionamiento del ***FDJ***, así como de su protocolo de comunicaciones, en función de facilitar estudios posteriores relacionados a la temática en particular.

2. Se deben corregir los problemas de temporización del protocolo, con la inclusión de algún mecanismo de control de flujo, tal como un esquema de ventanas deslizantes, para controlar la sobrecarga en el tráfico de mensajes.

3. Es recomendable que se incorporen en el protocolo los mecanismos necesarios para controlar los errores de transmisión de mensajes, así

como información de direccionamiento, directamente en las tramas que se intercambian, en vez de gestionarlas vía administración de procesos, a lo interno de la aplicación.

4. Debe incluirse un nivel adicional de procesamiento, de modo que se pueda implementar algún tipo de esquema de segmentación y ensamblado de paquetes. Esta reforma mejorará a la gestión de la aplicación, en términos de disminuir sus posibilidades de bloqueo por saturación del enlace de comunicación, vía el empleo de esquemas de multiplexación más eficientes que el trivial.

5. Se deben incluir mecanismos eficientes de secuenciación de paquetes, así como información que posibilite la detección o corrección de errores en la transmisión de mensajes, así como el cumplimiento cabal de los diálogos de tres partes, para controlar cabalmente las conexiones establecidas vía el protocolo.

6. Es recomendable incluir, en el protocolo, características que permitan ofrecer calidad de servicio en los intercambios de mensajes, así como algún esquema de cifrado en los paquetes, de modo que se ofrezca algún nivel de privacidad en la transferencia de información confidencial.

7. Se sugiere que en actualizaciones futuras del foro, se considere un mejor nivel de modularidad y estructuración de capas o niveles. Esta mejora tiene como objetivo, que se puedan caracterizar, con mayor claridad, las interfaces de servicios y las ***PDU's*** involucradas en los intercambios de mensajes de cada capa, así como lograr la implementación de servicios confirmados en los intercambios de

mensajes entre ellas; así como corregir los problemas de inicio y finalización de sesiones de usuarios con nombres duplicados.

8. Se recomienda completar el proyecto de estandarización del protocolo de aplicación del ***FDJ***, completando las fases de diseño e implementación de las normas analizadas en este estudio.

Las recomendaciones y mejoras antes descritas, pueden ser tomadas como punto de partida para algunos de los trabajos de investigación, que se derivan de los resultados de esta investigación.

5.3 CONTRIBUCIONES.

Haciendo una comparación de los objetivos propuestos en la sección 1.4 de esta investigación, frente a los resultados alcanzados al culminar el proyecto, se obtienen las contribuciones que se mencionan a continuación:

1. **El marco teórico que fundamenta el análisis de protocolos de comunicaciones, a nivel de primitivas de servicio:** esta meta se logró ya que, en la sección CAPÍTULO 2: de este documento, se exponen los aspectos más relevantes de estas teorías, en especial, la forma en que se debe plantear el intercambio de mensajes entre las capas e interfaces de los protocolos de comunicación, vía las categorías de primitivas de servicios disponibles. Es un documento bastante compacto en cuanto a extensión, pero completo en cuanto a la cobertura de los temas considerados.

2. **La comprensión de la arquitectura de comunicación que emplea el *FDJ*:** esta meta también se alcanzó, en las secciones 4.1 y 4.2, de este proyecto, al momento en que se analizó tanto la arquitectura de las

clases de la aplicación –relevantes para la investigación-, como al momento en que se analizaron los mecanismos que emplea para estructurar el intercambio de mensajes entre clientes y servidor.

3. **El contraste de la arquitectura de comunicaciones del *FDJ*, ante los elementos estructurales y funcionales que fundamentan el estudio de los protocolos de comunicaciones:** este objetivo también fue alcanzado, ya que en las secciones 4.3.1 y 4.3.2 de este documento, se expone este análisis, aunque se evidencian varias debilidades importantes que tiene esta aplicación, al momento de observar el cumplimiento de los estándares establecidos en la materia.

4. **La caracterización de las funciones que utiliza el *FDJ* para intercambiar información, en términos de primitivas de servicio:** este objetivo también se alcanzó, en las secciones 4.3.4, 4.3.5 y 4.3.6, aunque nuevamente, surgieron otras situaciones en las que el sistema se aleja de los estándares que norman la materia.

5. **Parte de la documentación que se debe elaborar al momento de implementar un protocolo de comunicaciones:** en efecto, todo el análisis planteado sobre el ***FDJ***, es un modelo de documentación parcial que se debe recabar al momento que se desea plantear la estandarización del sistema de comunicaciones que utiliza una aplicación cliente/servidor, bajo las normas de los modelos ***OSI*** o ***TCP/IP***

REFERENCIAS BIBLIOGRÁFICAS

[AGÜE87] **AGÜERO, Ulises.** *A Theory Of Plausibility For Computer Architecture Design.* Ph.D. thesis, Center for Advanced Computer Studies, University of Southwestern Lousiana, EUA, 1987.

[COME00] **COMER, Douglas E.** *Internetworking With TCP-IP Vol 1: Principles, Protocols And Architecture.* 4º Edition, Prentice Hall, USA, 2000.

[DAYJ08] **DAY, John D.** *Patterns In Network Architecture: A Return To Fundamentals.* 1º Edition, Pearson, USA, 2008.

[DUTA01] **DUTARI DUTARI, Raúl Enrique.** *Implementación De Un Foro De Discusión Jerárquico, Basado En XML, Con Consideraciones De Plausibilidad.* Tesis para optar al grado de Magíster Scientiae en Computación en el Instituto Tecnológico de Costa Rica, Costa Rica, 2001.

[ECKE07] **ECKEL, Bruce.** *Piensa En Java*. 1º Edición, Pearson, España, 2007.

[FOCH06] **FOROUZAN, Behrouz A.; CHUNG FEGAN, Sophia.** *TCP-IP Protocol Suite*. 3º Edition, McGraw-Hill, USA, 2006.

[FORO07] **FOROUZAN, Behrouz A.** *Data Communications And Networking.* 4º Edition, McGraw-Hill, USA, 2007.

[HEFE06] **HERNÁNDEZ SAMPIERI, Roberto, FERNÁNDEZ COLLADO, Carlos. Y BAPTISTA LUCIO, Pilar.** *Metodología de la Investigación.* 4º Edición. McGraw-Hill. México, 2006.

