Índice general

BLAS: Base Layer for Application Services

Eduardo Orive Vinuesa

17 de agosto de 2008

Capítulo 1

Introducción

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

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 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 funcion era comunicar con el ordenador central pudiendo utilizar enlaces dedicados o conmutados (lineas 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 informaticos.

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 comunicacion por conmutacion convencionales y los servicios empiezan a emplear 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 sigue siendo los mejores ambientes comunitarios.

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

1.2. Motivación

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 inversion en tiempo y no se obtienen resultados tangibles hasta que se ha implementado una buena parte del código. También 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 caracteristicas particulares de su protocolo. Basicamente cualquier protocolo puede entenderse como un automata más o menos complejo, siguiendo este modo de diseõ puede simplificarse de manera considerable el trabajo del programador.

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.

Aunque hoy día se pueden encontrar cientos de implementaciones de protocolos sobre multitud de tecnologías diferentes existe todavía la necesidad de realizar implementaciones de servicios de protocolos antiguos o nuevos.

1.2.1. Docencia

Existen numerosos proyectos academicos 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.

1.2.2. Microdispositivos

La mayoría de los dispositivos dotados de un sistema operativo mínimo disponen de una modesta pila TCP/IP ya que practicamente todos estos dispositivos se diseñan incluyendo algún tipo de hardware de interconexión. Sin embargo por razones de coste economico o rendimiento energetico se ven afectados por una leve capacidad de memoria y/o potencia de cálculo que hacen que la opcion de conectarlos a 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.

1.2.3. 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 que acciones realiza una vez dentro.

La capacidad de programar servicios simulados agilmente 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 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 TCP que consumen mucho tiempo y que normalmente su implementación resulta bastante repetitiva:

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

Capítulo 3

Estado del arte

3.1. Elección del Lenguaje

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

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

2. Reflexividad: Las clases en pytohn tienen conocimiento de sus propios metodos y atributos.

3. 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 lineas la implementación de un protocolo completo. Al tratarse de un lenguaje en que la identación es obligatoria y la sintaxis muy clara normalmente su codigo fuente esta bien estructurado y resulta facil de leer incluso para quienes apenas conozcan Python.

3.2. 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 versiones 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, Python esta implementado en C. Pero existen otras: Basada en Java una implementacion denominada Jython puede utilizarse para trabajar con codigo Java nativamente. Iron Python, una version 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 tambien intectuar perfectamente con otros lenguajes a través de sus correspondientes modulos.

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

3.2.1. Evolución

Primera publicación

En 1991 van Rossun puclido el codigo (con la versión 0.9.0) en alt.sources. Ya en este primer estado de desarrollo habia clases con herencias, manejo de expeciones, funciones y los tipos de datos basicos de listas, diccionarios, cadenas y demás. También en esta publicación inicial habian un modulo 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 clausula “else”. En 1994 arranca comp.lang.python, la primera lista de discussion de python, marcando un hito en el crecimiento de la base

de usuarios de Python.

Version 1.0

Python llego a la version 1.0 en Enero de 1994. Las principales nuevas funcionalidades incluidas en esta publicación fueron las utilidades de programación funcional lambda, map, filter y reduce. Van Rossum expreso que “Python a conseguido lambda, reduce, filter y map por cortesia (en mi opinión) de algun hacker de Lisp que las echabada de menos y envio parches operativos”.

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

Por la version 1.4, Python habia conseguido varias nuevas funciones. Es de destacar que entre estas son los parametros por palabra clave de Modula3 (Que son tambien similares a los parametros por palabra clave de Lisp) y el soporte integrado para números complejos. Tambin incluye una forma basica de ocultamiento de datos por manipulacion de nombre, aunque podia ser facilmente obviado.

