UNIVERSIDAD
REY JUAN CARLOS

INGENIERÍA INFORMÁTICA

Curso Académico 2007/2008

Proyecto Fin de Carrera

BLAS:
BASE LAYER FOR APPLICATION SERVICES

Autor: Eduardo Orive Vinuesa
Tutor: Gregorio Robles Martínez
Co-tutor: Agustín Santos Méndez

A mi familia, por su apoyo.

Agradecimientos

Quiero agradecer en primer lugar a Agustín la confianza que ha depositado en mi y en mi proyecto, así como a Gregorio que ha accedido a ser mi tutor. Los dos han dedicado cantidades ingentes de paciencia para que este proyecto tuviera el objetivo que había planteado y que verdaderamente me ilusionaba.

También mis compañeros han demostrado gran interés en el proyecto y no han dejado de proporcionarme ánimo y consejos, y en especial a Alberto, Jesus e Iván.

A mis padres y a mi hermano que siempre han mostrado su disposición para ayudarme en lo que pudieran, incluyendo la tortuosa búsqueda de erratas en el texto.

Resumen

BLAS es una plataforma pensada para orientar los esfuerzos dedicados a programar un servicio de Internet. Originalmente pensado para protocolos de tipo de texto sobre TCP/IP, se ha extendido su funcionalidad a UDP/IP y en un futuro se proveerá un soporte más fiable para protocolos binarios.

Con el objetivo de mejorar el proceso de diseño e implementación esta plataforma propone organizar el proceso de atención de las peticiones en una serie discreta de estados, lo más simples posible, estos estados se materializan en métodos con identificadores unívocos que van dirigiendo la ejecución de uno a otro. Este modo de diseño estructura y facilita de forma considerable el trabajo sobre protocolos con un formato determinado.

El desarrollo de servicios con este sistema provee de una estructura y permite acelerar el proceso de obtención de un prototipo estable.

La elección de Python como lenguaje ha permitido una gran flexibilidad en la programación y sobretodo claridad en el código resultante. Precisamente mantener la claridad en la implementación es una de las principales premisas a la hora de orientar la programación de los servicios.

Se ha optado por la licencia GPL para proteger el proyecto y garantizar su disponibilidad futura, en especial a la comunidad de estudiantes y facilitar la participación a aquellos interesados en la platafoma y en el software libre.

Índice general

Índice de figuras

Capítulo 1

Introducción

Actualmente casi la totalidad de las instalaciones informáticas son redes, y una mayoria de estas están conectadas a Internet e incluso la propia Internet forma parte de estas.

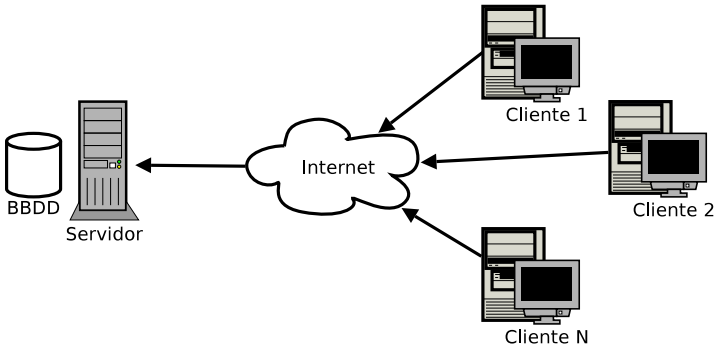

Figura 1.1: Ejemplo de un Sistema Distribuido

1.1. Historia de los servicios informáticos

Después del desarrollo de la primera generación de computadoras el número de estas empezo a aumentar, aunque su disponibilidad estaba normalmente limitada a una unidad por cada entidad (instituciones educativas e instalaciones militares). Estas máquinas requerían de un entorno ambiental muy concreto y el acceso a las instalaciones donde se albergaban estaba severamente restringido. Por estas mismas limitaciones y para maximizar la disponibilidad se ideó un sistema limitado de terminales remotas,

los teletipos, cuya función era comunicar con el ordenador central pudiendo utilizar enlaces dedicados o conmutados (líneas telefónicas), permitir la inserción de programas y recuperar el resultado una vez el sistema hubiera finalizado. Estos sistemas de comunicación y control remotos merecen la consideración de servicios informáticos.

Con el tiempo la versatilidad de los sistemas fue ampliandose, los ordenadores dejaron de emplearse como meras ejecutoras de algoritmos y empezaron a ofrecer distintos tipos de servicios:

1. Boletines de noticias
2. Transferencias de ficheros
3. Correo electronico

Es precisamente en esta época cuando empiezan implantarse sistemas de comunicación por conmutación convencionales y los servicios empiezan a ser utilizador desde estos nuevos medios de enlace. La oferta de servicios deja de estar limitada a las grandes entidades y empiezan a surgir “BBSes” (Bulletin Board System) albergadas en ordenadores privados. Durante más de 25 años han sido un medio de comunicación y entretenimiento, en opinión de muchos, siguen siendo los mejores ambientes comunitarios.

La capacidad monousuario de los sistemas de la época presentó la necesidad de empezar a dejar de utilizar enlaces de líneas conmutadas para añadir un nuevo nivel de paquetes conmutados para eliminar esta limitación.

Con el tiempo y con el inexorable cumplimiento de la ley de Moore, los sistemas fueron ampliando geométricamente su capacidad, también lo hacía su disponibilidad en el mercado haciéndose cada vez asequibles para el público en general, acompañado por una reducción drástica del volumen y consumo energético.

La popularización de Internet llevó a que practicamente cualquier ordenador doméstico estuviera dotado de algún dispositivo de comunicación metropolitana y una pila IP, permitiendo así acceder a los servicios ofrecidos en Internet a través del respetivo ISP.

En poco tiempo dispositivos portátiles imitaban la tendencia junto a otros microdispositivos. Es muy común ver que aparatos convencionales, desde electrodomesticos a vehículos lleven incorporados sistemas de comunicación que automáticamente son capaces de desempeñar tareas como emitir a la central la posición del vehículo y todo tipo de medidas cuantificables sobre su estado.

1.2. Motivación

En los último años cursando Ingeniería Informática y en proyectos profesionales se ha presentado en númerosas ocasiones la necesidad de escribir servicios orientados a Internet. La necesidad de encontrar un sistema estructurado y ágil para la elaboración de prototipos es lo que ha inspirado la creación de esta plataforma.

Es en este entorno donde se desarrollan las arquitecturas de la mayoría de los sistemas. Distribuidas, y en una gran proporción siguiendo el modelo Cliente-Servidor, casi todas las nuevas aplicaciones o sistemas basan gran parte de su funcionalidad en la conectividad mediante redes.

Sin embargo realizar aunque sea un prototipo operativo de un servicio de red requiere de una importante inversión en tiempo y no se obtienen resultados tangibles hasta que se ha implementado una buena parte del código.

También es muy común ver que muchos prototipos sean descartados (total o parcialmente) y deban re-implementarse ya que, aunque su rendimiento sea aceptable, la seguridad que ofrecen no lo es, con lo que el proceso de crear el servicio duplica su coste cuando el sistema llega a la etapa de producción.

No obstante la escritura de un servicio de Red puede reducirse considerablemente si se implementan tan solo las características particulares de su protocolo. Basicamente cualquier protocolo puede entenderse como un autómata más o menos complejo, siguiendo este modo de diseño puede simplificarse de manera considerable el trabajo tanto del programador como del analista.

La intención principal de esta plataforma es evitar la reimplementación innecesaria de funcionalidades que son comunes a todo tipo de servicio de red interactivo, además de proponer una metodología concreta en la implementación.

1.3. Disponibilidad del proyecto

Este proyecto se ha traspasado desde una de sus más primitivas versiónes al repositorio que Google dispone para desarrolladores. La elección de `http://code.google.com/` como repositorio se ha debido sobre todo al elenco de herramientas y servicios que Google está dedicando a los desarrolladores de software y sobre todo a los proyectos de programación colaborativos.

Todo el código y la documentación referente a este proyecto podrá encontrarse

en http://code.google.com/p/blas/. La participación de terceros quedará abierta en cuanto se evalue el proyecto, en mi opinión ese será el verdadero comienzo del desarrollo de la plataforma, cuando se pruebe su uso y las aportaciones de otros desarrolladores amplien sus funcionalidades.

La documentación puede encontrarse también en formato LaTex, un procesador de textos muy adecuado para documentación cientifica y técnica http://www.latex-project.org/ Además los diagramas que ilustran este documentos se encuentran disponibles en su formato nativo, editable por Dia http://www.gnome.org/projects/dia/.

1.4. Licencias

Este proyecto dispone de dos licencias: una para el propio software que garantice a futuros la disponibilidad de este para toda la comunidad de software libre, otra para este documento, de tal forma que su contenido pueda aprovecharse en otros esfuerzos de documentación colaborativa como http://es.wikipedia.org.

Ambas licencias pueden encontrarse en sus respectivos apéndices.

Capítulo 2

Objetivos

Esta plataforma se ha ideado para facilitar el diseño y la implementación a los colectivos que por diversos motivos se ven ante la necesidad de programar un servicio de red en poco tiempo.

Existen multitud de funcionalidades ligadas a los programas que ofrecen servicios sobre TCP y UDP que inevitablemente consumen mucho tiempo para su implementación y que normalmente los programadores se encuentran realizando funciones de implementación muy similares de un proyecto a otro pero sin permitir una reutilización eficiente del código.

Básicamente estas tareas se reducen a:

1. Entrada controlada de datos.
2. Registro (o registros) de actividad y error.
3. Lectura de los archivos de configuración.
4. Parseado de los parámetros de arraque.

El objetivo principal de esta librería es facilitar las tareas de diseño e implementación de este tipo de software, acelerando la aparición de prototipos que permitan iniciar lo antes posible la depuración tanto del servicio como de los clientes.

Es posible que existan dudas sobre si hoy día la comunidad de desarrolladores tiene la necesidad de la realizacion de este tipo de software. En la actualidad se pueden encontrar cientos de implementaciónes de protocolos sobre multitud de tecnologías diferentes, pero todavía siguen apareciendo proyectos que requieren de la implementación de servicios de protocolos antiguos o nuevos.

2.1. Docencia

Existen números proyectos académicos que requieren la implementación de:

1. Protocolos nuevos.
2. Protocolos ya existentes.
3. Modificaciones sobre protocolos antiguos.

y que pueden ver facilitada su realización utilizando esta plataforma.

2.2. Microdispositivos

La mayoría de los dispositivos dotados de un sistema operativo mínimo disponen de una modesta pila IP ya que, prácticamente todos estos dispositivos, se diseñan incluyendo algún tipo de hardware de interconexión mediante enlaces de radio o cableado. Sin embargo por razones de coste económico y rendimiento energético se ven afectados por una leve capacidad de memoria y/o potencia de cálculo que hacen que la opción de comunicarlos mediante un servicio convencional (HTTP y REST por ejemplo) resulte muy engorrosa (y caro) en su programación, e incluso a veces impracticable.

Es por este motivo que se suelen diseñar protocolos normalmente muy escuetos, especificamente adaptados a la funcionalidad del dispositivo y a sus limitaciones de proceso.

2.3. Seguridad Informática

En un entorno en que los ataques mediante gusanos son casi rutinarios, se emplea mucho tiempo y esfuerzo en crear servicios que se comporten como los originales que emplee el gusano para penetrar en el sistema y así poder estudiar qué acciones realiza una vez dentro.

La capacidad de programar servicios simulados ágilmente puede ser un complemento perfecto a otras herramientas de estudio de la seguridad como son las 'HoneyNets', dotando a las máquinas virtuales de un comportamiento mucho más flexible.

Capítulo 3

Estado del arte

3.1. Lenguaje: Python

3.1.1. Elección del Lenguaje

Se ha elegido Python como lenguaje de programación para esta plataforma por las siguientes razones:

Velocidad de aprendizaje cualquier persona que conozca un lenguaje orientado a objetos puede aprenderlo en muy poco tiempo.

Reflexividad e introspección Las clases en Python tienen conocimiento de sus propios métodos y atributos.

Interpretado Al ejecutarse bajo un interprete no harán falta compilar distintos ejecutables según la plataforma.

Con el empleo de Python se puede conseguir con pocas líneas la implementación de un protocolo completo. Al tratarse de un lenguaje en que la identación es obligatoria y la sintaxis muy clara, normalmente de produce un código fuente bien estructurado y que resulta fácil de leer incluso para quienes apenas conozcan Python.

3.1.2. Python: historia y características

Python fue creado como sucesor del lenguaje ABC por Guido van Rossum en 1990 cuando trabajaba en el Stichting Mathematisch Centrum (CWI). En 1995 Guido continuó su trabajo en Python en el CNRI donde creó muchas versiónes del lenguaje. En

mayo del 2000 Guido y el equipo de desarrollo de Python se trasladan a BeOpen.com y se forma el BeOpen PythonLabs. En octubre de este mismo año, PythonLabs se va a Digital Creations (actualmente Zope Corporation). En 2001, se crea la Python Software Foundation (PSF), una organización sin ánimo de lucro creada específicamente para proteger la libertad de Python como lenguaje de código abierto.

El nombre del lenguaje proviene de la afición de su creador original, Guido van Rossum, por los geniales humoristas británicos Monty Python.

Python ha sido usado para crear programas tan famosos como el gestor de listas de correo Mailman o los gestores de contenido Zope y Plone.

Python se puede encontrarse en varias encarnaciones. En el sitio web de la Python Software Foundation, aunque el interprete original de Python está implementado en C. existen otras:

1. Basada en Java existe una implementación denominada Jython puede utilizarse para trabajar con código Java nativamente.

2. Iron Python, una versión en C# existe para hacer uso de las plataformas Mono y .Net y que los programadores de estas puedan beneficiarse de su potencia y flexibilidad.

En cada una de estas encarnaciones Python se ha escrito en un lenguaje y funciona nativamente en este aunque puede también interactuar perfectamente con otros lenguajes a través de sus correspondientes módulos.

Con propósitos de investigación y desarrollo existe también una implementación de Python sobre Python. El proyecto PyPy fue iniciado en 2003 con la intención de permitir a los programadores en Python modificar el comportamiento del interprete. Además se trata de un proyecto de código abierto, desarrollado por una comunidad para su libre distribución y modificación. PyPy está patrocinado por la Unión Europea como un STReP (Specified Targeted Research Project), parte del plan de apoyo FP6.

3.1.3. Evolución

Primera publicación

En 1991 van Rossun publicó el código (con la versión 0.9.0) en alt.sources. Ya en este primer estado de desarrollo había clases con herencias, manejo de expeciones, funciones y los tipos de datos básicos de listas, diccionarios, cadenas y demás. También en

esta publicación inicial había un módulo de sistema que fue tomado prestado de modula3; van Rossum define este modulo como “una de las principales unidades de programación de Python”. El modelo de gestión de excepciones de Python también imita al de Modula3 salvo que incluye la cláusula “else”. En 1994 arranca comp.lang.Python, la primera lista de discusión de Python, marcando un hito en el crecimiento de la base de usuarios de Python.

Versión 1.0

Python llegó a la versión 1.0 en Enero de 1994. Las principales nuevas funcionalidades incluídas en esta publicación fueron las utilidades de programación funcional lambda, map, filter y reduce. Van Rossum expresó que “Python ha conseguido lambda, reduce, filter y map por cortesía (en mi opinión) de algún hacker de Lisp que las echabada de menos y envió parches operativos”.

La última versión publicada cuando van Rossun estaba en el CWI fue Python 1.2. En 1995 van Rossum continuó su trabajo en la Corporación para Iniciativas de Desarrollo Nacional (CNRI) en Reston (Virginia) desde la que publicó varias versiónes.