[KOZI05] **KOZIEROK, Charles M.** *The TCP/IP Guide.* Version 3.0, Canada, 2005.

[KURO07] **KUROSE, James F.; ROSS, Keith W.** *Computer Networking: A Top-Down Approach.* 4º Edition, Pearson, 2007.

[NÚÑE99] **NÚÑEZ MARÍN, Giannina.** *Toma De Decisiones En Grupo Para Juegos Educativos Colaborativos.* Tesis para optar al grado de Magíster Scientiae en Computación en el Instituto Tecnológico de Costa Rica, Costa Rica,1999.

[PRES05] **PRESSMAN, Roger.** *Ingeniería Del Software: Un Enfoque Práctico.* 6º Edición, McGraw-Hill, España, 2005.

[STAL04] **STALLINGS, William.** *Comunicaciones Y Redes De Computadoras.* 7º Edición, Pearson, España, 2004.

[STPO02] **STEVENS, Perdita; POOLEY, Rob.** *Utilización De UML En Ingeniería Del Software Con Objetos Y Componentes.* 1º Edición, Pearson, España, 2002.

[TANE03] **TANENBAUM, Andrew S.** *Computer Networks.* 4º Edición. Pearson. U.S.A. 2003.

[UITT93] **UNIÓN INTERNACIONAL DE TELECOMUNICACIONES.** *Tecnología De La Información -Interconexión De Sistemas Abiertos- Modelo De Referencia Básico: Convenios Para La Definición De Servicios En La Interconexión De Sistemas Abiertos: Recomendación UIT-T X.210.* Unión Internacional De Telecomunicaciones, Sector Normalización. Fecha De Actualización: 1993-11-16. Fecha De Consulta: 2009-02-01. Disponible En: http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.210-199311-I!!PDF-S&type=items.

APÉNDICE: TRAZA DEL DIÁLOGO REALIZADO ENTRE DOS CLIENTES Y UN SERVIDOR DEL FDJ

A continuación, se presentará una tabla donde se encuentra la traza del dialogo entre dos clientes y un servidor del ***FDJ***, obtenido mediante la inclusión de código adicional en los sistemas que imprimía en consola los mensajes que intercambiaban las aplicaciones.

Los mensajes se han colocado en la tabla de manera que se respete, hasta donde es posible, la secuenciación original del intercambio de información. Además, se deberá entender que el tráfico de información siempre se inicia y termina en los clientes del foro. Se inicia con envíos de mensajes al servidor, y culmina cuando los clientes reciben los mensajes de vuelta del servidor.

Acciones De Las Aplicaciones		
Servidor	**Usuario #1**	**Usuario #2**
Recibida: **Usuario #1**	Enviada: **Usuario #1**	
Enviada: **M/Usuario #1 se ha conectado exitosamente**	Recibida: **M/Usuario #1 se ha conectado exitosamente**	
Enviada: **T/Alternativas para salvar al planeta tierra**	Recibida: **T/Alternativas para salvar al planeta tierra**	
Recibida: **Usuario #2**		Enviada: **Usuario #2**
Enviada: **M/Usuario #2 se ha conectado exitosamente**	Recibida: **M/Usuario #2 se ha conectado exitosamente**	
Enviada: **M/Usuario #2 se ha conectado exitosamente**		Recibida: **M/Usuario #2 se ha conectado exitosamente**
Enviada: **T/Alternativas para salvar al planeta tierra**		Recibida: **T/Alternativas para salvar al planeta tierra**
Recibida: **A/4/41/10/1/nullSe debe controlar el calentamiento globalUsuario #2O**		Enviada: **A/4/41/10/1/nullSe debe controlar el calentamiento globalUsuario #2O**
Enviada: **M/El enunciado se agregó satisfactoriamente.**		Recibida: **M/El enunciado se agregó satisfactoriamente.**
Enviada: **I/InicioTrama** Enviada: **S/Sobreescribir** Enviada: **E/2/1/41/10/15/11/9/6/6/4/1/1.OSe debe controlar el calentamiento globalUsuario #22011/Octubre/2800:25:18 AMAceptableBlanco1.0000trueO** Enviada: **F/FinTrama**	Recibida: **I/InicioTrama** Recibida: **S/Sobreescribir** Recibida: **E/2/1/41/10/15/11/9/6/6/4/1/1.OSe debe controlar el calentamiento globalUsuario #22011/Octubre/2800:25:18 AMAceptableBlanco1.0000trueO** Recibida: **F/FinTrama**	
Enviada: **I/InicioTrama** Enviada: **S/Sobreescribir** Enviada: **E/2/1/41/10/15/11/9/6/6/4/1/1.OSe debe controlar el calentamiento globalUsuario #22011/Octubre/2800:25:18 AMAceptableBlanco1.0000trueO** Enviada: **F/FinTrama**		Recibida: **I/InicioTrama** Recibida: **S/Sobreescribir** Recibida: **E/2/1/41/10/15/11/9/6/6/4/1/1.OSe debe controlar el calentamiento globalUsuario #22011/Octubre/2800:25:18 AMAceptableBlanco1.0000trueO** Recibida: **F/FinTrama**
Recibida: **A/2/38/10/1/1.¿Pero y qué del crecimiento económico?Usuario #1C**	Enviada: **A/2/38/10/1/1.¿Pero y qué del crecimiento económico?Usuario #1C**	
Enviada: **M/El enunciado se agregó satisfactoriamente.**	Recibida: **M/El enunciado se agregó satisfactoriamente.**	