Durante la estancia de var Rossum en el CNRI, lanzo la iniciativa Programacion de Ordenadores para Todos (CP4E), que pretendia hacer la programacion mas accesible a mas gente, con un conocimiento basico en lenguajes de programacion, de forma similar a los conocimientos basicos de literatura inglesa y matematicas que se exigen a cualquier empleado. Python sirvio con un papel central en esto: devido a su enfoque en una sintaxis clara, ya podia considerarse adecuado, y los objetivos del CP4E ya apuntaba parecidos con su predecesor ABC. El proyecto fue fundado por DARPA. En 2007 el proyecto parece inactivo y aunque Python pretende ser de facil aprendizaje y no demasiado arcano en su sintaxis y semantica, llegar a ser comprensible para los no programadores no es una intencion activa.

Version 2.0

Python 2.0 introdujo la comprensión de listas, una funcion prestada de los lenguajes de programacion 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 puntuacion y la preferencia de Python por las palabras clave alfabeticas. Python 2.0 tambien introdujo un sistema de recoleccion de basura capaz de tratar referencias ciclicas.

Python 2.1 era bastante parecido a Python 1.6.1, al igual que Python 2.0. La licencia cambio su nombre a Python Software Foundation License. Todo el codigo, documentacion y especificaciones añadidas, desde la alpha de Python 2.1, es propiedad de la Python Software Foundation, una organizacion sin animo de lucro formada en 2001, modelada siguiendo el ejemplo de la Apache Software Foundation. La publicación incluyó un cambio en la especificacion 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 unificacion de los tipos de Python (tipos escritos en C) y clases (tipos escritos en Python) en una unica jerarquía. Esta unificacion hizo al modelo de objetos de python un modelo orientado pura y consistentemente a objetos. También se añadieron generadores inspiradores por el lenguajes Icon.

Version 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.

Disponibilidad

En muchos sistemas operativos Python es un componente estandar, se incluye en la mayoria 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.

3.3. Protocolos

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

codigo 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 estos sean lo más simple y claro posible se ahorrará mucho tiempo y complejidad en la producción del codigo.

Es dificil generalizar sobre protocolos ya que estos varian ampliamente en proposito y en complejidad. La mayoria de los protocolos de red poseen alguna 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 caracteristicas de la conexión.
4. Como indicar el comienzo y el final de un mensaje.
5. Formato de los mensajes.
6. Como actuar ante mensajes corruptos o con un formato incorrecto (corrección de errores)
7. Como detectar una perdida inesperada de la conexion y como comportarse ante esta.
8. Finalizar la sesión o conexión.

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

3.3.1. Telnet

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

TELNET (TELecommunication NETwork) es un protocolo de red utilizado en conexiones de internet o en redes de area local (LAN). Fue desarrollado en 1969 y especificado en el RFC 15 y despues estandarizado en el IETF STD 8, uno de los primeros estandares de internet.

El termino telnet tambien se refiere al software que implementa a la parte cliente del protocolo. Los clientes TELNET estan disponibles en prácticamente cualquier plataforma. La mayoria de los equipos de red que dispongan de una pila TCP/IP tienen soporte para algun tipo de servidor TELNET para permitir su configuracion remota (incluyendo a los basados en Windows NT). Debido a los problemas de privacidad con TELNET se ha reemplazado ampliamente por SSH (Secure SHell).

El termino “hacer un telnet” es la forma de llamar a establecer una conexión o utlizar TELNET u otras conexiones 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 com 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 conexion, el usuario podrá ingresar con sus datos de acceso y ejecutar comandos en el sistema remoto.

En algunos sistemas el cliente se puede utilizar para realizar sesiones TCP en bruto. Comunmente se cree que una sesion telnet que no incluye un IAC (el caracter 255) tiene la misma utilidad. Esta 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.

3.3.2. 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 programacion 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 informacion a través de Internet. Su utilización par recuperar documentos de texto vinculados (hipertexto) a llevado al establecimiento de la World Wide Web.