Por la versión 1.4, Python había conseguido varias nuevas funciones. Es de destacar que entre estas están los parámetros por palabra clave de Modula3 (que son también similares a los parámetros por palabra clave de Lisp) y el soporte integrado para números complejos. También incluye una forma básica de ocultamiento de datos por manipulación de nombre, aunque podía ser facilmente obviado.

Durante la estancia de var Rossum en el CNRI, lanzó la iniciativa Programación de Ordenadores para Todos (CP4E), que pretendía hacer la programación más accesible a mas gente, con un conocimiento básico en lenguajes de programación, de forma similar a los conocimientos básicos de literatura inglesa y matemáticas que se exigen a cualquier empleado. Python sirvió con un papel central en esto: debido a su enfoque en una sintaxis clara, ya podía considerarse un candidato adecuado para los objetivos del CP4E, estos ya apuntaban a su predecesor ABC. El proyecto fue fundado por DARPA. En 2007 este proyecto parece inactivo y aunque Python pretende ser de fácil aprendizaje y no demasiado arcano en su sintaxis y semántica, el llegar a ser comprensible para los no programadores no es una intencion activa.

Versión 2.0

Python 2.0 introdujo la comprensión de listas, una función prestada de los lenguajes de programación funcionales Lips y Haskell. La sintaxis de Python en su construccion es muy parecida a la de Haskell, dejando a un lado la preferencia de Haskell por los caracteres de puntuación y la preferencia de Python por las palabras clave alfabéticas. Python 2.0 también introdujo un sistema de recolección de basura capaz de tratar referencias cíclicas.

Python 2.1 era bastante parecido a Python 1.6.1, al igual que Python 2.0. La licencia cambió su nombre a Python Software Foundation License. Todo el código, documentación y especificaciónes añadidas, desde la versión alpha de Python 2.1, es propiedad de la Python Software Foundation, una organización sin ánimo de lucro formada en 2001, modelada siguiendo el ejemplo de la Apache Software Foundation. La publicación incluyó un cambio en la especificación del lenguaje para soportar intervalos (scopes) anidados, al igual que otros lenguajes con intervalos estáticos. Esta función estaría desactivada por defecto y no fue necesaria hasta Python 2.2.

Una de las principales innovaciones en Python 2.2 fue la unificación de los tipos de Python (tipos escritos en C) y clases (tipos escritos en Python) en una única jerarquía. Esta unificación hizo al modelo de objetos de Python un modelo orientado pura y consistentemente a objetos. También se añadieron generadores inspiradores por el lenguaje Icon.

Versión 2.6

En el momento de desarrollar esta librería, Python 2.5 y Python 3.0 conviven en muchos proyectos; sin embargo y aunque se pueda ejecutar sobre 3.0, el poco tiempo que lleva la versión 3.0 en funcionamiento hace poco recomendable su utilización en sistemas que requieran de una cierta estabilidad o sean candidatos a producción.

Disponibilidad

En muchos sistemas operativos Python es un componente estándar, se incluye en la mayoría de las distribuciones de Linux, NetBSD, OpenBSD y con Mac OS X. Slackware, Red Hat Linux y Fedora utilizan el instalador Anaconda, escrito en Python. Gentoo Linux utiliza Python en su sistema de administración de paquetes Portage, y su herramienta por defecto Emerge. Pardus lo utiliza para administración y durante el

arranque del sistema.

Editores

Python puede programarse con cualquier tipo de editor y siempre permitirá una lectura clara y una estructura adecuada del código. Sin embargo, son muchos los editores y entornos integrados de trabajo que incluyen funciones de ayuda en la programación:

1. Resaltado del código: Cualquier editor con unas funcionalidades de programación mínima incluirá resaltado de sintaxis.
2. Predicción de escritura: Prácticamente cualquier editor incluirá indentación automática y probablemente predicción de palabras clave.
3. Gestión de proyectos: Disponible sobre todo en IDEs y raramente en editores.
4. Shell integrada: Disponible en algúnos de los editores y la mayoría de los IDEs.

Para conseguir una lista actualizada de editores compatibles con Python en todas las plataformas puede conseguirse en esta URL: http://wiki.Python.org/moin/PythonEditors

3.2. Protocolos de red: IP

Es difícil generalizar sobre protocolos ya que estos varían ampliamente en propósito y en complejidad. La mayoróa de los protocolos de red poseen algúna de las siguiente propiedades:

1. Detección de la capa subyacente de enlace de red, o la existencia de otro punto de entrada o nodo.
2. Saludo y reconocimiento (Handshaking)
3. Negociación de varias de las características de la conexión.
4. Cómo indicar el comienzo y el final de un mensaje.
5. Formato de los mensajes.
6. Cómo actuar ante mensajes corruptos o con un formato incorrecto (corrección de errores)

7. Mecanismos de detección de pérdidas inesperada de la conexión y cómo comportarse ante estas.

8. Finalizar la sesión o conexión.

En concreto el protocolo de Internet (IP) se utiliza en los extremos de la comunicación para mantener un flujo de datos a traves de una red de paquetes conmutados.

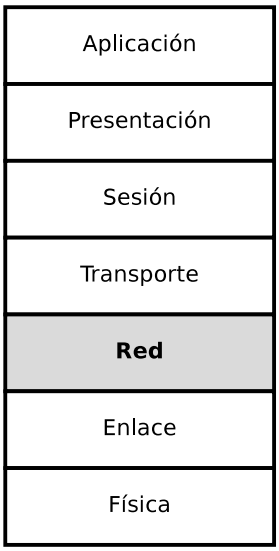

Figura 3.1: Niveles OSI: protocolo de red

Precísamente debido a la naturaleza de estas redes los datos son divididos en cantidades coherentes a las limitaciones del medio (por ejemplo 1500 bytes para redes Ethernet), estos fragmentos de información se les denominan paquetes o datagramas. Además el protocolo no requiere que las rutas que deben seguir estos paquetes sean establecidas de antemano, es el propio protocolo el que encontrará la ruta idónea.

Con el protocolo IP se provee un sistema para la transmisión de datagramas no fiable (best effort), este método de transmisión no garantiza resultados pero intentará utilizar la ruta más efectiva. Debe aclararse que IP no posee mecanismos para garantizar que la transmisión ha llegado a destino, aunque sí dispone de mecanismos para la detección de errores en los datos que contenga el paquete. Debido a la naturaleza dinámica de las rutas y a la hetereogeneidad de los dispositivos, es probable que los datagramas en que ha sido dividido el flujo de datos lleguen fuera de orden e incluso que llegue repetido; es tarea de los protocolos de transporte detectar estos sucesos.

A lo largo de la ruta es posible que la longitud maxima de transmisión (MTU) vaya cambiando (por ejemplo de 1500 bytes de un enlace Ethernet a los 576 de una conexión punto a punto). Es tarea del protocolo IP volver a fragmentar estos datagramas en otros más pequeños y recomponerlos cuando sea necesario, debe tenerse en cuenta que estos fragmentos pueden utilizar vías diferentes en según qué tramos de la ruta.

Para manejar estos incidentes el protocolo acompaña a las tramas de datos en los datagramas de unas cabeceras, estas llevan incluídas las direcciónes IP del origen y destino de la comunicación. Estas son requeridas por los dispositivos que retransmitirán los datos a través de las rutas para seleccionar el camino adecuado.

Actualmente el protocolo más utilizado en Internet es el IP versión 4 (IPv4 en adelante). Sin embargo ya se ha propuesto un sucesor, IPv6, llamado a eliminar las limitaciones que afectan a IPv4 desde su diseño. Una de estas nuevas características es el empleo de direcciónes de 128bits, que permitirá otorgar una dirección unívoca a más de 670 mil millones de dispositivos.

Cabe señalar las versiónes 0 a la 3 del protocolo IP no llegaron a emplearse y que la versión 5 se utilizó para un prototipo experimental. Además IPv4 dispone de diferentes implementaciónes que buscan mejorar su eficiencia o al menos limitar el efecto de sus deficiencias.

3.3. Protocolos de transporte

Basándonse en el nivel de Red (lo que proporciona una ruta de comunicación y un medio libre de errores inesperado en los datos recibidos) existe un nuevo nivel OSI. Los protocolos de este nivel tienen la responsabilidad de controlar la entrega en destino de los datos y en la medida de lo posible conseguirlo con una cantidad mínima de redundancia de datos.

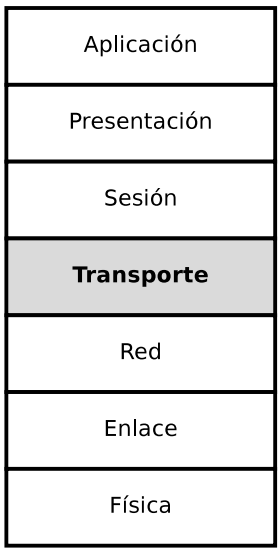

Figura 3.2: Niveles OSI: protocolos de transporte

Basándonos en esta jerarquía y situándonos sobre el protocolo de transporte los extremos de comunicación pueden dar las siguientes características por garantizadas :

1. Garantizar una transferencia libre de datos entre los interlocutores
2. Mantener una ruta de comunicacion entre puntos que no estén directamente conexiónados
3. Control de flujo y gestión del buffer
4. Multiplexión de las vías de comunicación
5. Recuperación y redirección

A continuación se van a estudiar los protocolos de transporte TCP y UDP, que aunque no son los únicos

protocolos sobre IP sí son los más utilizados, prácticamente cualquier servicio ofrecido a través de Internet funcionará sobre uno de estos dos protocolos. Se va a profundizar en su funcionamiento para que el lector se de cuenta de las diferencias entre ambos y pueda percatarse de que entre los dos cubren casi cualquier necesidad de comunicación.

3.3.1. Protocolo TCP

El Transfer Control Protocol es el encargado de proporcionar un medio de comunicación orientado a conexión. Permite controlar que los mensajes llegan de un extremo a otro en el orden correspondiente. Además detecta y según el caso corrige fallos en la conexión, ya sean errores en los mensajes recibidos, pérdidas de tramas enteras o errores en parte de estas.

Las aplicaciónes sobre TCP envían y reciben flujos de datos de forma transparente (salvo error) a la aplicación que las utiliza. Es el propio protocolo el que fragmenta la tramisión según los requisitos del medio y organiza que la recomposición de los datos sea coherente sea cual sea el orden de llegada.

Las cabeceras del protocolo TCP son bastante complejas.

Los programas que utilizan una conexión TCP para transmitir datos lo hacen de manera transparente, envían el flujo de bytes y es el protocolo el que la divide en paquetes según la configuración de la capa de red, adjunta además su propia cabecera, que será indispensable para conseguir una recepción coherente de los datos, ya que incluye entre otros campos el número de secuencia. Durante la operación una pila TCP va recibiendo los paquetes desde la capa IP, los ordena y organiza segrún el número de secuencia. Para detectar la falta de paquetes y repetir la transferencia existe un mecanismo de asentimiento de la trasmisión: en la cabecera, un campo NACK indica al emisor de la información hasta que número de secuencia se han recibido todos los paquetes y se han recompuesto los datos. El mecasnismo de caducidad (timeout) se activará si en un cierto tiempo el emisor no ha recibido confirmación por parte del receptor.

A continuación se desglosan los campos contenidos en las cabeceras:

1. Puertos de origen y destino: identifica a qué puerto de los extremos debe dirigirse el paquete.

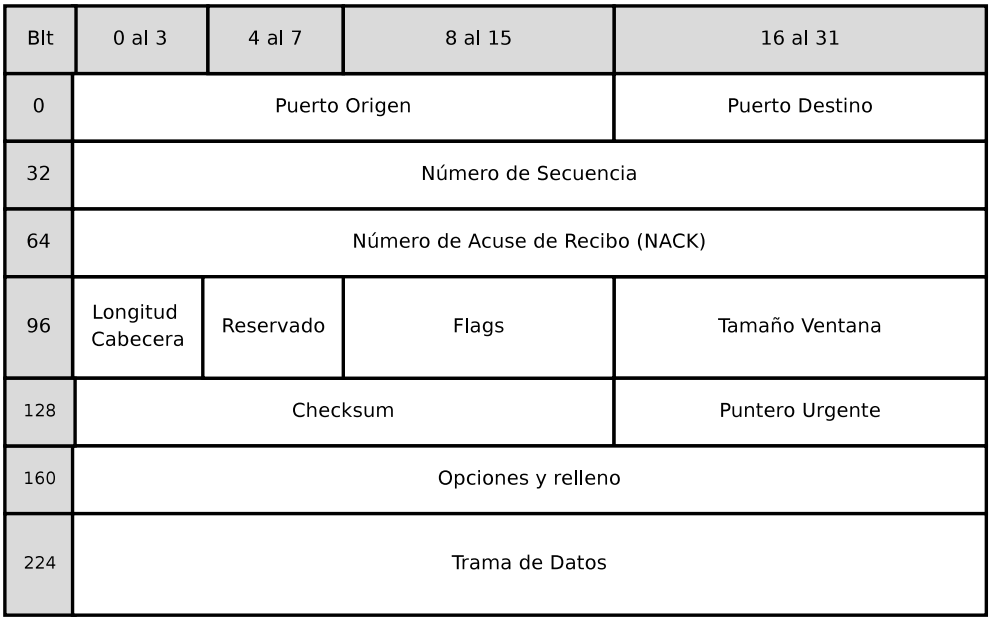

Figura 3.3: TCP: Diagrama de cabecera

2. Número de secuencia: sirve para garantizar la coherencia y completud del flujo de datos que se está envíando. Dependiendo de si el flag SYN está o no activado, puede indicarnos si ese número de secuencia será el inicial para esa transferencia de datos o si está desactivado marcará el número de secuencia del primer byte de datos.

3. NACK: si el paquete es de tipo ACK entonces el receptor está esperando que el siguiente paquete recibido sea el correspondiente al siguiente número de secuencia.

4. Longitud de cabecera: especifica el tamaño de la cabecera en divisores de 32bits. A este campo también se le denomina data-offset por ser el desplazamiento necesario para acceder a la trama de datos.

5. Reservado: estos bits deberán dejarse a 0, están reservados para aplicaciones futuras del protocolo.

6. Flags: estos 8 bits de control se utilizan para regular el funcionamiento del protocolo:

a) CWR: Congestion Window Reduced, avisa al receptor de que se está produciendo congestión en la red y debe negociarse una reducción de la ventana del buffer.

b) ECN-Echo: Se avisa al destino que puede emitir notificaciones ECN, este flag se usa durante la etapa de saludo.

c) URG: Urgente, avisa de que el paquete debe ser tratado con prioridad en la pila.

d) PSH: Push, hace que los datos almacenados en el buffer sean traspasados a la aplicación.

e) ACK: Acknowlegde, hace que el número de acuse de recibo sea considerado por el receptor.

f) RST: Reset, reinicializa la conexión

g) SYN: Synchronize, obliga a que los números de secuencia estén sincronizados.

h) FIN: Comunica el final del flujo de datos.

7. Ventana: es la capacidad de la ventana de recepción, indica en bytes la cantidad de datos que el receptor está esperando.

8. Checksun: suma de comprobación que abarca tanto cabecera como datos.

9. Urgente: Si está el flag URG activado indica el offset respecto del número de secuencia del último byte que debe considerarse urgente.

10. Opciones: Campo de información ajena al contenido de los datos, debe ser múltiplo de 32bits

11. Datos: trama con el segmento de datos que se espera en el destino.

Funcionamiento

El protocolo TCP basa su funcionamiento en una serie de procedimientos consecutivos:

1. Saludo o establecimiento de la conexión: Consistente en la negociación de 3 pasos en la que también se establecen los parámetros clave de esta.