Acciones De Las Aplicaciones		
Servidor	**Usuario #1**	**Usuario #2**
Enviada: **I/InicioTrama** Enviada: **S/Sobreescribir** Enviada: **E/2/1/41/10/15/11/15/6/6/5/1/1.DSe debe controlar el calentamiento globalUsuario #22011/Octubre/2800:25:18 AMInsatisfactorioBlanco0.5000falseO** Enviada: **E/4/1/38/10/15/11/9/6/6/4/1/1.1.C¿Pero y qué del crecimiento económico?Usuario #12011/Octubre/2800:35:08 AMAceptableBlanco1.0000trueC** Enviada: **F/FinTrama**	Enviada: **I/InicioTrama** Enviada: **S/Sobreescribir** Enviada: **E/2/1/41/10/15/11/15/6/6/5/1/1.DSe debe controlar el calentamiento globalUsuario #22011/Octubre/2800:25:18 AMInsatisfactorioBlanco0.5000falseO** Enviada: **E/4/1/38/10/15/11/9/6/6/4/1/1.1.C¿Pero y qué del crecimiento económico?Usuario #12011/Octubre/2800:35:08 AMAceptableBlanco1.0000trueC** Enviada: **F/FinTrama**	
Enviada: **I/InicioTrama** Enviada: **S/Sobreescribir** Enviada: **E/2/1/41/10/15/11/15/6/6/5/1/1.DSe debe controlar el calentamiento globalUsuario #22011/Octubre/2800:25:18 AMInsatisfactorioBlanco0.5000falseO** Enviada: **F/FinTrama**		Recibida: **I/InicioTrama** Recibida: **S/Sobreescribir** Recibida: **E/2/1/41/10/15/11/15/6/6/5/1/1.DSe debe controlar el calentamiento globalUsuario #22011/Octubre/2800:25:18 AMInsatisfactorioBlanco0.5000falseO** Recibida: **F/FinTrama**
Recibida: **S/**	Enviada: **S/**	
Enviada: **M/Usuario #1 se retiró de la sesión...**		Recibida: **M/Usuario #1 se retiró de la sesión...**
Enviada: **M/Usuario #1 finalizó su sesion en el foro...**	Recibida: **M/Usuario #1 finalizó su sesion en el foro...**	
Recibida: **S/**		Enviada: **S/**
Enviada: **M/Usuario #2 finalizó su sesion en el foro...**		Recibida: **M/Usuario #2 finalizó su sesion en el foro...**

Tabla 5.1: Traza Del Diálogo Realizado Entre Dos Clientes Y un Servidor Del *FDJ*
Fuente: El Autor

ANEXO: DETALLE DE LOS CAMPOS QUE INTEGRAN A LOS MENSAJES SIMPLES QUE EMPLEA EL FDJ

A continuación, se presenta el detalle de los campos de integran a los mensajes simples del ***FDJ***, de acuerdo a **[DUTA01]**.

Mensaje	Cadena	Componentes
Tema de Discusión	T/DATA	✓ T: "T"ema de discusión ✓ /: delimitador ✓ DATA: texto del mensaje
Inicio de Mensaje Complejo	I/InicioTrama	✓ I: "I"nicio de una trama ✓ /: delimitador ✓ InicioTrama: texto del mensaje
Reemplazar Enunciados	R/Reemplazar	✓ R: "R"eemplazar los enunciados existentes en el cliente, por los enunciados contenidos en este mensaje, exclusivamente. Este mensaje se utiliza cuando el cliente cambia el tipo de consulta que realiza ✓ /: delimitador ✓ Reemplazar: texto del mensaje
Fin de Mensaje Complejo	I/FinTrama	✓ F: "F"in de trama ✓ /: delimitador ✓ FinTrama: texto del mensaje
Mensajes de la Línea de Estado	M/DATA	✓ M: "M"ensaje enviado ✓ /: delimitador ✓ DATA: Texto del mensaje
Anexar Enunciados	A/Anexar	✓ A: "A"nexar los enunciados contenidos en el mensaje, a los enunciados existentes en el cliente. En caso que algún enunciado se encuentre dentro de la consulta, se sobrescribe. Este mensaje es utilizado cuando un cliente agrega un enunciado y se requiere dicho enunciado aparezca en la consulta del cliente – normalmente, el mismo usuario que lo agregó, o alguno otro que debe observarlo, con base en la consulta que realiza ✓ /: delimitador

Mensaje	Cadena	Componentes
		✓ Anexar: texto del mensaje
Sobrescribir Enunciados	S/Sobrescribir	✓ S: “S”obrescribir los enunciados existentes en el árbol, con los enunciados que contenidos en el mensaje. En caso que algún enunciado no se encuentre dentro de la consulta, se anexa. Este mensaje es utilizado cuando un cliente modifica un enunciado y se requiere actualizar las vistas de los usuarios del foro, con los enunciados que resultaron alterados. ✓ /: delimitador ✓ Sobrescribir: texto del mensaje
Enunciado	E/1/2/3/4/5/6/7/8/9/10/11/DATA	✓ E: “E”nunciado ✓ /: delimitador ✓ 1/../11: representan las 11 componentes que integran a un enunciado que se envía a los clientes para integrarlos a sus consultas. ✓ DATA: mensaje que se envía al cliente, está compuesto por la concatenación de las 11 componentes que integran a un enunciado que se envía a los clientes.

Tabla 5.2: Mensajes Simples Que Envía El Servidor A Los Clientes Del *FDJ*
Fuente: [DUTA01]

Mensaje	Cadena	Componentes
Evidencia Asociada de un Enunciado	E/LLAVE	✓ E: “E”videncia asociada de un enunciado ✓ /: delimitador ✓ LLAVE: es la llave o identificador que distingue al enunciado dentro del diálogo.
Cambiar de Consulta	V/DATA	✓ V: “V”ista solicitada ✓ /: delimitador ✓ DATA: nombre de la vista solicitada
Salir de la sesión	S/	✓ S: “S”alir de la sesión ✓ /: delimitador

Mensaje	Cadena	Componentes
Agregar Enunciado	A/1/2/3/4/DATA	✓ A: “A”gregar Enunciado ✓ /: delimitador ✓ 1/../4: representan las 4 componentes que integran a un enunciado que se envía desde un cliente hacia el servidor, para agregarlo al diálogo. ✓ DATA: mensaje que se envía al cliente, está compuesto por la concatenación de las 4 componentes que integran a un enunciado que se envía al servidor, para agregarlo al diálogo.
Modificar Enunciado	M/1/2/3/4/5/6/DATA	✓ M: “M”odificar Enunciado ✓ /: delimitador ✓ 1/../6: representan las 6 componentes que integran a un enunciado que se envía desde un cliente hacia el servidor, para modificar al diálogo. ✓ DATA: mensaje que se envía al cliente, está compuesto por la concatenación de las 6 componentes que integran a un enunciado que se envía al servidor, para modificar al diálogo.

Tabla 5.3: Mensajes Simples Que Envían Los Clientes Al Servidor Del *FDJ*
Fuente: [DUTA01]

ANEXO: DIAGRAMAS DE CLASES ESTÁTICAS DEL CLIENTE Y EL SERVIDOR DEL FDJ

En este anexo se presentan los diagramas de clases estáticas del cliente y del servidor del ***FDJ***, de acuerdo a **[DUTA01]**.