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

HTTP es un estandar petición/respuesta entre un cliente y un servidor. El cliente será el usuario final mientra que el servidor es el sitio web (web site). El cliente, realizando una peticion 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 imagenes, 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 GTTP no esta limitado a su utilización sobre TCP/IP y sus capas, más aun es la aplicación mas extendida en Internet. Ciertamente HTTP puede implementarte sobre sobre otros protocolos en Interneto en otras redes. HTTP solo 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 peticion. Se establece una conexion TCP a un puerto particular en un servidor (el puerto 80 por defecto). el servidor HTTP escuchando en ese puerto esperara a que el cliente emita el mensaje de petición. Una vez recibida la petición el servidor reponde con una linea 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 informaciones.

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 pagina web y a que TCP provee control de transmisión, envia 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).

3.3.3. SIP: Session Initiation Protocol

Es un protocolo desarrollado por el IETF MMUSIC Working Group en la busqueda de un estandar 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 instantanea, 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 direcciones 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 diseñador por Henning Schulzrinne (Universidad de Columbia) y Mark Handley (UCL) en 1996. La ultima version de la especificacion es el 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 alli habian instalados y recibir posteriormente sus resultados. A diferencia de hoy aquellos sistemas funcionaban directamente sobre enlaces dedicados y cada uno de estos teletipos disponia de una conexión permanente.

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.

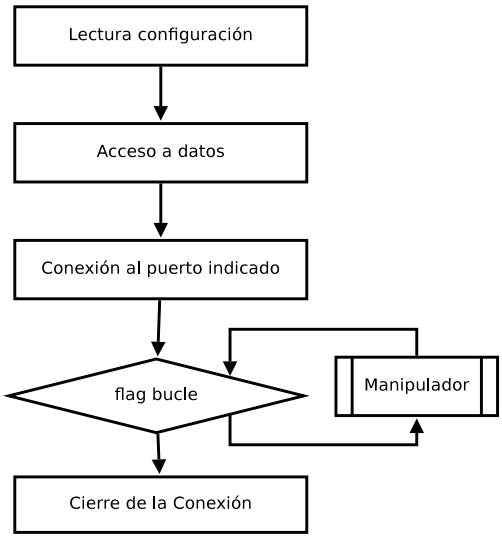

Figura 4.1: niveles OSI

En el contexto de los sistemas informáticos un protocolo es un conjunto de reglas predefinidas, cuyo proposito es estandarizar 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 contacto, posiblemente otros protocolos.

La forma clásica de programar un servicio consiste en un blucle de escucha que inicia subprocesos para cada cone-

xión entrante y que finaliza cuando esta acaba o se produce un error. Se presentan dos entidades en este caso, la del servicio que incluiria las rutinas para iniciar la escucha y el acceso a los datos y la del manipulador, que atiende la conexion 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 linea de comandos
2. La lectura de parámetros desde el archivo de configuracion
3. El registro de actividad en el(los archivo(s) pertinentes
4. Preparar el acceso a los datos necesarios, ficheros o bases de datos

Parámetros

Existen parametros que son comunes practicamente a cualquier servicio por ejemplo:

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

O parametros particulares para segun que tipos de servicio:

1. Acceso al medio de datos
2. Parametros de configuración concretos

Debe tenerse en cuenta que los parametros en el fichero de configuracion tienen menor prevalencia que los indicados por linea de comandos, salvo el que especifique precisamente que 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 implementacion 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 parametros se pueden utilizar comodines para ampliar la versatilidad:

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

Locuacidad

Para que puedan establecerse mensajes de depuración a distintos niveles debe adoptarse un sistema de categorias de mensajes, 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 minimo, 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 mayoria 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 intencion 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 configuracion o a los medios salida por pantalla dedicados a tal fin).