2. Transferencia de los datos: en la que se van emitiendo asentimientos para confirmar la llegada de grupos de tramas

3. Desconexión: para la que se procede a una negociación de 4 pasos

Habitualmente para iniciar una conexión TCP un dispositivo cliente se comunica con un dispositivo servidor, que se encuentra escuchando en un puerto TCP concreto a través de la pila IP. Antes de que tanto el cliente como el servidor dispongan de un socke se realiza una negociación de 3 pasos. El cliente emite un paquete SYN al puerto específico del servidor, ahí el sistema operativo enviará la información al software que haya realizado el bind() a ese puerto. Si no existe ningún software escuchando se emitirá un paquete RST al cliente para que conozca el fallo de la operación. Si se ha alcanzado un puerto abierto el sistema responderá con un paquete SYN+ACK y para dar por establecida la conexión el cliente responderá a su vez con un paquete ACK. Es en este proceso cuando se elige al azar un número de secuencia que permitirá proceder correctamente en las operaciones de transferencia de datos.

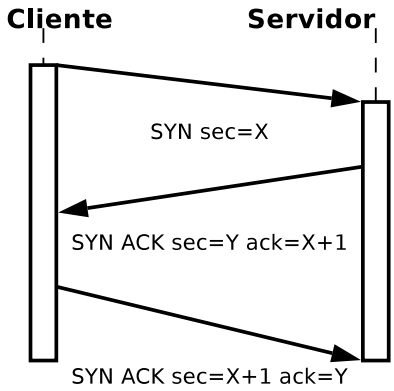

Figura 3.4: Protocolo TCP: secuencia de saludo

Una vez establecida la conexión cliente y servidor dispondrán de un mecanismo de intercambio de datos muy similar al que se dispone cuando se abre un dispositivo de tipo carácter, el nivel de aplicación emitirá y recibirá datos de forma transparente y serán las capas de Transporte y Control las encargadas de que lleguen y se ordenen correctamente en destino.

El proceso de envío de datos requiere que se hayan determinado dos parámetros clave obligatorios en el establecimiento:

1. Número de secuencia: que corresponderá al primer byte de la trama contenida en el paquete y que se espera en el otro extremo. Cabe destacar que el número de secuencia volverá a 0 cuando haya sobrepasado el máximo

2. Tamaño de ventana: es la cantidad de datos que pueden enviarse desde el emisor sin haber recibido confirmación de receptor.

Durante la transferencia el emisor envía un número de paquetes. Este número depende del número de asentimiento (que es el número de secuencia reconocido por el receptor), el número de secuencia de los paquetes que está envíando y la capacidad de la ventana de transferencia.

Normalmente el cliente irá recibiendo con un cierto orden los paquetes y modificando su propio número de secuencia. Periódicamente se informará al emisor con un paquete ACK del número de secuencia confirmado. Mientras tanto, este reenviará el conjunto de paquetes que correspondan a la ventana actual. También se emitirán ACKs cuando el paquete que se ha recibido complete a un conjunto.

En ciertas implementaciónes de TCP puede encontrarse una nueva función, el asentimiento selectivo, que permite avisar al emisor de qué paquetes son necesarios para completar el conjunto actual indicando cuales se han recibido.

Uno de los motivos más comunes de la pérdida de paquetes es la saturación del canal. Se emite más información de la que este puede enviar, con lo que los paquetes empiezan a ser descartados. Por este motivo TCP ha implementado sistemas para la detección de la saturación del canal, que regulan de una forma efectiva el tamaño de la ventana y el ritmo de emisión de los paquetes.

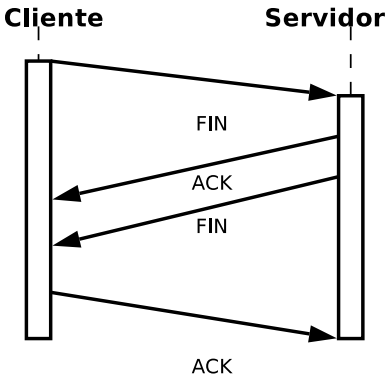

Figura 3.5: Protocolo TCP: secuencia de desconexión

Durante la finalización de una conexión uno de los extremos emite un paquete FIN, la respuesta consiste en dos paquetes, un ACK y posteriormente un FIN, cuando el emisor de la despedida emita un ACK la conexión se dará por concluida.

3.3.2. Protocolo UDP

De entre los protocolos de transporte sobre IP, el User Datagram Protocol (UDP) es el mas básico de todos.

Orientado al simple envío de mensajes UDP ofrece un método de operación muy sencillo para el acceso al protocolo de red. Sin embargo UDP no garantiza la recepción del mensaje, con lo que el origen de los datos no tiene forma de saber mediante el propio protocolo en qué estado se encuentra la transacción de datos, ni incluso si estos han salido a la red. Lo que sí garantiza UDP

es el multiplexado, la verificación de errores y una minima proporción del tamaño de la cabecera respecto de la trama útil.

Sin embargo si se desea obtener algún mecanismo para el control de la entrega de mensajes debe implementarse a nivel de aplicación.

Bit	0 al 15	16 al 31
0	Puerto Origen	Puerto Destino
32	Longitud Mensaje	CheckSum
64	Trama de Datos	

Figura 3.6: UDP: Diagrama de cabecera

En su cabecera UDP sólo necesita de 2 campos obligatorios y otros 2 opcionales (con letras grises en el diagrama). Cada uno de estos datos ocupan 16 bits, con lo que las cabeceras tienen un tamaño fijo de 64bits u 8bytes, lo que lo hace muy ligero respecto de otros protocolos de transporte. Los campos para indicar el número de puerto sirven para identificar la aplicación que estará a la escucha, ya que suelen reservar un número de puerto pactado previamente. El puerto origen suele utilizarse para indicar dónde deben enviarse las respuestas, pero es opcional ya que muchas veces los protocolos basados en UDP no esperán retorno por parte del destino. Para dejar indicado que no hay puerto de retorno debe dejarse a 0 este campo. El otro campo opcional, el checksum, es una suma de verificación sobre los datos de cabecera (incluyendo las direcciónes y el protocolo en la cabecera IP, longitud y los datos) Este campo no suele ser deshabilitado. El campo de longitud recoge precisamente la longitud del propio segmento de datos incluido en el paquete.

Este protocolo es ampliamente utilizado en el streaming de multimedia y en los entornos de juegos en red, donde se busca alcanzar la mayor velocidad de transferencia y de que se garantice la solidez de los datos.

3.4. Protocolos de aplicación

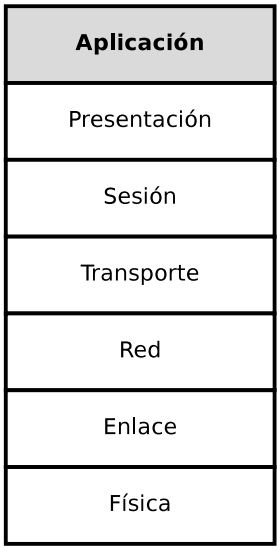

Figura 3.7: Niveles OSI: protocolos de aplicación

En este proyecto se van a implementar una serie de protocolos para demostrar la versatilidad de la plataforma y sobre todo la comodidad y la limpieza de código que aportan. Se pretende conseguir una plataforma que facilite la implementación de servicios para protocolos a nivel de Aplicacion.

Limitando la comunicación al nivel de aplicación, todos los protocolos existentes pueden resumirse como una secuencia de emisión y recepción de datos en una serie de etapas y formatos establecidos por el protocolo.

De esta manera se puede entender al servicio como un autómata sensible al contexto para cada conexión que recibe. Orientando el diseño del servicio como una secuencia de pasos y que éstos sean lo más simples y claros posible, se ahorrará mucho tiempo y complejidad en la producción del código.

Los protocolos elegidos para demostrar la idoneidad de la plataforma son los siguientes:

3.4.1. Telnet

Se trata de un clásico obligado entre cualquier conjunto de servicios de red (o su sustituto Secure SHell). La función de este servicio consiste en proveer de autentificacion y acceso al control de las terminales de un sistema.

TELNET (TELecommunication NETwork) es un protocolo de red utilizado en conexiónes de Internet o en redes de area local (LAN). Fue desarrollado en 1969 y especificado en el RFC 15 y después estándarizado en el IETF STD 8, uno de los primeros estándares de internet.

El término telnet también se refiere al software que implementa a la parte cliente del protocolo. Los clientes TELNET están disponibles en prácticamente cualquier plataforma. La mayoría de los equipos de red que dispongan de una pila TCP/IP tienen soporte para algún tipo de servidor TELNET para permitir su configuración remota (incluyendo a los basados en Windows NT). Debido a los problemas inherentes para mantener la privacidad, TELNET se ha visto reemplazado ampliamente por SSH (Secure SHell).

El término “hacer un telnet” es la forma de llamar a establecer una conexión o utilizar TELNET u otras conexiónes TCP interactivas. Como ejemplo: “Para cambiar tu contraseña, haz telnet al servidor y ejecuta el comando passwd”.

Muy a menudo un usuario hará telnet a un sistema tipo Unix o a un simple dispositivo de red como un router. Por ejemplo un usuario podría “hacer telnet desde casa para mirar su correo en el colegio”. De la misma manera podria utilizar un cliente telnet para conectar desde su equipo de escritorio con sus servidores. Una vez que la conexión se ha establecido, el usuario podrá ingresar con sus datos de acceso y ejecutar comandos en el sistema remoto.

En algúnos sistemas la aplicación cliente puede utilizarse para realizar sesiones TCP en bruto. Comúnmente se cree que una sesion telnet que no incluye un IAC (el caracter 255) tiene la misma utilidad. Este no es el caso ya que debido a NVT (Terminales Virtuales de Red) con reglas especiales como el empleo constante de los caracteres 0 o 13.

3.4.2. HTTP: HiperText Transfer Protocol

Elaborado por el W3C y la IETF el HTTP es seguramente uno de los servicios de red más extendido en internet y en redes internas. Define las normas de comunicación entre los participantes del servicio (clientes, servidores y opcionalmente proxies). Servirá como ejemplo para abordar la programación de un protocolo más avanzado y permita su uso desde los diversos tipos de navegadores.

El Hypertext Transfer Protocol (HTTP) es un protocolo de comunicaciones para transferencia de información a través de Internet. Su utilización para recuperar documentos de texto vinculados (hipertexto) ha llevado al establecimiento de la World Wide Web.

El desarrollo del HTTP fue coordinado por el Consorcio World Wide Web (W3C) y la Internet Engineering Task Force (IETF), culminando en la publicación de una serie de RFCs, siendo el más destacable el RFC 2616, que en Junio de 1999 definió el HTTP versión 1.1, que es la que más ampliamente se utiliza en la actualidad.

HTTP es un estándar petición/respuesta entre un cliente y un servidor. El cliente será el usuario final mientras que el servidor es el sitio web (web site). El cliente, realizando una petición HTTP mediante un navegador web, araña web u otras herramientas, se le denomina user agent. El servidor correspondiente, que almacena o crea recursos como ficheros HTML e imágenes, es denominado servidor de origen. Entre

el user agent y el servidor origen pueden aparecer varios intermediarios, como son los proxys, pasarelas y tuneles. El HTTP no está limitado a su utilización sobre TCP/IP y sus capas, más aún es la aplicación más extendida en Internet. Ciertamente HTTP puede implementarte sobre otros protocolos en Internet en otras redes. HTTP sólo asume que se encuentra sobre un protoclo de transporte confiable, con lo que HTTP puede utilizarse sobre cualquier protocolo que lo garantice.

Normalmente es el cliente HTTP el que inicia la petición. Se establece una conexión TCP a un puerto particular en un servidor (el puerto 80 por defecto) El servidor HTTP escuchando en ese puerto esperará a que el cliente emita el mensaje de petición. Una vez recibida la petición el servidor responde con una línea de estado, como por ejemplo “HTTP/1.1 200 OK”, un mensaje propio, el cuerpo de lo que probablemente sea el fichero solicitado, mensajes de error si procede y otras informaciónes.

La razón por la que HTTP se basa en TCP y no en UDP es debida a la cantidad de datos que se generan al enviar una página web y a que TCP provee control de transmisión, envría los datos en orden y provee corrección de errores.

Los recursos a los que se puede acceder mediante HTTP se identifican mediante Identificadores Unificados de Recursos (URIs) y en concreto Localizadores Unificador de Recursos (URLs).

3.4.3. SIP: Session Initiation Protocol

Es un protocolo desarrollado por el IETF MMUSIC Working Group en la búsqueda de un estándar para iniciar, finalizar o modificar sesiones interactivas de usuario, que se emplearan para contactar con dichos usuarios. Normalmente se utiliza en conjunción con otros servicios, principalmente multimedia (video, voz, mensajeria instantanea). Se ha convertido en el protocolo de señalización para voz por IP más utilizado junto al H.232.

El Session Initiation Protocol es un protocolo de señalización, ampliamente utilizado para iniciar y desconectar sesiones de comunicaciones multimedia, como llamadas de voz y video a través de internet. Otras aplicaciones interesantes incluyen conferencias de video, streaming multimedia, mensajeria instantánea, información de presencia y juegos online. En Noviembre del 2000 SIP fue adoptado como protocolo de señalización para 3GPP y elemento permanente de la arquitectura IMS para streaming multimedia sobre IP para telefonos celulares.

Este protocolo puede utilizarse para crear, modificar y finalizar sesiones unicast o

multicast consistentes en uno o varios flujos de medios. La modificacion puede suponer cambiar direcciónes o puertos, invitar a otros participantes, añadir o retirar flujos de medios, etc.

SIP puede situarse sobre los proctolos TCP, UDP o SCTP. Fue originariamente diseñado por Henning Schulzrinne (Universidad de Columbia) y Mark Handley (UCL) en 1996. La última versión de la especificación es el [7] RFC 3261 del IETF SIP Working Group.

Cabe destacar que SIP funciona en modo texto, lo que hace más facil analizar su funcionamiento.

Capítulo 4

Análisis y Diseño

4.1. Análisis

El modelo Cliente-Servidor es tan básico como antiguo. Se pueden considerar que los primeros sistemas de este tipo eran los teletipos utilizados en las universidades para emitir programas a los equipos que allí habían instalados y recibir posteriormente sus resultados. A diferencia de hoy, aquellos sistemas funcionaban directamente sobre enlaces dedicados y cada uno de estos teletipos disponía de una conexión permanente.

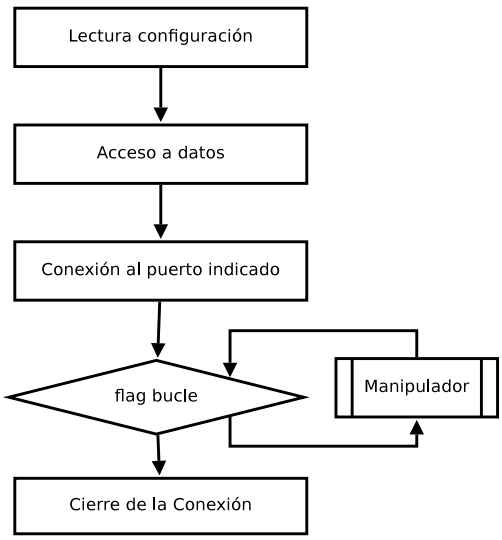

Figura 4.1: Niveles OSI

Sin embargo, la implantación de sistemas de red por capas, enlaces no dedicados y sobre todo la diversificación de equipos mucho mas baratos y potentes han modificado el panorama de manera importante: Además de diversificarse los tipos de servicios era posible que varios de ellos coexisitieran en el mismo equipo.

En el contexto de los sistemas informáticos un protocolo es un conjunto de reglas predefinidas, cuyo proposito es estándarizar actividades y procesos. Siguiendo un mismo protocolo se garantiza que se mantendrá la compatibilidad entre los sistemas involucrados, independientemente del medio sobre el que estén en con-

tacto, posiblemente otros protocolos.