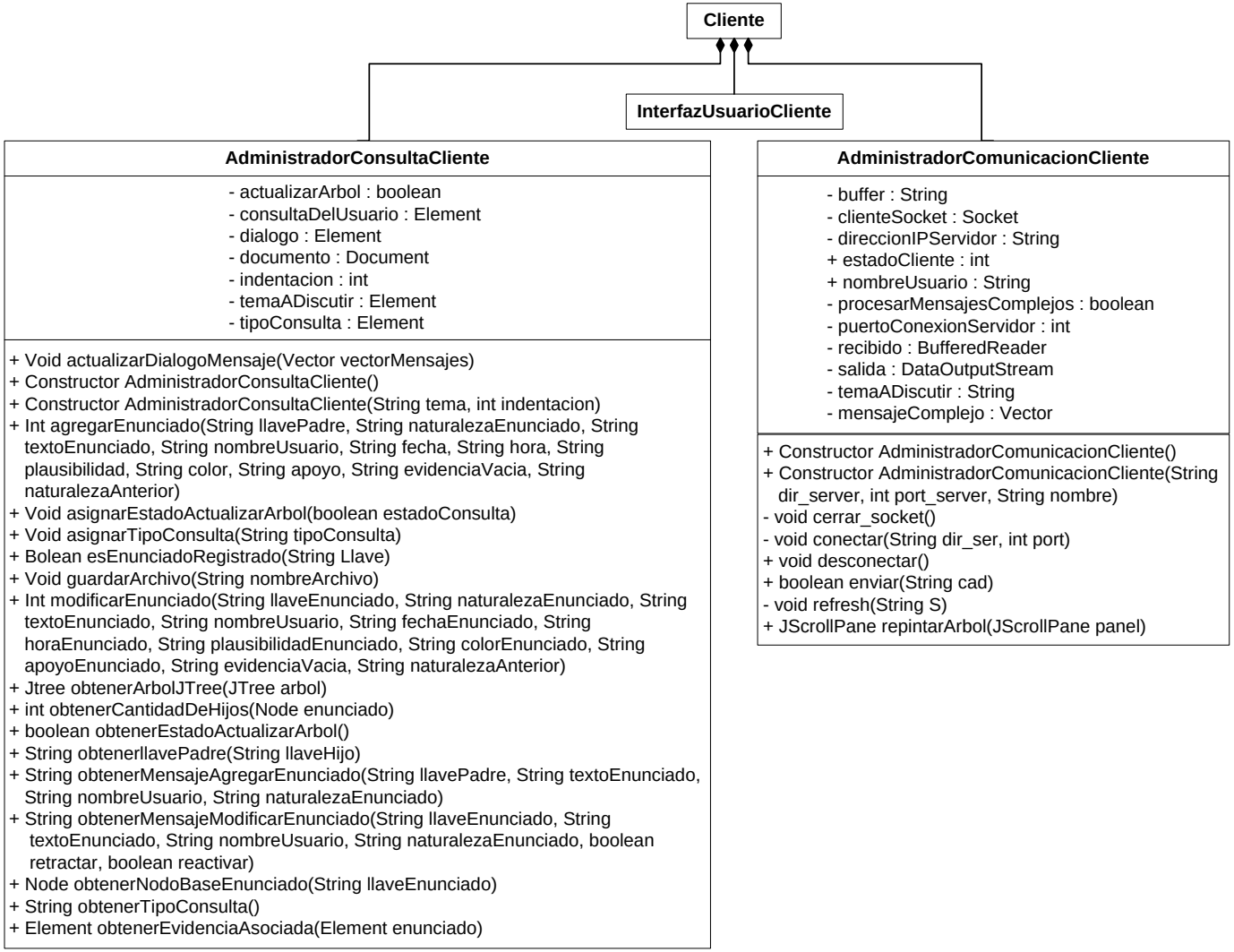

Ilustración 5.1: Diagrama De Clases Estáticas: Cliente Del *FDJ*
Fuente: [DUTA01]