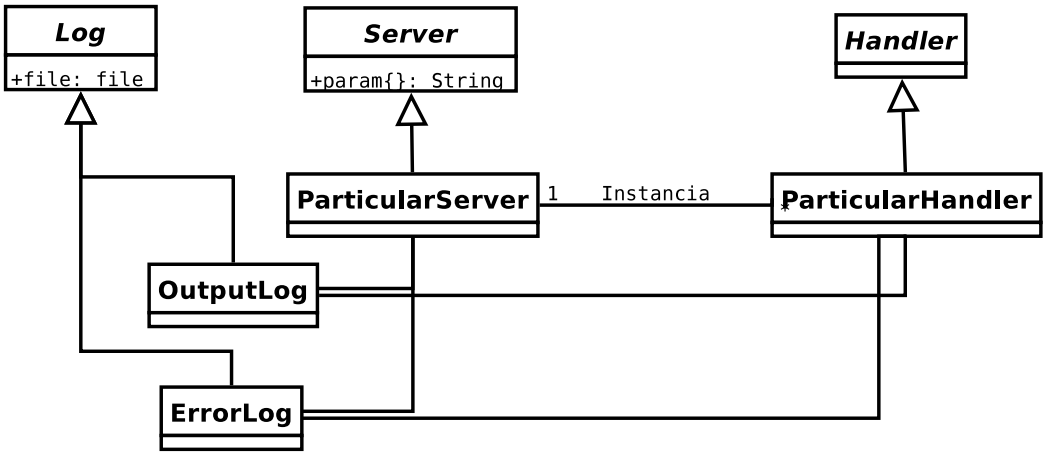

Figura 4.2: Diagrama de Clases de Análisis

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 informacion entre los sistemas, la que permite abstraer al programa servidor como un automata sensible al contexto. Cada etapa del manipulador equivale al estado del automata 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 especificaciones 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 reimplementaciones 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 deberia contener un numero limitado de procedimientos, para reducir la complejidad.

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

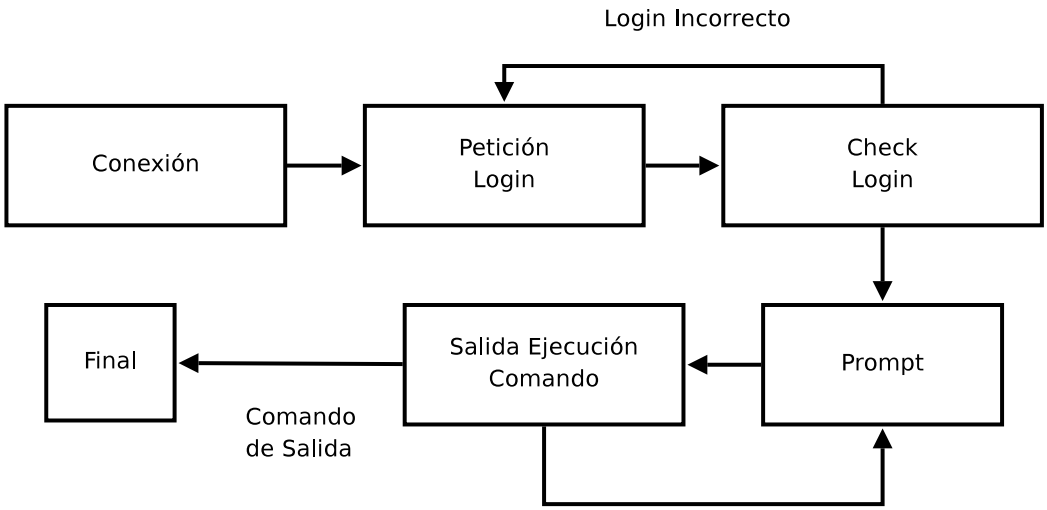

Figura 4.3: Flujo de desarrollo del protocolo Telnet

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

1. Se establece la conexión con el cliente
2. Se presenta el dialogo de peticion 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