La forma clásica de programar un servicio consiste en un bucle de escucha que inicia subprocesos para cada conexión entrante y que finaliza cuando esta acaba o se produce un error. Se presentan dos entidades en este caso; la del servicio, que incluirría las rutinas para iniciar la escucha y el acceso a los datos, y la del manipulador, que atiende la conexión siguiendo las pautas del protocolo.

4.1.1. Servicio

Al arrancar un servicio antes de iniciar el bucle de escucha deben realizarse una serie de procedimientos:

1. La lectura de parámetros desde línea de comandos
2. La lectura de parámetros desde el archivo de configuración
3. El registro de actividad en el/los archivo(s) pertinente(s)
4. Preparar el acceso a los datos necesarios, ficheros o bases de datos

Parámetros

Existen parámetros que son comunes prácticamente a cualquier servicio, por ejemplo:

1. Fichero de configuración.
2. Número de puerto para la escucha.
3. Número máximo de conexiónes simultáneas.
4. Ficheros de registro, eventos, errores, etc.
5. Nivel de locuacidad (verbosity) en los registros.

O parámetros particulares para según qué tipos de servicio:

1. Acceso al medio de datos.
2. Parámetros de configuración concretos para este servicio.

Debe tenerse en cuenta que los parámetros en el fichero de configuración tienen menor prevalencia que los indicados por línea de comandos, salvo el que especifique precisamente qué fichero de configuracion debe leerse.

Ficheros de configuración

Debe acordarse una estructura común a todos los ficheros de configuración que se parsearán desde la plataforma pero permitiendo a la vez mantener la genericidad suficiente como para no perder la flexibilidad que haga este sistema util a cualquier implementación de servicios y que a su vez permita que los usuarios puedan entender el fichero de forma clara y manipularlo facilmente.

Se espefican con ese fin las siguiente sintaxis:

$$< nombreparametro >=< valor > \{, < valor - N >\}$$

Para los valores que deben utilizar los parámetros se pueden utilizar comodines para ampliar la versatilidad:

1. %H - Hostname
2. %i - Ip de la conexión entrante
3. %p - Puerto de origen de la conexión
4. %d - fecha actual
5. %t - hora actual

Locuacidad

Para que puedan establecerse mensajes de depuración a distintos niveles debe adoptarse un sistema de categorías o prioridades para estos. En cada etapa o proceso del servicio y del manipulador se emiten mensajes que incorporan el nivel de locuacidad para el que son generados, será la clase encargada de mostrar mensajes la que los muestre segun si el nivel de estos es menor o no que el indicado. Se proponen los siguientes niveles

1. 0 - Básico: mensajes de operaciones mínimo, inicio del sistema, errores en los procedimientos.
2. 1 - Moderado: mensajes comunes, entrada en servicio de un manipulador y puntos clave del desarrollo de la atención.
3. 2 - Productivo: indica la entrada en la mayoría de las etapas de un servicio. Se amplian los detalles de los mensajes de error.

4. 3 - Expresivo: indica la entrada a cada etapa del servicio, se detalla el contexto de cada mensaje de error.

5. 4 - Locuaz: detalla el contexto de cada etapa.

Registro

Con la intención de simplificar la tarea de gestionar la salida de mensajes, bien a la salida de la consola o a ficheros de registro, deben implementarse clases destinadas a tal fin, que vuelquen o no los mensajes al medio según la locuacidad que se ha fijado y que en caso de que el medio predeterminado falle (por falta de espacio o cambio de permisos) sea capaz de cambiar a otro (indicado en la configuración o a los medios salida por pantalla dedicados a tal fin).

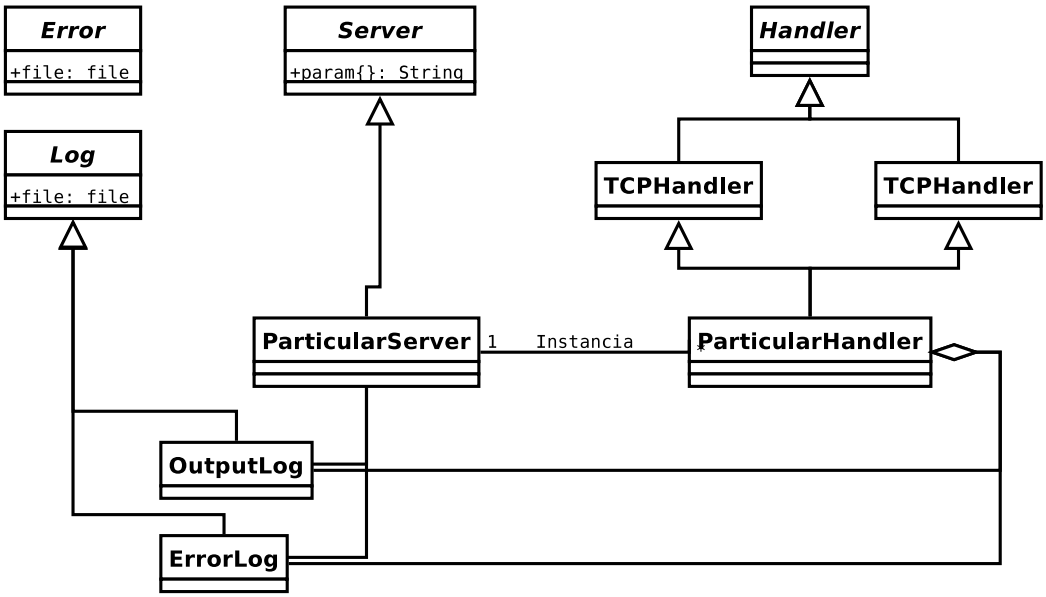

Figura 4.2: Diagrama de Clases de Análisis

4.1.2. Manipulador

Precisamente es la naturaleza de un protocolo, como un conjunto ordenado de etapas en las que se organiza el flujo de información entre los sistemas, la que permite abstraer al programa servidor como un autómata sensible al contexto. Cada etapa del manipulador equivale al estado del autómata y pueden realizarse un conjunto determinado de acciones:

1. Recibir datos desde la conexión
2. Emitir datos a la conexión
3. Tratar, almacenar o recuperar datos
4. Avanzar, saltar o retroceder a otra etapa

La recepción de los datos puede además controlarse mediante expresiones regulares según las especificaciónes del protocolo, detectando rapidamente errores en la entrada. Debe facilitarse además la adopción de distintos tipos de intercambio de datos, controlados, formato en bruto o encriptados para evitar reimplementaciónes de esta clase de rutinas.

En el diseño del manipulador debe tenerse en cuenta que normalmente existe un orden convencional en las etapas. Previsiblemente estas deberían contener un número limitado de procedimientos, para reducir la complejidad.

Durante la implementación de los protocolos ha aparecido la necesidad de trabajar sobre UDP. Para permitir esta funcionalidad y mantener un diseño eficiente se ha optado por dividir el manipulador en tres nuevas clases que permitieran trabajar tanto sobre el protocolo TCP como UDP y en concordancia a sus características:

Handler la clase Handler primitiva se centra en la gestión de estados, el tránsito entre estos, y su inicialización.

TCPHandler esta clase que hereda de Handler contiene una versión especial de las rutinas de emisión y recepción de datos. Se inicializa pasando el hash del socket.

UDPHandler Cada vez que un datagrama llega al servidor se instancia al manipulador, integrando la dirección de origen del datagrama y los datos que portaba, esta clase tendra sus propias rutinas de emisión y recepción. Debe tenerse en cuenta que en este tipo de manipulador se hace necesario algún sistema que permita recuperar desde el servidor en qué estado se encontraba al finalizar la última transmisión.

4.1.3. Flujos de los protocolos

A continuación se va a especificar el flujo de los protocolos con los que se va a probar la plataforma.

Telnet

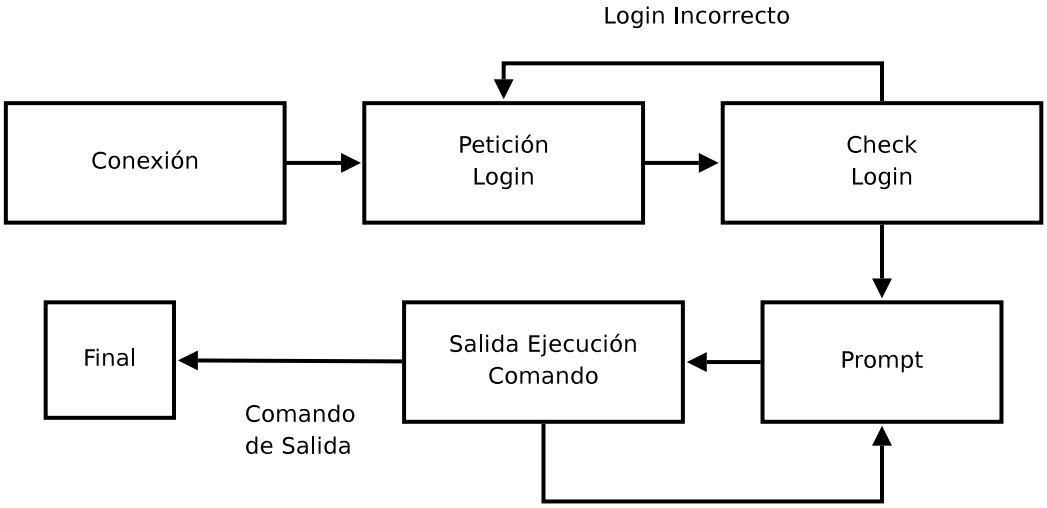

Figura 4.3: Flujo de desarrollo del protocolo Telnet

El protocolo Telnet tiene un flujo básico de desarrollo muy simple:

1. Se establece la conexión con el cliente
2. Se presenta el diálogo de petición de contraseña y se espera a que se introduzcan los datos.
3. Se checkea la contraseña contra la base de datos correspondiente
4. Se solicita el comando a ejecutar en el sistema
5. Se ejecuta el comando pertinente y se muestra su salida. y se vuelve al paso 4.

Existe un pequeño conjunto de pasos alternativos:

1. (4.Alternativo) El usuario debe repetir la autentificación.
2. (5.Alternativo) Se ha solicitado finalizar la sesión con lo que procedemos a la desconexión.

HyperText Transfer Protocol

El protocolo HTTP/1.1 tiene un flujo básico bastante más amplio, ya que después de recibir la orden las rutinas que atenderán el comando difieren drásticamente. En consecuencia deberán implementarse nuevos estados para cada tipo de comando.

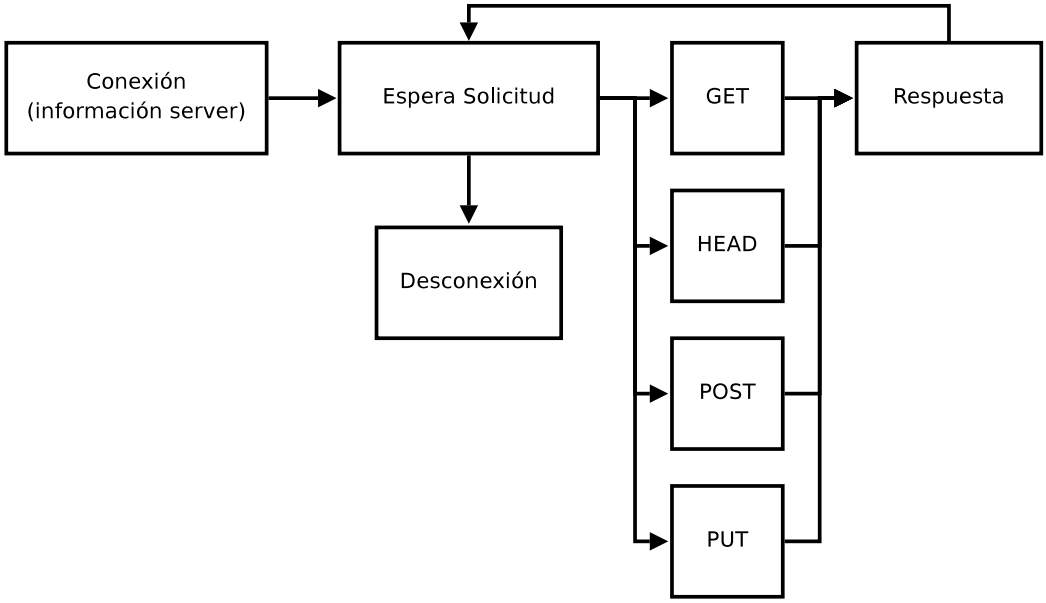

Figura 4.4: Flujo de desarrollo del protocolo HTTP

1. Se establece la conexión con el cliente
2. Se muestra la información del servidor
3. Se reciben las cabeceras y el comando
4. En el estado de atención se recibe el resto de la información (si procede)
5. Se responde informando del estado.

Aunque la conexión pueda perderse inesperadamente en cualquier momento, es en el momento de esperar el comando cuando se produciría una desconexión correcta.

Session Initialization Protocol

El protocolo SIP es responsable de varias tareas, aunque puede operar sobre UDP la implementación sobre BLAS solo funcionará sobre UDP.

Es precisamente en este punto por la necesidad de trabajar sobre UDP cuando se ha tomado la decisión de dividir el manipulador en tres nuevas clases que permitieran trabajar tanto sobre el protocolo TCP como UDP y en concordancia a sus características.

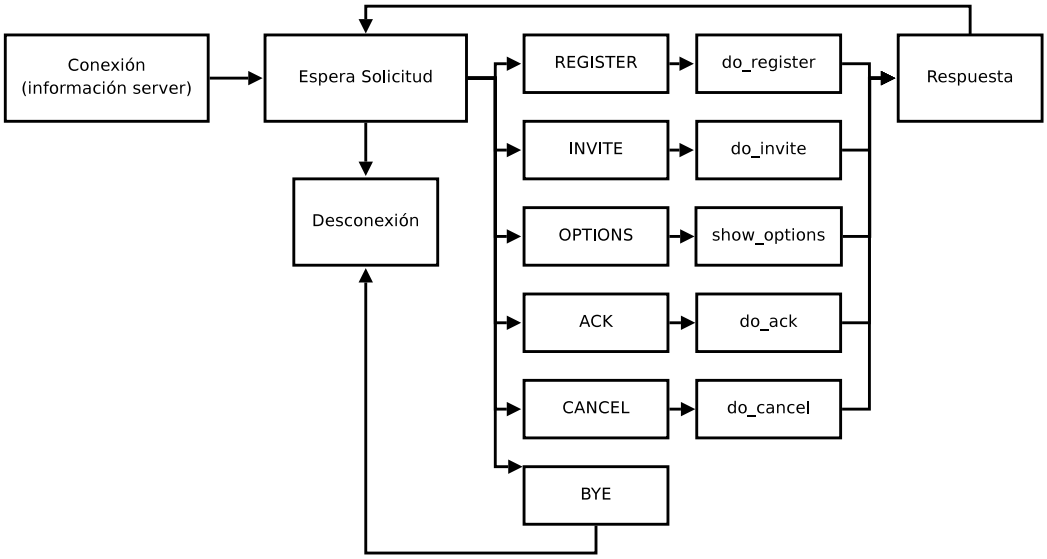

Figura 4.5: Flujo de desarrollo del protocolo SIP

1. Registro(REGISTER): El usuario se da de alta en el sistema y figura en el directorio de usuario.

2. Invitación(INVITE): Dirigido a un usuario se le autoriza a acceder a un medio compartido. Debe aceptarse con ACK o denegarse con CANCEL.