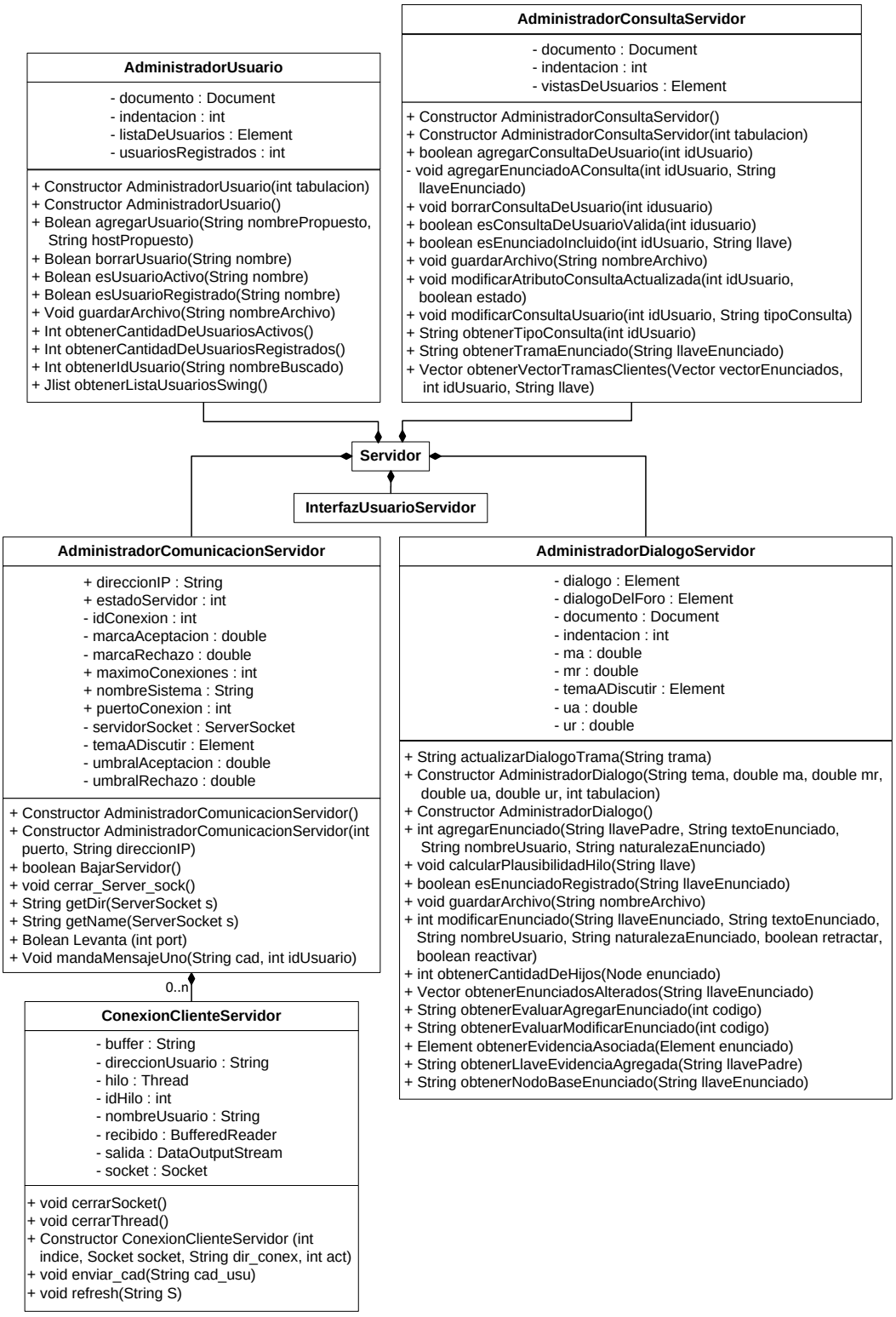

Ilustración 5.2: Diagrama De Clases Estáticas: Servidor Del *FDJ*
Fuente: [DUTA01]

ANEXO: DESCRIPCIÓN DE LOS ATRIBUTOS PÚBLICOS DE LOS DIAGRAMAS DE CLASES ESTÁTICAS DEL CLIENTE Y EL SERVIDOR DEL FDJ

En este anexo se presentan las descripciones y significados de los atributos públicos que disponen los diagramas de clases estáticas del cliente y del servidor del ***FDJ***, mencionadas en la sección 4.1.de acuerdo a **[DUTA01]**, a través de las tablas que se muestran a continuación.

Atributo	Alcance	Tipo	Explicación
direccionIP	público	String	Representa la dirección DNS o IP del servidorsistema que está conectado al socket
estadoServidor	público	int	Representa la disponibilidad del servidor a recibir conexiones (0 = servidor desactivado, 1 = servidor activado)
idConexion	privado	int	Representa el identificador del hilo de ejecución asignado a un cliente particular
marcaAceptacion	privado	double	Representa el valor que asumirá la marca de aceptación para una sesión particular del foro.
marcaRechazo	privado	double	Representa el valor que asumirá la marca de rechazo para una sesión particular del foro.
maximoConexiones	público	int	Representa la cantidad máxima de usuarios activos que pueden presentarse concurrentemente
nombreSistema	público	String	Representa el nombre del sistema donde se ejecuta el servidor
puertoConexion	público	int	Representa el puerto de comunicaciones que se utilizará al momento de establecer la conexión entre el cliente y el servidor
servidorSocket	privado	ServerSocket	Representa el servidor de conexiones (Socket de Flujo, orientado a conexión)

Atributo	Alcance	Tipo	Explicación
temaADiscutir	privado	Element	Representa al elemento que contiene el tema que se discute dentro del foro.
umbralAceptacion	privado	double	Representa el valor que asumirá el umbral de aceptación para una sesión particular del foro.
umbralRechazo	privado	double	Representa el valor que asumirá el umbral de rechazo para una sesión particular del foro.

Tabla 5.4: Atributos De La Clase *AdministradorComunicacionServidor* Del Servidor En El Diagrama De Clase Del *FDJ*
Fuente: [DUTA01]

Atributo	Alcance	Tipo	Explicación
buffer	privado	String	Representa la cadena de texto que se recibe del cliente y debe ser procesada
direccionUsuario	privado	String	Representa la dirección DNS o IP del sistema que está conectado al socket
hilo	privado	Thread	Representa el hilo de ejecución asignado a un cliente particular
idHilo	privado	int	Representa el identificador del hilo de ejecución asignado a un cliente particular
nombreUsuario	privado	String	Representa el nombre del usuario que está conectado al socket
recibido	privado	BufferedReader	Representa el flujo de datos que entra al servidor, desde el sistema del cliente
salida	privado	DataOutputStream	Representa el flujo de datos que sale del servidor, y se dirije hacia el sistema del cliente
socket	privado	Socket	Representa la conexión asignada a un cliente particular (Socket de Flujo, orientado a conexión)

Tabla 5.5: Atributos De La Clase *ConexionClienteServidor* Del Servidor En El Diagrama De Clase Del *FDJ*
Fuente: [DUTA01]

Atributo	Alcance	Tipo	Explicación
buffer	privado	String	Representa la cadena de texto que se recibe del cliente y debe ser procesada
clienteSocket	privado	Socket	Representa la conexión del cliente con el servidor (Socket de Flujo, orientado a conexión)
direccionIPServidor	privado	String	Representa la dirección DNS o IP del servidor al que se conecta el cliente

Atributo	Alcance	Tipo	Explicación
estadoCliente	público	int	Representa la disponibilidad de la conexión que ofrece el servidor (0 = servidor desactivado, 1 = servidor activado)
nombreUsuario	público	String	Representa el nombre del usuario se une a la sesión
procesarMensajesComplejos	privado	boolean	Representa una bandera que señala cuando se ha terminado de recibir los mensajes complejos
puertoConexionServidor	Privado	int	Representa el puerto de comunicaciones que se utilizará al momento de establecer la conexión con el servidor
recibido	Privado	BufferedReader	Representa el flujo de datos que entra desde servidor
salida	Privado	DataOutputStream	Representa el flujo de datos que sale del cliente, y se dirige hacia el servidor del foro
temaADiscutir	Privado	String	Representa al tema que se discute dentro del foro.
mensajeComplejo	Privado	Vector	Representa a los mensajes complejos que se reciben del servidor

Tabla 5.6: Atributos De La Clase *AdministradorComunicacionCliente* Del Cliente En El Diagrama De Clase Del *FDJ*
Fuente: [DUTA01]

Atributo	Alcance	Tipo	Explicación
documento	privado	Document	Representa una instancia del documento XML en memoria, que a su vez contiene a los elementos estructurales del documento -nodos-. Utilizada al guardar e imprimir el documento XML-DOM.
indentacion	privado	int	Representa la cantidad de espacios en blanco que se dejará entre cada nivel de indentación, al imprimir o guardar el documento XML-DOM, para facilitar la comprensión de la jerarquía.
vistasDeUsuarios	privado	Element	Representa al elemento raíz de la jerarquía del documento XML-DOM. Utilizándolo como base, se puede localizar cualquier otro elemento dentro de la jerarquía, como la vista de un usuario particular, y cualquiera de sus atributos.