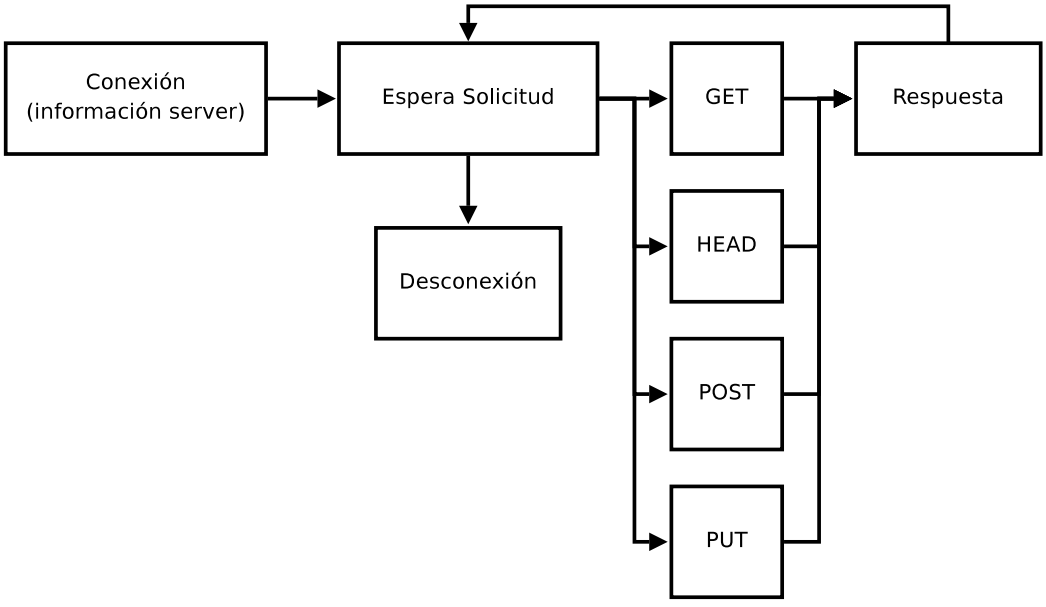

Figura 4.4: Flujo de desarrollo del protocolo HTTP

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

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 atencion 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 TCP. Estas tareas son:

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.

4.2. Diseño

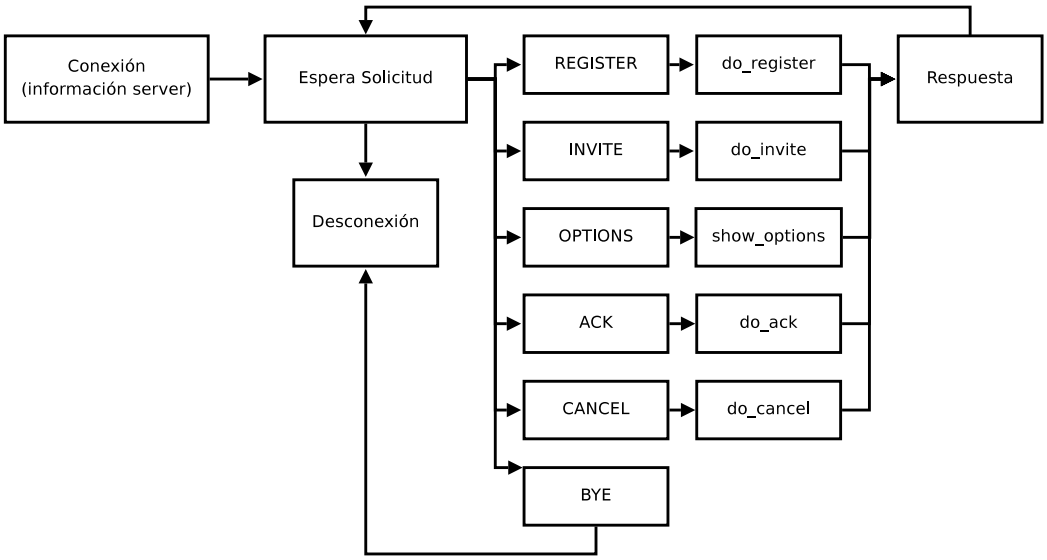

Figura 4.5: Flujo de desarrollo del protocolo SIP

Capítulo 5

Previsiones y proyectos futuros