3. Opciones(OPTIONS): Solicita al servidor el listado de las funcionalidades.

4. Cierre(BYE): Avisa de que va a procederse con la desconexión.

4.1.4. Acceso a datos

Muchos servicios tienen la necesidad de leer y escribir en medios de datos centralizados, por eso es importante diseñar según se debe hacer un esfuerzo en desarrollar un modelo de acceso a datos que permita una lectura lo más transparente posible, facilitando la lectura del código.

Normalmente existen unos tipos limitados de medios de datos:

1. Registos unificados volátiles: Son medios de datos que se borrarán cuando se desactive el servicio, deben permitir el acceso concurrente por parte de los hilos. Este tipo de medios sería muy adecuado para sistemas de chat, etc.

2. Accesos a bases de datos: Python tiene soporte para acceder a muchos tipos de bases de datos: Mysql, Postgresql, ODBC, etc. Sin embargo, diseñar un acceso

a datos mediante objetos que eviten la manipulación de peticiones SQL en el código de la atención del servicio será básico para la comprensión del código y su depuración.

3. Acceso remoto a datos: En ocasiones los datos que se necesitan estarán en sistemas remotos, accediendo a ellos mediante conexiónes IP/TCP y normalmente sobre estos métodos HTTP, SOAP o REST. Al igual que en otros medios de acceso es importante que el método de acceso se abstraiga del código de atención del servicio

En definitiva y aunque para cada tipo de servicio el modelo de acceso a datos varía considerablemente se pueden establecer las siguientes líneas para el analisis y la orientación que tomará el sistema:

1. Claridad: Debe evitarse la molesta aparición de líneas SQL, peticiones HTTP, XML y similares en el código.

2. Orientación a objetos: Un buen modelo de acceso a datos reducirá sensiblemente el tiempo que se requiera para depuración y posteriores.

3. Manejo de errores: Es importante integrar el manejo de excepciones en las rutinas de adquisición de datos ya que no estarán exentas de fallos.

4. Oportunidad de acceso: Deben elegirse adecuadamente en qué momentos se accederá a los datos, para no producir latencias en puntos del proceso sensibles al tiempo de respuesta. Además si el dispositivo no dispone de control de acceso al medio debe implementarse algún método que controle la concurrencia.

4.2. Diseño

4.2.1. Librería principal

Existe una parte común a toda la plataforma que contiene las clases abstractas, puede verse su diagrama de clases de diseño en la figura 4.6

En estas clases se incorporarán todas las funcionalidades que puedan emplearse desde cualquier implementación de un servicio. Para simplificar el procedimiento de herencia de componentes se incluirán todas en el archivo core.py, que acompañaría al

archivo que contenga el código del servicio en sí mismo en cada despliegue que se haga a posteriori del sistema.

Server

Esta clase incluye las rutinas necesarias para la inicialización y configuración del servicio:

1. Carga y parseado de los parámetros de línea de comando.
2. Carga e interpretación de los archivos de configuración.
3. Reserva del socket al que se conectarán los clientes del servicio.
4. Bucle principal de inicializacion y despacho de los hilos.
5. Inicialización de los registros de eventos y errores del sistema.
6. Recopilación de las cadenas de documentación relacionadas con cada parámetro del sistema.
7. Inicialización de los medios de acceso a datos comunes (si procede).

Error

Esta clase que hereda de la clase de sistema Exception se utiliza como tipo de datos por defecto en caso de que algúno de los procesos o rutinas del sistema no pueda realizar su función, en estos casos devolverá un objeto de clase Error.

Además Error contiene 2 datos importantes para la infraestructura:

msg Cadena que contiene una cadena explicando el suceso que originó el error.

level Entero que se utiliza para saber si la gravedad del suceso obliga a su aparición en los registros.

Los métodos son una sobrecarga de los métodos básicos de la clase Exception:

__init__ Que inicializa el objeto proveyendo de la información con el mensaje de error y el nivel de importancia.

__str__ Devolverá la cadena con la explicación del error y el nivel de importancia cuando se necesite imprimirlo en el registro.

Log

En esta versión del registro solo se volcarán los eventos y errores a salida estándar o a ficheros. Para reducir el acoplamiento en las llamadas en los objetos de registro integran los siguientes datos durante su tiempo de vida:

file File salvo que su valor sea None será el descriptor de fichero donde se volcará la información.

level Int indicará el nivel de importacia que debe igualarse o superarse para que el evento se vuelque en el registro.

Los métodos propios de los objetos tipo Log son:

__init__() Método de inicialización del registro, indicando el nombre del fichero de volcado y el nivel mínimo de importancia. Si la apertura del fichero falla se pasarán a volcar los datos a la salida estándar.

put() Con este método se vuelcan eventos al registro, además del mensaje debe acompañarse del nivel de importancia del evento. Para simplificar su invocación normalmente se utiliza un puntero desde self.log[“report”].put() a self.report() con lo que las llamadas son mucho más cortas.

Handler: UDP y TCP

Seguramente las clases más importantes que se incorporan en un servicio:

1. Handler: Es la clase principal, hereda de la clase hilo (Thread) y sus principales funcionalidades son:

   a) Controlar el avance, retroceso y tránsito entre estados del manipulador en ejecución.

   b) Sobrecargar las clases mínimas de la clase Thread para un funcionamiento coherente.

2. TCPHandler: Versión del manipulador que contiene los atributos y métodos concretos para tratar con el socket que les corresponda. En este caso basta con guardar el puntero al Socket, ya que es lo único que necesitan las rutinas send() y receive() para enviar y recibir datos con el cliente.

3. UDPHandler: Versión del manipulador destinada a atender las peticiones UDP. En el caso de protocolo UDP es más complicado ya que no existe un Socket como tal. En su lugar se guarda la dirección ip y puerto de donde procedía la comucación y ademas el buffer de los datos recibidos, con lo que los procedimientos de envío y recepción de datos tienen las siguientes particularidades:

   a) send(): debe crear un nuevo socket UDP con los datos de conexión que posee, estos pueden facilitarse desde los parámetros o utilizar los que figuran como origen de los datos. No existe confirmación sobre si los datos han llegado por lo que la aplicación debe buscar su forma de confirmar la transmisión.

   b) receive(): Todos los datos que puedan llegar al hilo de atención llegan de una sola vez desde el momento en que se inicia, se van consumiendo según especifique el patrón que se pasa como parámetro. Si el buffer de datos está vacío se devolverá un objeto Error.

4.2.2. Servicio de ejemplo

A continuación se presenta un ejemplo de como se realiza la implementación de un servicio heredando de las clases de BLAS. En la figura 4.7 puede observarse como se relacionan las clases.

El arranque del sistema se realiza de la siguiente manera:

1. Se carga el fichero con el intérprete de Python, presumiblemente el fichero se llamaría particular.py.

2. Una vez finalizadas las importaciones de los modulos se cargan las definiciones de clases y sus métodos.

3. Cuando se llega a la zona de ejecución se instancia un objeto ParticularServer con los argumentos recibidos por línea de comandos. Después se invoca al método mainloop indicándole la clase de manipulador que debe instanciar.

4. Durante la inicialización de un objeto ParticularServer se instancian al mismo tiempo los objetos de registro, a la vez que se llama a la inicializacion de Server, que será común a todos los servidores. Aquí se parsearán las opciones de la línea

de comandos, cuyas directrices estarán distribuidas en el código de ParticularServer y Server en métodos que inician su nombre con “config_” seguido del nombre del parámetro, en su campo de documentación se incluye el prefijo que debe utilizarse en la línea de comandos para modificar estos parámetros.

5. En mainloop() discriminando el tipo de manipulador(TCP o UDP) abrirá el tipo de socket pertinente e iniciará un bucle while. Dentro de estas conexiónes se irán atendiendo por parte de los manipuladores. Para que los manipuladores puedan hacer su trabajo deben primero inicializarse y luego emitir la orden de arranque.

6. Al inicializar el hilo del manipulador se le aporta bien el socket de conexión para los tipo TCP y el buffer y la dirección de origen para los UDP. El método start() que se hereda de la clase Thread iniciará la ejecución del hilo que se desarrollará a lo largo de los estados que se deban seguir para atender la petición.

7. la mayor parte del código del manipulador serán métodos cuyo nombre empiece port “step_” seguido del identificador del paso en si mismo, en cada paso debe indicarse si se pasa al siguiente, al anterior o se salta a otro; de otra manera se repetirá el mismo paso.

4.2.3. Servicio Telnet

Para imitar un servicio de terminal interactiva con autentificación 4.8

4.2.4. Servicio HTTP

Bifurcación y flujo de estados

Una vez que el servidor ha recibido la trama con la petición e instanciado el hilo de atención correspondiente se inicia el siguiente proceso:

Request En este paso se recibe la primera línea de la petición, se informa en el registro del origen de la transmisión y se almacena el comando en los datos del hilo. Se pasa a la etapa Run.

Headers Este paso irá recolectando y parseando las cabeceras de las peticiones, no se pasará al siguiente paso Run hasta que se encuentre con una cabecera en blanco.

Run La función de este estado es elegir a cual se pasará según el comando que se ha recibido.

Run_Get En este paso se relativizará la ruta solicitada a la que está configurada como raiz del servidor, además si no se ha indicado un fichero se intentará localizar “index.html”. Si al intentar la apertura del fichero se produce una excepción será capturada y se emitirá al cliente una respuesta “404 Not Found”. En caso de que la apertura del fichero sea correcta se emitirá antes un mensaje “200 OK”, las cabeceras de información mínimas, incluyendo el tamaño del contenido a transmitir y finalmente el cuerpo de la petición. A continuación se indicará que el estado siguiente es “End”.

End Esta parte del proceso guardará la información obtenida de los pasos anteriores y emitirá el suceso del fin de la petición al registro de eventos.

Puede encontrarse el diagrama de clases de diseño en 4.9

4.2.5. Servicio SIP

El diagrama de diseño de clases del servicio SIP esta en la figura 4.10

Bifurcación y flujo de estados

Una vez que el servidor ha recibido la trama con la petición e instanciado el hilo de atención correspondiente, se inicia el siguiente proceso:

Request En este paso se recibe la primera línea de la petición, se informa en el registro del origen de la transmisión y se almacena el comando en los datos del hilo. Se pasa a la etapa Run.

Headers Este paso irá recolectando y parseando las cabeceras de las peticiones. No se pasará al siguiente paso Run hasta que se encuentre con una cabecera en blanco.

Run La función de este estado es elegir a cual se pasará según el comando que se ha recibido.

Run_Register Este paso acaba procesando cualquier comando Register. Si no hay autentificación, la solicita generando antes un código “nonce” de desafio para que el cliente responda. A continuación emite un “401 Unauthori-

zed" con la información del desafio. En caso de que el cliente haya aportado autentificación se llama al método "digest" para realizar el contraste del campo "response" y la respuesta producida por el propio servidor. A continuación se procederá al estado "End".

Run_Subscribe En este estado se guarda información de usuario en el registro para aviso de eventos. A continuación se pasa al estado "End".

Run_Invite En este estado, si tanto el usuario origen como destino estan validados, se pasa al estado "Invite_Request". En caso contrario se pasará a "End".

Invite_Request En este estado se enviará una petición al destinatario para que acepte la solicitud, se pasará al estado "End" después de marcar la invitacion como pendiente.

Run_Ack Se revisa en el registro de invitaciones pendientes si existe registro con esa referencia. De ser así se enviarán una confirmación al emisor de la misma y se pasará al estado "End".

End Esta parte del proceso guardará la información obtenida de los pasos anteriores y emitirá el suceso del fin de la petición al registro de eventos.

Autentificación

El procedimiento de autentificación en SIP se realiza a través de un mecanismo heredado de HTTP Digest. Para que se produzca la autentificación se desarrolla la siguiente secuencia:

1. El cliente pide registrarse en el servicio. No provee método de contraste algúno.

```
REGISTER sip:192.168.1.100 SIP/2.0
Tue Aug 26 21:09:38 2008:192.168.1.11:5072:current state:receiving param
CSeq: 127 REGISTER
Via: SIP/2.0/UDP 192.168.1.11:5072;branch=z9hG4bK2686af38-1072-dd11-9855-0016e6dfa1a5;rport
User-Agent: Ekiga/2.0.12
From: <sip:edorka@192.168.1.100>;tag=4279af38-1072-dd11-9855-0016e6dfa1a5
Call-ID: 205faf38-1072-dd11-9855-0016e6dfa1a5@Newton
To: <sip:edorka@192.168.1.100>
Contact: <sip:edorka@192.168.1.11:5072;transport=udp>
Allow: INVITE,ACK,OPTIONS,BYE,CANCEL,NOTIFY,REFER,MESSAGE
Expires: 3600
Content-Length: 0
Max-Forwards: 70
```

2. El servidor contesta con un error “401 Unauthorized“ y provee en las cabeceras del desafio de autentificación.

```
SIP/2.0 401 Unathorized
WWW-Authenticate: Digest realm=‘‘192.168.1.100’’, nonce=‘‘ff8c..’’, qop=‘‘auth’’
CSeq: 102 REGISTER
Via: SIP/2.0/UDP 192.168.1.11:5069;branch=z9hG4bKc2448342-...;rport=5069
From: <sip:edorka@192.168.1.100>;tag=5c418342-f671-dd11-9855-0016e6dfa1a5
Call-ID: 00a1a017-e771-dd11-9855-0016e6dfa1a5@Newton
To: <sip:edorka@192.168.1.100>;tag=5c418342-f671-dd11-9855-0016e6dfa1a5
Contact: <sip:edorka@192.168.1.11:5069;transport=udp>
Content-Length: 0
Server: PythonSIP prototype
```

3. El cliente reintenta el registro en el servidor incluyendo en la cabecera un campo ”Authorization” con los datos que ha empleado para calcular el campo “response”.

```
REGISTER sip:192.168.1.100 SIP/2.0
CSeq: 129 REGISTER
Via: SIP/2.0/UDP 192.168.1.11:5072;
branch=z9hG4bK2870b438-1072-dd11-9855-0016e6dfa1a5;rport
User-Agent: Ekiga/2.0.12
Authorization: Digest username=‘‘edorka’’, realm=‘‘192.168.1.100’’,
nonce=‘‘ff8cf8e764fca9134c1cd5499bc4684b’’, uri=‘‘sip:192.168.1.100’’,
algorithm=md5, response=‘‘365cf4061c002b09c75408a4907ac911’’,
cnonce=‘‘3261b438-1072-dd11-9855-0016e6dfa1a5’’,
nc=‘‘00000001’’, qop=‘‘auth’’
From: <sip:edorka@192.168.1.100>;tag=4279af38-1072-dd11-9855-0016e6dfa1a5
Call-ID: 205faf38-1072-dd11-9855-0016e6dfa1a5@Newton
To: <sip:edorka@192.168.1.100>
Contact: <sip:edorka@192.168.1.11:5072;transport=udp>
Allow: INVITE,ACK,OPTIONS,BYE,CANCEL,NOTIFY,REFER,MESSAGE
Expires: 3600
Content-Length: 0
Max-Forwards: 70
```

4. Si el servidor da por buena la respuesta al desafio, después de calcularlo por su cuenta emitirá una respuesta “200 OK” en referencia a la última petición.

```
SIP/2.0 200 OK
CSeq: 107 REGISTER
Via: SIP/2.0/UDP 192.168.1.11:5069;
branch=z9hG4bK460a1e59-0572-dd11-9855-0016e6dfa1a5;rport=5069
From: <sip:edorka@192.168.1.100>;tag=da541c59-0572-dd11-9855-0016e6dfa1a5
Call-ID: 00a1a017-e771-dd11-9855-0016e6dfa1a5@Newton
To: <sip:edorka@192.168.1.100>;tag=da541c59-0572-dd11-9855-0016e6dfa1a5
Contact: <sip:edorka@192.168.1.11:5069;transport=udp>
Content-Length: 0
Server: PythonSIP prototype
```

Para calcular el response correcto se ha implementado un método digest() al que se pueden pasar como argumentos el método que lo ha llamado y la contraseña que se debe comprobar, el proceso de "digestión" es el siguiente

1. $S1 = MD5(< username >:< realm >:< password >)$
2. $S2 = MD5(< metodo >:< URI >:)$
3. $RESP = MD5(S1 :< nonce >:< nc >:< cnonce >:< qop >: S2)$

4.3. Manual de usuario

Para ejecutar cualquiera de los servicios debe situarse en la raiz del CD que acompaña a la memoria y ejecutar el interprete de Python indicando el fichero del servicio. Por ejemplo:

```
Python telnet.py
Python http.py
Python sip.py
```

Debe tenerse en cuenta que para puedan realizarse operaciones de escritura en el medio con información de los registros de eventos y errores, deben copiarse al disco duro los ficheros contenidos en el CD y ejecutar desde ahí.