Tabla 5.7: Atributos De La Clase *AdministradorConsultaServidor* Del Servidor En El Diagrama De Clase Del *FDJ*
Fuente: [DUTA01]

Atributo	Alcance	Tipo	Explicación
actualizarArbol	privado	boolean	Representa una bandera, que indica cuando se debe actualizar el árbol que observa el usuario
consultaDelUsuario	privado	Element	Representa al elemento raíz de la jerarquía del documento XML-DOM. Utilizándolo como base, se puede localizar cualquier otro elemento dentro de la jerarquía, como enunciados, Tema a discutir, tipo de consulta.
dialogo	privado	Element	Representa al elemento que contiene la consulta que se realiza al dialogo del foro. Utilizándolo como base, se puede localizar cualquier enunciado dentro de la jerarquía.
documento	privado	Document	Representa una instancia del documento XML en memoria, que a su vez contiene a los elementos estructurales del documento -nodos-. Utilizada al guardar e imprimir el documento XML-DOM.
indentacion	privado	int	Representa la cantidad de espacios en blanco que se dejará entre cada nivel de indentación, al imprimir o guardar el documento XML-DOM, para facilitar la comprensión de la jerarquía.
temaADiscutir	privado	Element	Representa al elemento que contiene el tema que se discute dentro del foro.
tipoConsulta	privado	Element	Representa al elemento que contiene el tipo de consulta que realiza el usuario

Tabla 5.8: Atributos De La Clase *AdministradorConsultaCliente* Del Cliente En El Diagrama De Clase Del *FDJ*
Fuente: [DUTA01]

Atributo	Alcance	Tipo	Explicación
documento	privado	Document	Representa una instancia del documento XML en memoria, que a su vez contiene a los elementos estructurales del documento -nodos-. Utilizada al guardar e imprimir el documento XML-DOM.
indentacion	privado	Int	Representa la cantidad de espacios en blanco que se dejará entre cada nivel de indentación, al imprimir o guardar el documento XML-DOM, para facilitar la comprensión de la jerarquía.
listaDeUsuarios	privado	Element	Representa al elemento raíz de la jerarquía del documento XML-DOM. Utilizándolo como base, se puede localizar a cualquier usuario y sus atributos.

Atributo	Alcance	Tipo	Explicación
usuariosRegistrados	privado	int	Representa la cantidad de usuarios que se encuentran registrados en la lista -activos e inactivos-.

Tabla 5.9: Atributos De La Clase *AdministradorUsuario* Del Servidor En El Diagrama De Clase Del *FDJ*
Fuente: [DUTA01]

Atributo	Alcance	Tipo	Explicación
dialogo	privado	Element	Representa al elemento que contiene el dialogo que se desarrolla dentro del foro. Utilizándolo como base, se puede localizar cualquier enunciado dentro de la jerarquía.
dialogoDelForo	privado	Element	Representa al elemento raíz de la jerarquía del documento XML-DOM. Utilizándolo como base, se puede localizar cualquier otro elemento dentro de la jerarquía, como enunciados, marcas, umbrales o el tema de discusión.
documento	privado	Document	Representa una instancia del documento XML en memoria, que a su vez contiene a los elementos estructurales del documento -nodos-. Utilizada al guardar e imprimir el documento XML-DOM.
indentacion	privado	int	Representa la cantidad de espacios en blanco que se dejará entre cada nivel de indentación, al imprimir o guardar el documento XML-DOM, para facilitar la comprensión de la jerarquía.
ma	privado	double	Representa el valor que asumirá la marca de aceptación para una sesión particular del foro.
mr	privado	double	Representa el valor que asumirá la marca de rechazo para una sesión particular del foro.
temaADiscutir	privado	Element	Representa al elemento que contiene el tema que se discute dentro del foro.
ua	privado	double	Representa el valor que asumirá el umbral de aceptación para una sesión particular del foro.
ur	privado	double	Representa el valor que asumirá el umbral de rechazo para una sesión particular del foro.

Tabla 5.10: Atributos De La Clase *AdministradorDialogo* Del Servidor En El Diagrama De Clase Del *FDJ*
Fuente: [DUTA01]

ANEXO: DESCRIPCIÓN DE LOS MÉTODOS DE LOS DIAGRAMAS DE CLASES ESTÁTICAS DEL CLIENTE Y EL SERVIDOR DEL FDJ.

En este anexo se presentan las descripciones y significados de los métodos que disponen los diagramas de clases estáticas del cliente y del servidor del ***FDJ***, mencionadas en la sección 4.1.de acuerdo a **[DUTA01]**, a través de las tablas que se muestran a continuación.

Método	Alcance	Tipo	Explicación
AdministradorComunicacionServidor()	público	Constructor	Crea una instancia de la clase sin inicializar los atributos.
AdministradorComunicacionServidor(int puerto, String direccionIP)	público	Constructor	Crea una instancia de la clase inicializando sus atributos.
BajarServidor()	público	boolean	Desactiva el servidor.
cerrar_Server_sock()	público	void	Cierra el servidor de socket.
getDir(ServerSocket s)	público	String	Recupera la dirección IP del sistema donde se aloja el servidor.
getName(ServerSocket s)	público	String	Recupera el nombre del sistema donde se aloja el servidor del foro.
Levanta (int port)	público	boolean	Activa el servidor, en el puerto indicado.
mandaMensajeUno(String cad, int idUsuario)	público	void	Envía un mensaje al usuario señalado por idUsuario.

Tabla 5.11: Métodos De La Clase *AdministradorComunicacionServidor* Del Servidor En El Diagrama De Clase Del *FDJ*
Fuente: [DUTA01]

Método	Alcance	Tipo	Explicación
cerrarSocket()	público	void	Cierra el socket de un cliente.
cerrarThread()	público	void	Cierra el hilo en que se ejecuta una conexión de un cliente.
ConexionClienteServidor (int indice, Socket socket, String dir_conex, int act)	público	Constructor	Crea una conexión que escuchará a un cliente dado, a través de un socket, y una dirección de conexión.
enviar_cad(String cad_usu)	público	void	Envía un mensaje al cliente que tiene asignada la conexión.
refresh(String S)	público	void	Escucha los mensajes que envía el cliente que tiene asignada la conexión.

Tabla 5.12: Métodos De La Clase *ConexionClienteServidor* Del Servidor En El Diagrama De Clase Del *FDJ*
Fuente: [DUTA01]

Método	Alcance	Tipo	Explicación
AdministradorComunicacionCliente()	público	Constructor	Permite que se declaren instancias de la clase sin alterar sus atributos.
AdministradorComunicacionCliente(String dir_server, int port_server, String nombre)	público	Constructor	Permite declarar instancias de la clase inicializando la comunicación con el servidor.
cerrar_socket()	privado	void	Cierra la conexión socket con el servidor.
conectar(String dir_ser, int port)	privado	void	Establece la comunicación vía socket con el servidor
desconectar()	público	void	Elimina la comunicación vía socket con el servidor.
enviar(String cad)	público	boolean	Envía las cadenas al servidor, que contienen las órdenes que anota el cliente.
refresh(String S)	privado	void	Escucha al servidor, esperando recibir sus mensajes, los analiza, y envía las órdenes recibidas a las clases que lo ameriten.
repintarArbol(JScrollPane panel)	público	JScrollPane	Retorna un ScrollPane que muestra al usuario la consulta que realiza.

Tabla 5.13: Métodos De La Clase *AdministradorComunicacionCliente* Del Cliente En El Diagrama De Clase Del *FDJ*
Fuente: [DUTA01]

Método	Alcance	Tipo	Explicación
AdministradorConsultaServidor()	público	Constructor	Permite declarar instancias de la clase, sin inicializar sus atributos.
AdministradorConsultaServidor(int tabulacion)	público	Constructor	Permite declarar instancias de la clase, inicializando sus atributos.
agregarConsultaDeUsuario(int idUsuario)	público	boolean	Agrega la consulta del usuario identificado por idUsuario.
agregarEnunciadoAConsulta(int idUsuario, String llaveEnunciado)	privado	void	Permite agregar un enunciado a la consulta de un usuario.
borrarConsultaDeUsuario(int idusuario)	público	void	Elimina la consulta del usuario identificado por idusuario.
esConsultaDeUsuarioValida(int idusuario)	público	boolean	Permite determinar si el usuario idusuario posee una consulta.
esEnunciadoIncluido(int idUsuario, String llave)	público	boolean	Permite determinar si el enunciado definido por llave se encuentra incluido en la consulta del usuario identificado por idUsuario.
guardarArchivo(String nombreArchivo)	público	void	Permite guardar las consultas en un archivo XML.
modificarAtributoConsultaActualizada(int idUsuario, boolean estado)	público	void	Permite establecer que la consulta del usuario idUsuario esta actualizada o no.
modificarConsultaUsuario(int idUsuario, String tipoConsulta)	público	void	Permite modificar el tipo de consulta que observa un usuario dado.
obtenerTipoConsulta(int idUsuario)	público	String	Permite conocer el tipo de consulta que realiza un usuario dado.
obtenerTramaEnunciado(String llaveEnunciado)	público	String	Permite recuperar la trama que se enviará a un usuario, para que incluya en su consulta local, a un enunciado.

Método	Alcance	Tipo	Explicación
obtenerVectorTramasClientes(Vector vectorEnunciados, int idUsuario, String llave)	público	Vector	Permite actualizar las vistas de los usuarios dentro del servidor, y retorna un vector con los mensajes que se debe enviar a cada usuario, para que se actualize su consulta, con los datos de refresco del servidor.

Tabla 5.14: Métodos De La Clase *AdministradorConsultaServidor* Del Servidor En El Diagrama De Clase Del *FDJ*
Fuente: [DUTA01]

Método	Alcance	Tipo	Explicación
actualizarDialogoMensaje(Vector vectorMensajes)	público	Void	Transforma los mensajes que se reciben del servidor, en órdenes que alterarán la estructura de la consulta del cliente, agregando, modificando o reemplazando enunciados.
AdministradorConsultaCliente()	público	Constructor	Crea instancias de la clase, sin alterar su estado.
AdministradorConsultaCliente(String tema, int indentacion)	público	Constructor	Crea instancias de la clase, inicializando sus parámetros.
agregarEnunciado(String llavePadre, String naturalezaEnunciado, String textoEnunciado, String nombreUsuario, String fecha, String hora, String plausibilidad, String color, String apoyo, String evidenciaVacia, String naturalezaAnterior)	público	Int	Agrega enunciados a la consulta del cliente y retorna un código de error, dependiendo si se acepta o no la adición del enunciado.
asignarEstadoActualizarArbol(boolean estadoConsulta)	público	Void	Altera el estado de la bandera de actualización del árbol en el cliente.
asignarTipoConsulta(String tipoConsulta)	público	Void	Cambia el tipo de consulta que realiza el cliente
esEnunciadoRegistrado(String Llave)	público	Bolean	Comprueba si un enunciado dado está incluido en la consulta del cliente.
guardarArchivo(String nombreArchivo)	público	Void	Guarda la consulta del usuario como un archivo XML, con el nombre propuesto como parámetro.
modificarEnunciado(String llaveEnunciado, String naturalezaEnunciado, String textoEnunciado, String nombreUsuario, String fechaEnunciado, String horaEnunciado, String plausibilidadEnunciado, String colorEnunciado, String apoyoEnunciado, String evidenciaVacia, String naturalezaAnterior)	público	Int	Modifica enunciados en la consulta del cliente y retorna un código de error, dependiendo si se acepta o no la modificación del enunciado.
obtenerArbolJTree(JTree arbol)	público	Jtree	Genera un árbol JTree que representa a la consulta del cliente
obtenerCantidadDeHijos(Node enunciado)	público	int	Permite obtener la cantidad de hijos que forman la evidencia asociada de un enunciado.
obtenerEstadoActualizarArbol()	público	boolean	Permite determinar si se debe actualizar el árbol de la consulta del cliente
obtenerllavePadre(String llaveHijo)	público	String	Permite obtener la llave del padre de un enunciado, dada la llave del enunciado hijo.