El interprete incluido en el CD puede funcionar sin instalación. Sin embargo se ha incluido una copia del instalador para windows de la última versión disponible de Python.

Al ser multiplataforma el interprete y con él, el entorno, sabiendo que las reglas y permisos de cada sistema serán diferentes y para evitar problemas en la ejecución se han modificado los puertos de escucha, con lo que telnet estará a la escucha en el puerto TCP 2300 y el servidor HTTP en el puerto TCP 8000. El protocolo SIP estará a la escucha en su puerto habitual, 5060 UDP.

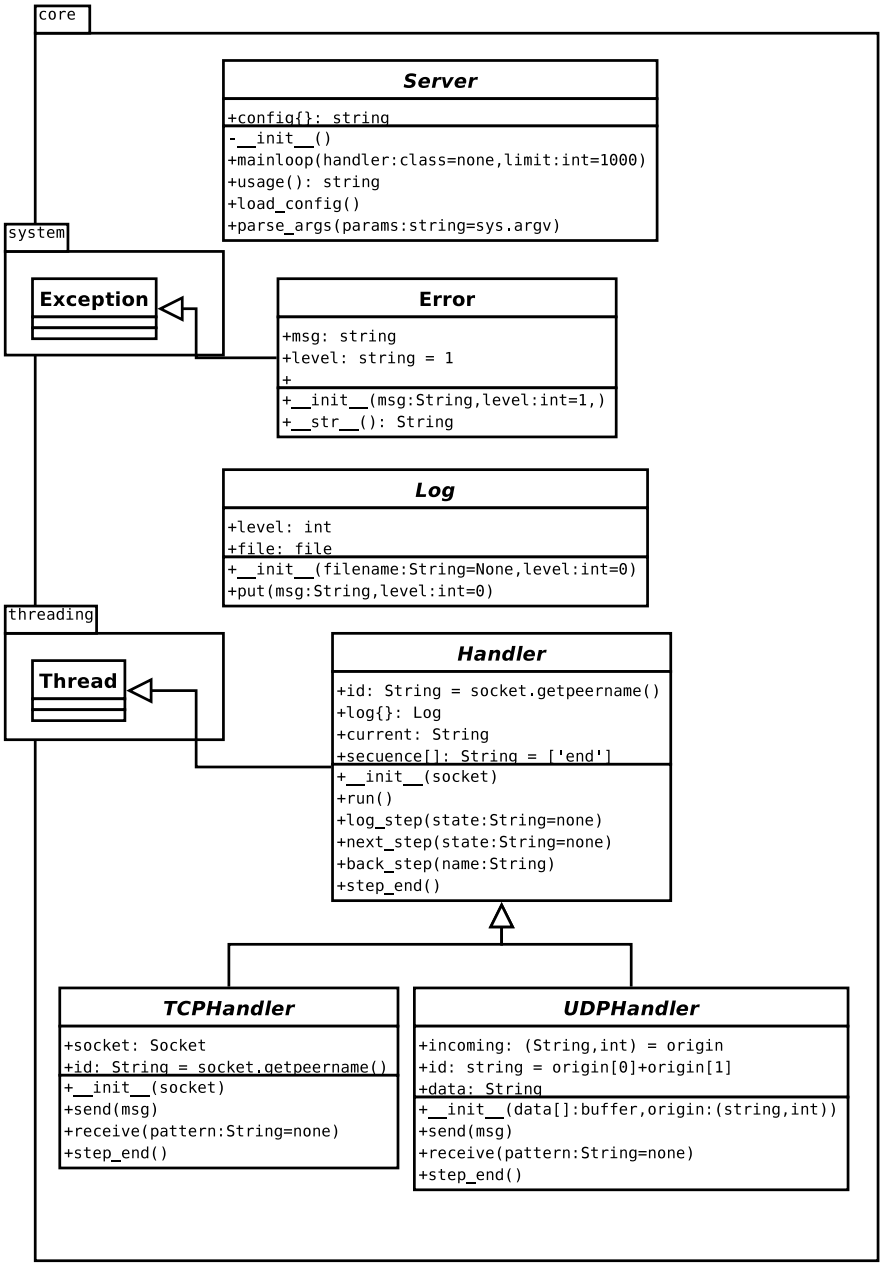

Figura 4.6: Diagrama de Clases de la librería principal

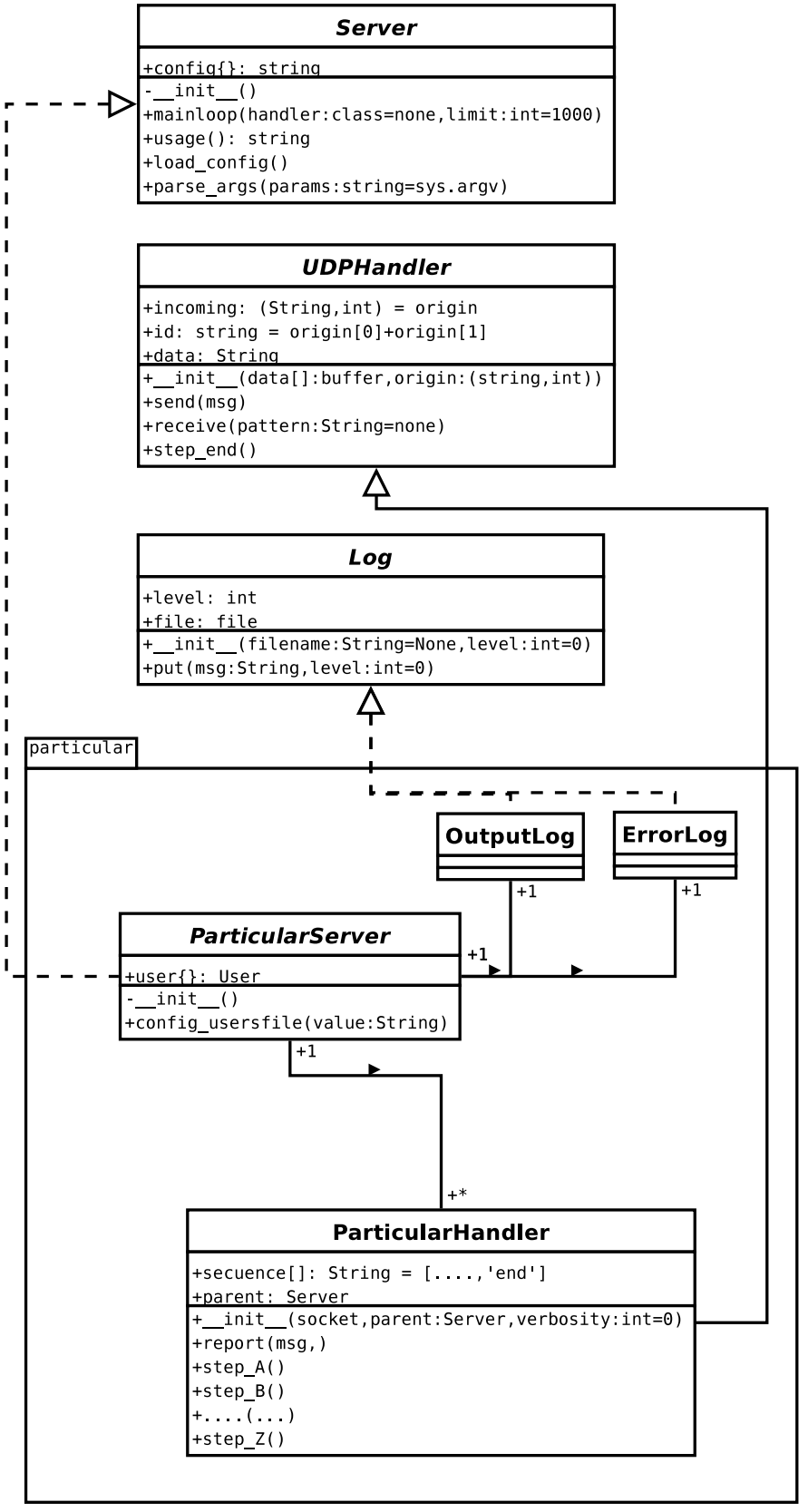

Figura 4.7: Diagrama de Clases de un servicio de ejemplo

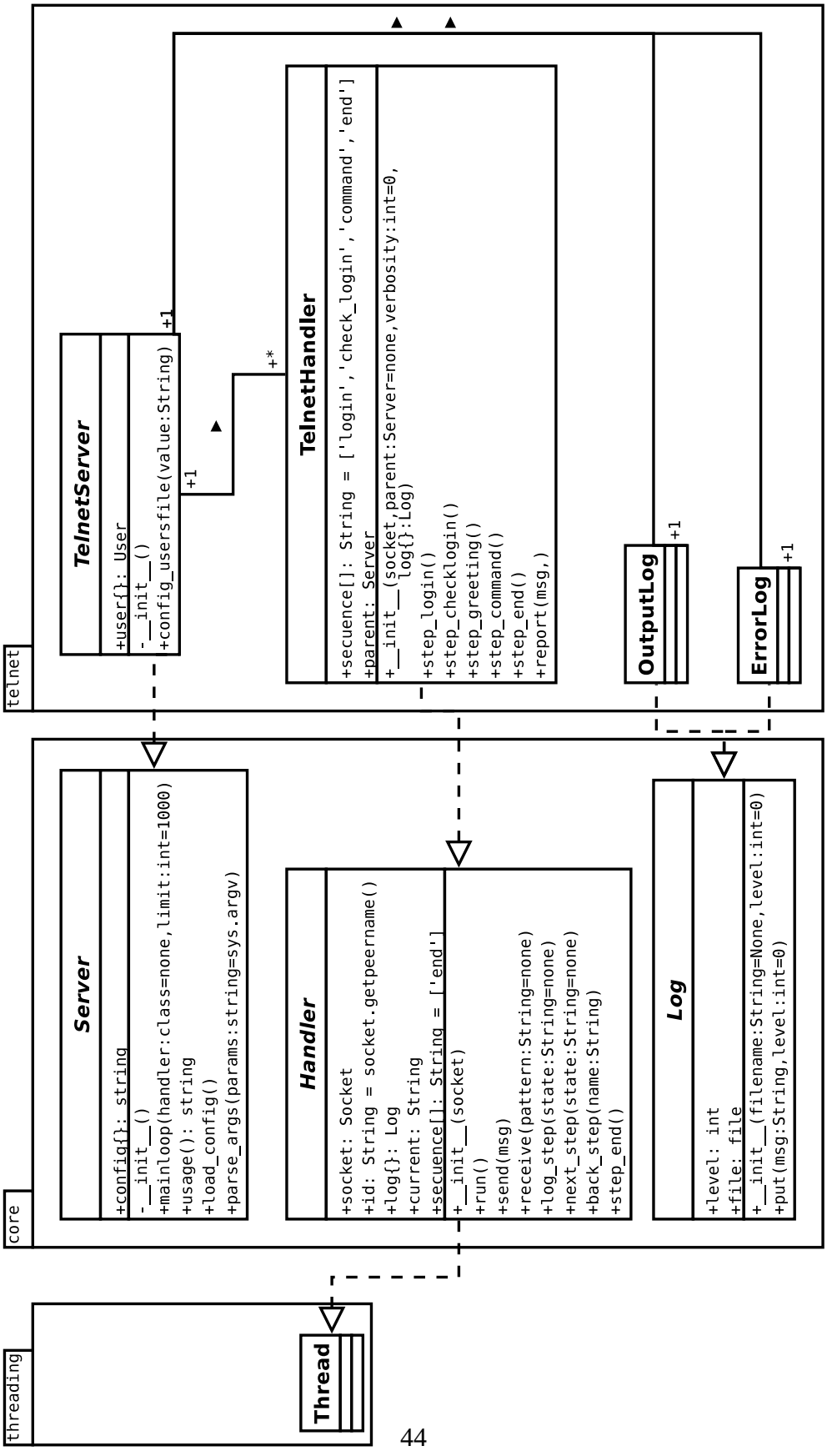

Figura 4.8: Diagrama de Clases de Diseño: Servicio Telnet

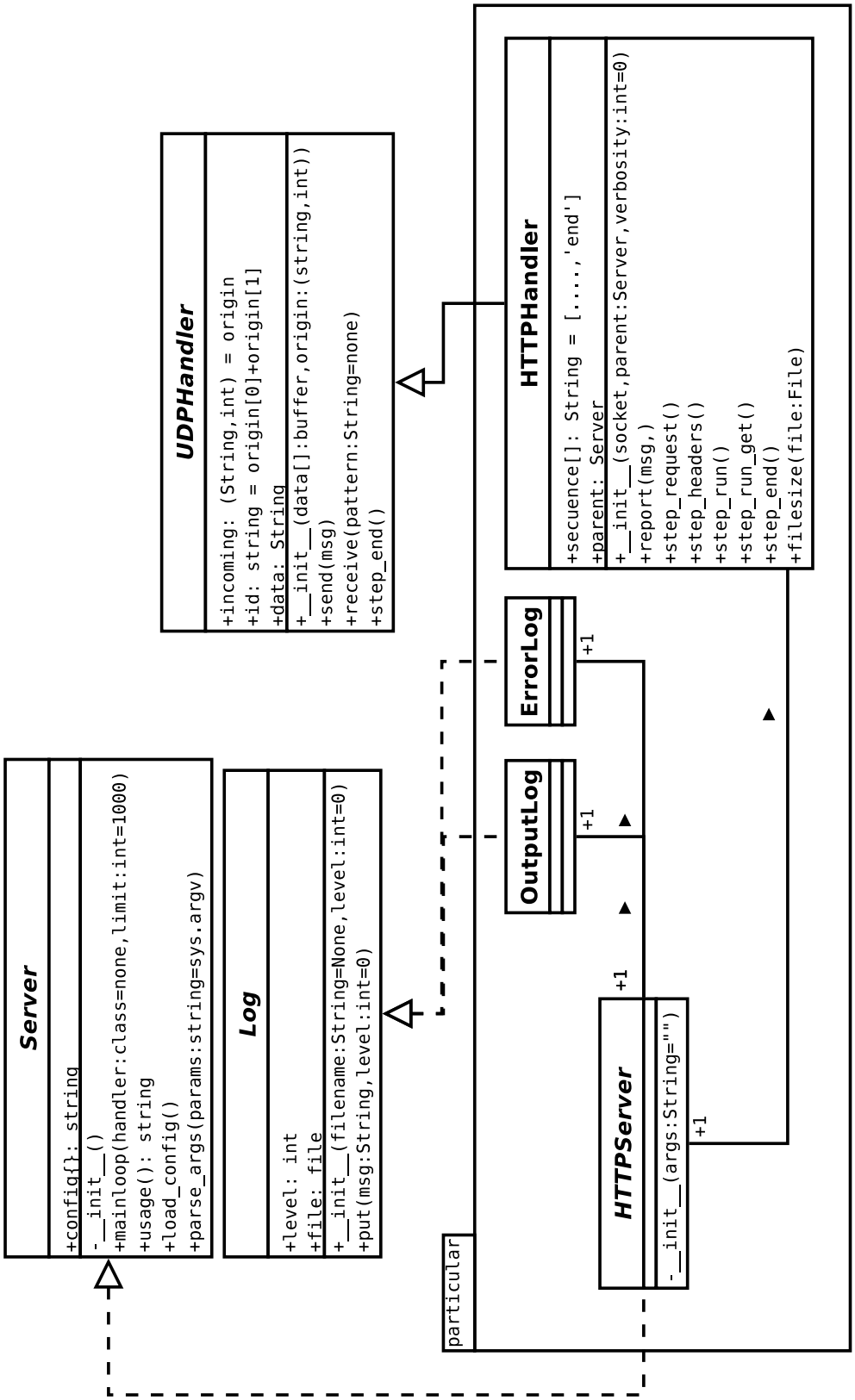

Figura 4.9: Diagrama de Clases de Diseño: Servicio HTTP

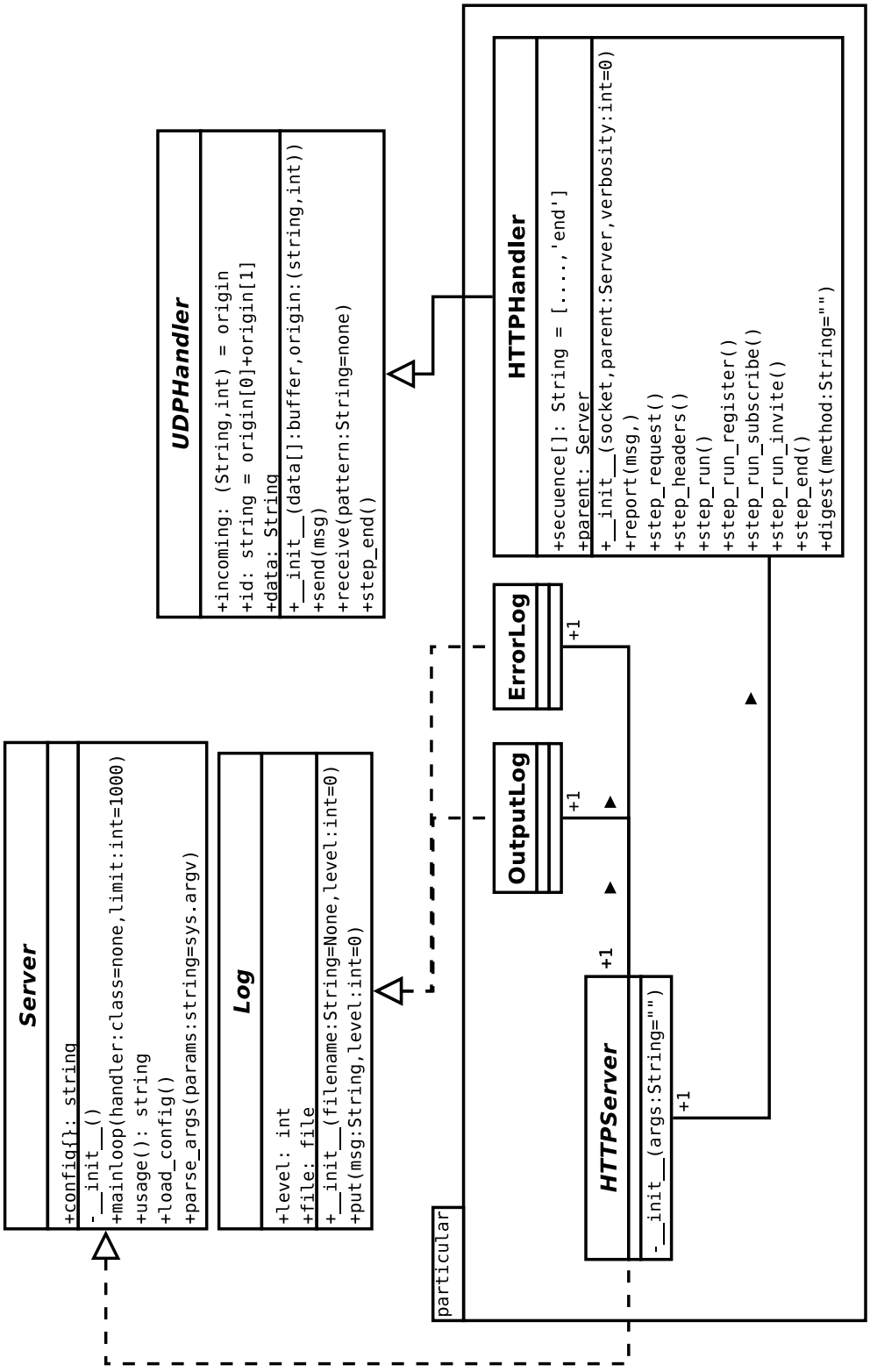

Figura 4.10: Diagrama de Clases de Diseño: Servicio SIP

Capítulo 5

Conclusiones y proyectos futuros

5.1. Conclusiones

A lo largo de la realización del proyecto se han presentado varias dificultades, que se detallan a continuacion:

1. Problemas en la depuración de expresiones regulares: para corregir un fallo durante la entrada de datos, aunque esta ya se hubiera localizado en la expresión regular, su corrección resultaba bastante engorrosa. El proceso para comprobarla consistia en abrir un interprete Python aparte, importar los modulos correspondientes e imitar el mismo procedimiento que realiza el programa, lo que supone escribir 10 líneas de código que irremediablemente se perderán. Para evitar esto sería interesante escribir algúna clase de utilidad que facilite tanto la elaboración de patrones como su comprobación.

2. Problemas con los clientes, ya que muchos de estos no se acomodan correctamente a las especificaciónes del protocolo, salvo en los casos en que se ha podido llegar a una solución elegante realizando pequeñas variaciones. En otros casos aceptar modificaciones sobre los protocolos normalmente ha implicado producir bastante código extra para subsanarlo.

3. Las especificaciónes RFC son en muchos casos dificiles de entender. Para la implementación de los protoclos se ha hecho necesario obtener capturas de operaciones de dichos protocolos para comprender el procedimiento.

4. Aún a pesar de utilizar un importante número de trazas a lo largo de la progra-

mación del servicio muchos errores solo pueden localizarse analizando el tráfico. Por ello disponer de programas como Wireshark `http://www.wireshark.org/` puede resultar de gran utilidad.

5. Sería interesante adoptar una nomenclatura para los estados que derivan de otros estados sin alargar innecesariamente la longitud de las llamadas.

Sin embargo en este proyecto si se han superado obstaculos en un tiempo menor de los esperado:

1. El soporte para Datagram User Protocol se implementó en apenas 3 horas. Las diferencias con la versión que manejaba sockets TCP eran menores de lo esperado y la facilidad de implementación en Python redujo de manera importante el tiempo de implementación.

2. Los módulos que incorpora Python para casi cualquier necesidad imaginable reduce el tiempo de implementación, sin embargo debe tenerse en cuenta que estos módulos deben estar presentes también en los entornos donde vaya a ejecutarse el servicio.

3. Las estructuras de datos primitivas de Python como listas y diccionarios han resultado muy utiles a la hora de almacenar la información de las cabeceras, al igual que las rutinas para tratamiento de cadenas.

5.2. Mejoras

En futuras implementaciónes de BLAS se preveen una serie de mejoras planteadas para aumentar la funcionalidad de la plataforma o solventar sus carencias

5.2.1. Cifrado

Actualmente casi todos los servicios de red se comunican empleando canales seguros. Probablemente imitar cualquier servicio de forma completa requeriría aportar el soporte de distintos métodos criptograficos. Algunos de los algoritmos mas comunes que se pueden localizar en internet son:

1. DES y 3DES: Publicados en 1976 y 1978 respectivamente. Son los algoritmos más veteranos y aun presentes en muchos sistemas aunque hayan perdido robusted con el paso del tiempo.

2. IDEA: Publicado en 1991 para sustituir a DES

3. Blowfish: Publicado en 1993

4. RC5: Publicado por RSA en 1994

Para facilitar su uso por parte del programador y así mantener uno de los principales valores de la plataforma deben diseñarse métodos para que una vez negociada una contraseña común el flujo de datos pueda mantenerse de forma identica al uso que se realizara sobre send() y receive(). Ya que Python permite la sobrecarga de métodos desde la propia clase durante la ejecución implementar una rutina que reemplace los métodos de entrada y salida no revestirá dificultad.

5.2.2. Sistemas de registro opcionales

Además del volcado de la información de registros de operación y errores al sistema de ficheros o a la salida estándar, existen otras posibilidades que pueden extender la funcionalidad de la plataforma:

1. Terminal TCP: Consistente en un servicio que tras conectarse mediante telnet permita revisar los eventos en el sistema.

2. Volcado UDP: se enviarán a una ip y puerto predeterminados la información en texto claro de los eventos del sistema.

De esta manera se debería extender la clase Log, creando nuevas subclases que heredarian de Log, sobrecargando los métodos para actuar de forma transparente al programador.

5.2.3. Establecimiento como servicio del sistema

Ya que Python es multiplataformai, según BLAS madure deberían acompañarsele procedimientos faciles para que los servicios producidos puedan implantarse en el sistema junto a otros.

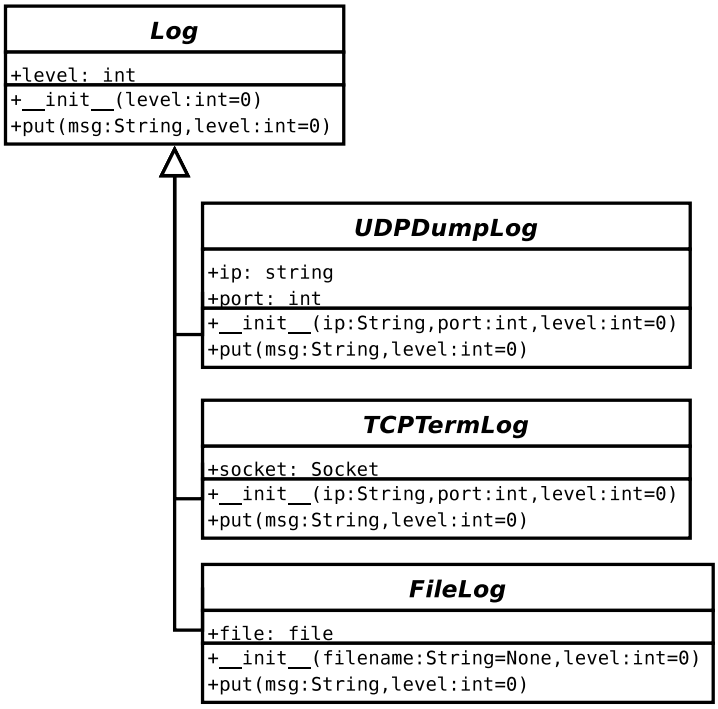

Figura 5.1: Diagrama de diseño. Nuevos registros

Es importante implementar una rutina que permita un empaquetado facil del código y el software asociado (el imterprete Python con el juego de librerias necesario) para un despliegue del sistema lo mas agil y rápido posible.

5.2.4. Construcción de expresiones regulares