Método	Alcance	Tipo	Explicación
obtenerMensajeAgregarEnunciado(String llavePadre, String textoEnunciado, String nombreUsuario, String naturalezaEnunciado)	público	String	Permite recuperar una correspondiente al mensaje de una orden para agregar un enunciado al dialogo, lista para ser enviada al servidor del foro
obtenerMensajeModificarEnunciado(String llaveEnunciado, String textoEnunciado, String nombreUsuario, String naturalezaEnunciado, boolean retractar, boolean reactivar)	público	String	Permite recuperar una correspondiente al mensaje de una orden para modificar un enunciado del dialogo, lista para ser enviada al servidor del foro
obtenerNodoBaseEnunciado(String llaveEnunciado)	público	Node	Permite recuperar el nodo base, dentro de la estructura XML-DOM, de un enunciado, dada la llave que lo identifica.
obtenerTipoConsulta()	público	String	Permite recuperar el tipo de consulta que efectúa el cliente.
obtenerEvidenciaAsociada(Element enunciado)	público	Element	Permite recuperar la evidencia asociada de un enunciado, como un elemento, que contiene a todos los enunciados hijos, recursivamente.

Tabla 5.15: Métodos De La Clase *AdministradorConsultaCliente* Del Cliente En El Diagrama De Clase Del *FDJ*
Fuente: [DUTA01]

Método	Alcance	Tipo	Explicación
AdministradorUsuario(int tabulacion)	público	Constructor	Permite declarar instancias de la clase, inicializando sus atributos.
AdministradorUsuario()	público	Constructor	Permite declarar instancias de la clase, sin alterar sus atributos.
agregarUsuario(String nombrePropuesto, String hostPropuesto)	público	boolean	Permite agregar usuarios a la lista, retorna true si el usuario se agregó exitosamente, o false, en caso que no se pueda agregar.
borrarUsuario(String nombre)	público	boolean	Permite desactivar usuarios de la lista, retorna true en caso que se pueda desactivar exitosamente al usuario, o false en caso que no se pueda desactivar
esUsuarioActivo(String nombre)	público	boolean	Permite determinar si existe algún usuario activo en la sesión, registrado bajo el nombre propuesto
esUsuarioRegistrado(String nombre)	público	boolean	Permite determinar si existe algún usuario registrado en la sesión, bajo el nombre propuesto, la búsqueda incluye a los usuarios desactivados.
guardarArchivo(String nombreArchivo)	público	void	Permite guardar la lista de usuarios en un archivo con formato XML.
obtenerCantidadDeUsuariosActivos()	público	int	El nombre del método se explica por si solo.
obtenerCantidadDeUsuariosRegistrados()	público	int	El nombre del método se explica por si solo.
obtenerIdUsuario(String nombreBuscado)	público	int	Permite recuperar el identificador de usuario, dado el nombre de un usuario, en caso que no exista el usuario dentro de la sesión, retorna 0.

Método	Alcance	Tipo	Explicación
obtenerListaUsuariosSwing()	público	JList	Permite recuperar un lista JList que contiene a los usuarios registrados en la sesión, activos y desactivados.

Tabla 5.16: Métodos De La Clase *AdministradorUsuario* Del Servidor En El Diagrama De Clase Del *FDJ*
Fuente: [DUTA01]

Método	Alcance	Tipo	Explicación
actualizarDialogoTrama(String trama)	público	String	Permite transformar los mensajes que se reciben de los clientes, en órdenes que alterarán la estructura del díalogo del servidor, agregando o modificando enunciados.
AdministradorDialogo(String tema, double ma, double mr, double ua, double ur, int tabulacion)	público	Constructor	Permite declarar instancias de la clase inicializando sus atributos internos.
AdministradorDialogo()	público	Constructor	Permite declarar instancias de la clase sin inicializar sus atributos internos.
agregarEnunciado(String llavePadre, String textoEnunciado, String nombreUsuario, String naturalezaEnunciado)	público	int	Permite agregar enunciados al diálogo del servidor. El enunciado se agrega basándose en la llave del enunciado padre.
calcularPlausibilidadHilo(String llave)	público	void	Permite calcular, en cascada, el estado de plausibilidad de un hilo de discusión.
esEnunciadoRegistrado(String llaveEnunciado)	público	boolean	Determina si un enunciado, definido por su llave, se encuentra registrado o no en el diálogo.
guardarArchivo(String nombreArchivo)	público	void	Permite guardar el diálogo realizado en la sesión, en un archivo XML.
modificarEnunciado(String llaveEnunciado, String textoEnunciado, String nombreUsuario, String naturalezaEnunciado, boolean retractar, boolean reactivar)	público	int	Permite modificar un enunciado del diálogo, con base en su llave.
obtenerCantidadDeHijos(Node enunciado)	público	int	Permite obtener la cantidad de enunciados que forman la evidencia asociada de otro enunciado.
obtenerEnunciadosAlterados(String llaveEnunciado)	público	Vector	Permite obtener un vector que contiene los enunciados que resultan alterados al realizarse la adición o modificación de un enunciado definido por llaveEnunciado.
obtenerEvaluarAgregarEnunciado(int codigo)	público	String	Permite evaluar los códigos numéricos que se recuperan al agregar enunciados y recuperar el mensaje correspondiente.
obtenerEvaluarModificarEnunciado(int codigo)	público	String	Permite evaluar los códigos numéricos que se recuperan al modificar enunciados y recuperar el mensaje correspondiente.
obtenerEvidenciaAsociada(Element enunciado)	público	Element	Permite recuperar la evidencia asociada de un enunciado, como un elemento del documento XML, que incluye a todos los enunciados hijos, en cascada.

Método	Alcance	Tipo	Explicación
obtenerLlaveEvidenciaAgregada(String llavePadre)	público	String	Permite recuperar el valor que se asignará a la llave de un enunciado que se agrega al foro, como evidencia asociada de otro enunciado.
obtenerNodoBaseEnunciado(String llaveEnunciado)	público	String	Permite recuperar el nodo base de la estructura XML, correspondiente a la llave de un enunciado previamente registrado.

Tabla 5.17: Métodos De La Clase *AdministradorDialogo* Del Servidor En El Diagrama De Clase Del *FDJ*
Fuente: [DUTA01]