La construcción de expresiones regulares que acepten cadenas de acuerdo a un patron consumen tiempo y en ocasiones son el origen de fallos dificiles de diagnosticar. Por este motivo implementar un sistema para construir expresiones regulares de forma rapida y clara debería ser una de las prioridades a la hora de extender el proyecto.

El resultado ideal sería conseguir que no aparezcan expresiones regulares que entorpezcan la lectura del código de atención del servicio.

5.3. Rendimiento y pruebas

Para profundizar en las capacidades de la plataforma para implementar servicios en cierto nivel de producción, deberían realizarse una serie de estudios.

5.3.1. Benchmarking

Comparando con el rendimiento de otros servicios implementados sobre lenguajes más ligeros, como C, debería ofrecer una idea de que la necesidad de recursos que tiene la plataforma en función de la cantidad de clientes y el tipo de exigencia de estos. Conociendo estos requisitos podrán conocerse las posibilidades reales de que el prototipo pueda entrar en algún nivel de producción.

Sería interesante realizar distintos intentos según el nivel de optimización con que se configure la compilación a bytecode del producto.

5.3.2. Seguridad

Para obtener datos fiables sobre la confiabilidad del servicio producido, sobretodo si está expuesto a accesos de terceros, deben realizarse pruebas con intención de desestabilizar el programa o incluso acceder a recursos restringidos a traves de el.

Apéndice A

Licencia GNU Free Documentation License

A.1. Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not

limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

A.2. Applicability and definitions

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

A.3. Verbatim copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

A.4. Copying in quantity

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you

begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

A.5. Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this,

add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

A.6. Combining documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine

any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

A.7. Collections of documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

A.8. Aggregation with independent works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

A.9. Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with trans-

lations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

A.10. Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

A.11. Future revisions of this license

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

A.12. Addendum: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with . . . Texts.” line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Apéndice B

Licencia GNU Public License Versión 3

The technologies of Software development and deployment have changed dramatically since 1991 while the GNU General Public License (“the GPL”) has remained unmodified, at version level 2. This is extraordinary longevity for any widely-employed legal instrument. The durability of the GPL is even more surprising when one takes into account the differences between the free software community at the time of version 2’s release and the situation prevailing in 2005.

Today, the GPL is employed by tens of thousands of software projects around the world and while the Free Software Foundation’s body of GPL licensed works is vital, it consists of no more than a tiny fraction of them. GPL’d software runs on or is embedded in devices ranging from cellphones, PDAs, and home networking appliances to mainframes and supercomputing clusters. Independent software developers around the world, as well as every large corporate IT buyer and seller, and a surprisingly large number of individuals, interact with the GPL. Moreover, free software transcends national boundaries. The GPL’s use is global.

Richard M. Stallman, who founded the free software movement and who was the author of the GNU GPL, released version 2 in 1991 after taking legal advice and collecting developer’s opinions concerning version 1 of the license, which had been in use since 1989. Given that the Free Software Foundation directly controlled the licensing of the GNU project, which comprised the largest then-existing collection of copylefted software assets, no public comment process and no significant interim transition period seemed necessary. The Free Software Foundation immediately relicensed the

components of the GNU Project and in Finland Linus Torvalds adopted GPL Version 2 for his operating system kernel, called Linux.

Many provisions of the GPL could benefit from modification to fit today's circumstances and to reflect what we have learned from experience with version 2. Given the scale of revision it seems proper to approach the work through public discussion in a transparent and accessible manner.

The Free Software Foundation plans to decide the contents of version 3 of the GPL through the fullest possible discussion with the most diverse possible community of drafters and users. A major goal is to identify every issue affecting every user, and to resolve those issues.

For these reasons, the process of GPL revision will be a time of self-examination. Consequently, the process of drafting and adopting changes must be as close to "best practices" as possible, for both lawyers and lay people. Experience has thrown new light on the text of the current GPL. The utility of some provisions has altered over time, while others need to increase their reach in order to protect freedom in the new world of software. Most of the issues caused by this gradual development of the software world can be addressed with minor changes in the text of the GPL.

People who use software, whether they receive copies on CD, or interact with remote installations of the software, have the right to share and improve that software. (Clearly, many, perhaps most, will not modify software; but they share it and desire fixes and improvements. This means they and others must have the right.)

While the GPL is the most popular Free Software License, followed by the LGPL, a significant set of free software is licensed under other terms which are not compatible with version 2 of the GPL. Version 3 of the GPL will provide compatibility with more non-GPL free licenses.

Our primary concern remains, as it has been from the beginning, to give users freedom that they can rely on. As the community around free software has grown larger the issues involved in this creation and protection of freedom have grown more diverse and complex. Therefore, we have consulted, formally and informally, a very broad array of participants in the free software community, from industry, the academy, and the garage. Those conversations have occurred in many countries and several languages, over almost two decades, as the technology of software development and distribution changed around us. We recognize that the best protection of freedom is a growing and vital community of the free and we hope the spread of knowledge inherent in public discussion of version 3 of the GPL drafts will continue to support and nurture this

community.

When a discussion draft of version 3 of the GPL is released, the pace of the revision conversation will change, as a particular proposal becomes the centerpiece. The reversioning of the GPL is a crucial moment in the evolution of the free software community; and the Foundation intends to meet its responsibilities to the makers, distributors, and users of free software. In doing so, we hope to hear all relevant points of view, and to make decisions that fit the many circumstances that arise in the use and development of GPL-covered software.

B.1. Objectives

In drafting a new version of the GPL, the Free Software Foundation have been guided by a few basic principles. These will inform the processes of discussing and promulgating the license as described herein. These principles and their impact on the discussion process are listed below.

B.1.1. A Global License

As a legal document, the GPL licenses copyrighted material for modification and redistribution in every one of the world's systems of copyright law. Ours is an approach that most legal drafters would do anything possible to avoid. Publishers in general do not use worldwide copyright licenses: they try to tailor their licensing arrangements to local legal requirements for each system in which their works are distributed. Publishers rarely license the redistribution of modified or derivative works. When they do, those licenses are tailored to the specific setting.

But free software requires legal arrangements that permit copyrighted works to follow arbitrary trajectories, in both geographic and genetic terms. Indeed, modified versions of free software works are distributed from hand to hand across borders in a pattern that no copyright holder could or should be permitted to trace.

GPL version 2 performed the task of globalization relatively well, because its design was elegantly limited to a minimum set of copyright requirements. Every signatory to the Berne Convention—which means most countries in the world—must offer those principles in their national legislation, in one form or another. But GPL version 2 was constructed only with attention to the details of US law. To the extent possible, without any fundamental changes, version 3 of the GPL should reduce the difficulties of

internationalization. Version 3 should more fully approximate the otherwise unsought ideal of the global copyright license.

B.1.2. Protection of Existing Freedoms

Our cardinal principle is to make no change impeding any of the four basic freedoms for software users that the free software movement enshrined in GPL version 2: to run, study, copy, modify and redistribute software. (It goes without saying that people have the freedom to run a program under the GPL.) These freedoms are as important in version 3 of the GPL as they were in version 2. Honoring the commitment stated in earlier versions of the license, we will preserve these basic rights.

We have judged all changes proposed since the adoption of GPL version 2 against those yardsticks and we will present, in the rationale documents described in section Rationales, reasons tying our changes to those fundamental freedoms. Parties who question changes should recognize when writing their comments that these freedoms remain the cornerstone of the license. We will evaluate all proposed changes with reference to them.

B.1.3. Do No Harm

Unintended consequences can imperil freedom. In approaching GPL version 3 we recognize the enormous expansion in the use of free software since 1991, as well as the many modes of use and distribution that have been invented since. These make the risks of unintended consequences much more severe than when the GPL was last modified.

A large part of the value of the public discussion and issue development described in this document is the identification of unintended consequences worldwide. This is vital to ensuring that version 3 of the GPL is a global license that works as intended in all major legal systems.

Our revision process is intended to make an exhaustive analysis of each considered change in order to explore as much as possible, in as many situations as possible, with as many users and distributors as possible.

B.1.4. Consulting the Community

In short, the essence of the drafting process here described is to make it possible for the Free Software Foundation to decide the contents of the GPL through the fullest possible discussion with the most diverse possible community of drafters and users. Ideally, we would identify every issue affecting every user of the license and resolve these issues with a full consideration of their risks and benefits. In order to accomplish such a large task, the discussion process involves individual community members and Discussion Committees that represent different types of users and distributors.

Each proposed change and the resolution of each issue needs the fullest description of risks and benefits, as laid out in section Feedback.

The Discussion Committees, as described in section DiscussionCommittees, will serve as important centralized points among the different types of user. Among other actions, their role will be to identify issues from the large body of user experience and develop those issues for full presentation and resolution by the Free Software Foundation.

B.2. Process

Periodic releases of the current draft will take place as the license-drafting process progresses. Each draft will represent the most current proposed changes to the GPL. This will take into account all resolved issues, see issueresolution, as well as discussions. We plan to release at least two drafts for public comment. As with all materials and announcements during the discussion process, these drafts will be available from `GPLv3.fsf.org`.

B.2.1. Initial Draft Announcement

The first Discussion Draft of version 3 of the GPL will be released at the the first International Public Conference, January 16-17, 2006, at the Massachusetts Institute of Technology, see Appendix schedule. To accompany the first discussion draft, we will also release a Rationale Document explaining the reasons behind each change in an effort to clarify the nature and necessity of such changes. Similar Rationale documents will accompany each subsequent Discussion Draft of the license as it is released, see Rationales.

B.2.2. Publication of Revised Drafts

At least two discussion drafts of GPL version 3 will be released for public comment. Publication of the second discussion draft will occur after four or five months of discussion, issue identification, and resolution. A third discussion draft may be produced in approximately October 2006, see Appendix schedule, after a second or subsequent iterative process of comment, issue identification, etc. One of these will be the "last call" draft, according to conditions outlined in section lastcall.

B.2.3. Draft Discussion

All consultation with parties outside the Free Software Foundation and the Software Freedom Law Center concerning each discussion draft will be a matter of publicly accessible record available from `GPLv3.fsf.org`. Written deliberations from the Discussion Committees will also be available at `GPLv3.fsf.org`; sound and video recordings of live events and deliberation may become available at a later time. We expect to develop this license through public discussion in a transparent and accessible manner. To that end every effort will be made to make public all documents pertaining to the process.

The GPL revision comment process is a matter of information sharing. Below, in sections DiscussionCommittees and Feedback, we lay out ways that the community as a whole will tell version 3 drafters of issues with the current license. They will speak of ways to increase the positive impact of the license on the world. The Rationale Documents, outlined in section Rationales, and the process for Issue Resolution, section issueresolution, are designed so that, in turn, the drafters at FSF can directly address the community and present the reasoning behind changes.

B.2.4. Last Call Draft

Either the second or third discussion draft will be designated the "last call" draft. This draft will begin a final period of public comment lasting at least 45 days, ending no later than January 15th, 2007. The second discussion draft may be designated the last call draft without further process if there are no major unresolved issues after full discussion of the initial draft.

B.2.5. Promulgation

No later than March 2007, and preferably on January 15, 2007, at the conclusion of the last call process, with all issues resolved, the Free Software Foundation will formally adopt version 3 of the GNU General Public License. At that time, the Free Software Foundation will relicense under GPL version 3 or later all parts of the GNU Project for which the Free Software Foundation is the copyright holder. All parties with authority to relicense programs whose current license terms are "GPL version 2 only" will then be in a position to decide whether to relicense their code. The Free Software Foundation hopes that Discussion Committee members will encourage the relicensing of such works, which is at the discretion of the relevant copyright holders.

B.2.6. Rationales

To make the commentary process easier and to keep the license-drafting process open, each successive draft of the GPL version 3 will be accompanied by a Rationale Document. This document will explain all planned changes in light of the purposes of the license and the freedoms it protects. It will also summarize the public commentary and response relevant to any changed portions of the license. In this, the Rationale Documents will complement the opinion papers issued by the Free Software Foundation regarding resolution of individual issues as identified by the Discussion Committees, see section B.4.2. Rationale Documents will be available through the website `GPLv3.fsf.org`.

B.2.7. Outreach

Transparency does not guarantee widespread distribution; we need to work for that. Much of our effort will therefore be invested in publication and outreach. All information submitted by the public through the revision process will be passed on to the drafters, whether by direct comment submission, Discussion Committee analysis, or transcript of International Conference meetings, and all of it will remain available to the public at `GPLv3.fsf.org`.

Community members who share their experiences with the drafters are encouraged to share them with the rest of the community. People with knowledge of the GPL and the free software movement can educate their fellow community members, as well as people with no previous knowledge of free software. In an effort to extend the process

to the greatest possible number of settings, a team of editors from the FSF will help develop comprehensive issue guides and introductions.

The process of revising the GPL is an opportunity for the community that cares about freedom to educate the rest of the societies they live in. Everyone concerned with the GPL will be asked to examine the license in detail and articulate its impact and possible ways for it to better protect their and others' freedoms. For successful drafting and spread of version 3 of the GPL, this commentary must not only educate the drafters but also the community and public at large.

B.3. Discussion Committees

Dealing with what will probably be extensive public comments is the task of the Discussion Committees, which must structure the flow of comments into issues that can be productively analyzed and whose proposed solutions can be debated. Their work in discovering, developing, and presenting issues is the heart of the version 3 public discussion process.

B.3.1. Composition

Most issues for GPLv3 are global. Therefore we plan to form committees including all categories of relationship to the GPL itself, rather than adopting a regional formation. Thus, elements of the larger GPL community around which Discussion Committees will be formed will include large and small enterprises, both public and private; vendors, commercial and noncommercial redistributors; development projects that use the GPL as a license for their programs; development projects that use other free software licenses, but are invested in the contents of the GPL; and unaffiliated individual developers and people who use software.

Coincident with the publication of this document, the Free Software Foundation will issue invitations to participate in Discussion Committees. These invitations will form nuclei of people. We hope that our invitations will result in Committees that reflect the full breadth of opinion within those sections of the community they functionally represent. But we expect that the Committees themselves will choose to invite additional participants—people whose commitment to the license is undoubted—to add the weights of their opinions to the deliberations. Such invitations, issued after

the invitation of the process, shall be by majority vote of each Committee as already constituted.

B.3.2. Process Commitments

The Committees and their chairs should actively encourage public participation from the sectors of the public they represent.

In addition, Committees are responsible for developing all the opinions and analysis concerning issues they identify from the stream of commentary. As each Committee feels that an issue has been fully discussed among its members, it will be expected to present to the Free Software Foundation its deliberation and analysis of the issue as well as a summary of the public comment that informed its position. Where technically feasible, both the deliberations of the Discussion Committees and the arguments and analysis that they present to the Free Software Foundation will be published at `GPLv3.fsf.org`.

At the conclusion of the public discussion process, we hope to ask members of the Discussion Committees to assist the Free Software Foundation in promulgating the new license; that is, to work with the knowledge gained from their central position within the discussion and revision process to advocate the relicensing of existing GPL programs under version 3 of the GPL.

B.3.3. Organizational Structure

Discussion Committees should be free to choose their own working structure. The Free Software Foundation will provide a template working structure for each committee.

Discussion Committees should operate largely through network-based communication, voice and data, synchronously and asynchronously. They will organize themselves through regular meetings and web-based interactions, encourage public comment and participation, identify and discuss issues, and present those issues and all relevant argument to the Free Software Foundation for ultimate resolution.

B.4. Issue Management and Resolution

From the Foundation's point of view, the revision process is characterized by the presentation and closure of issues revealed in draft discussions.

B.4.1. Forming Issues

The purpose of this public discussion process is to encourage information about the GPL and its role in the expansion and protection of software freedom. This purpose is empty without public commentary. In order to make the most well-informed changes possible to the GPL, we seek commentary from a wide selection of the public. Comments and suggestions are encouraged at `http://GPLv3.fsf.org` as well as in person at any of our International Conferences, see Appendix schedule.

After someone has made a comment, either directly to `GPLv3.fsf.org` or in a discussion at an International Conference, a number of steps will be taken to associate that comment with one or more currently known issues. While comments are the substance of the feedback process, issues are the containers through which they will move.

If the comment or suggestion presents a problem not already identified as an issue, it will be forwarded to the appropriate Discussion Committee where it will join other comments in the identification of a new issue.

For each comment to `GPLv3.fsf.org`, this process has three steps. First, when making the comment the commentator can specify what portion of the license or issue about the license their comment addresses. Once submitted, the comment will be read by an associate member of the Free Software Foundation who will direct it to the appropriate Discussion Committee either for issue identification, if no preexisting issues matches with the comment, or to inform the discussion of the particular issue it addresses.

After making a comment at `GPLv3.fsf.org`, the person involved will be given a comment-identifying number that he or she can use to see towards what issue and Discussion Committee the comment was directed, as well as other comments on the issue and the documents relevant to its discussion (transcripts; Discussion Committee analysis, see B.3; Draft Rationale documents, see B.2.6; FSF Opinion documents, see B.4.2, etc.).

B.4.2. Issue Resolution

Each issue identified in the course of public participation can be resolved in one of four ways: by modification of the license draft, by alteration of descriptive material, by advice concerning the use of the license, or an issue may not require any change. Discussion Committees will characterize issues as Major or Minor. Major issues will be placed on the agenda of all other Discussion Committees and, until resolved, may be placed on the agenda for successive International meetings. All issues unresolved at the end of each drafting stage will be carried over for discussion and resolution during the next discussion stage. All issues not resolved before the issuance of the last discussion draft will be finally determined by the Free Software Foundation at the close of the last call period. All Major issues resolved by the Foundation will be described by a written opinion, publicly available, at `GPLv3.fsf.org`.

B.5. Other Concerns

B.5.1. LGPL

The Free Software Foundation may present drafts of LGPL along with drafts of GPL subsequent to the first discussion draft of GPLv3. Such drafts would also be subject to public comment and issue resolution.

B.5.2. Support of the Revision Process

The revision process is financially supported by donors to the Free Software Foundation, the Free Software Foundation's associate members, and a grant from Stichting NLnet. Logistical, legal, and technical support for this process is also provided by the Software Freedom Law Center, acting as the Free Software Foundation's outside counsel. The SFLC is supported by a variety of donors including vendors and those who use free software.

Aside from resources contributed by the Free Software Foundation and the Software Freedom Law Center, this process will be supported, only to the extent of logistical provision for International Meetings, by industry organizations hosting the events. Outside logistical support is accepted only in order to ensure that participants around the world will have the maximum possible level of access to the discussion process

of version 3 of the GPL. All participants in the discussion process can therefore be assured of equal treatment for their interests and concerns.

B.5.3. Public Statements

During this process the Free Software Foundation will make public statements concerning the process, deadlines, issues, comments, and drafts. Such public statements will be made through announcements at `GPLv3.fsf.org`, and by messages to mailing lists to which parties can subscribe. The Free Software Foundation and SFLC will not hold confidential communications with others concerning version 3 of the GPL. Public commentary on these announcements, as with all comments relating to the GPL version 3 discussion process, should be routed through the GPL comment system described in section B.4.1. Interested members of the Press should see B.5.4 below.

B.5.4. Press Contact

Press contacts may occur and statements may be issued to the press through the Free Software Foundation and the Software Freedom Law Center. All such statements will be published at `GPLv3.fsf.org` or referenced there. Press interested in covering this process should follow the contact information available at the website or write to `press@GPLv3.fsf.org`.

Bibliografía

[1] Neal Stephenson: In the Beginning was the Command Line Editorial Harper Perennial, 1999.

[2] Python Software Foundation Web-Site: `http://www.Python.org`

[3] Especificación del Protocolo de Control de Transmisión (TCP): `http://www.rfc-es.org/rfc/rfc0793-es.txt`

[4] Especificación del Protocolo de Datagramas de Usuario (UDP): `http://www.rfc-es.org/rfc/rfc0768-es.txt`

[5] Especificación del Protocolo Telnet: `http://www.rfc-es.org/rfc/rfc0854-es.txt`

[6] Especificación del Protocolo de Transferencia de HiperTexto (HTTP): `http://www.ietf.org/rfc/rfc2616.txt`

[7] Especificación del Protocolo de Inicialización del Servicio (SIP): `http://www.rfc-editor.org/rfc/rfc3261.txt`