Universidad
del Cauca

HERRAMIENTA WEB PARA LA GESTIÓN DE SERVICIOS DE INTERNET EN UNA LAN Y SU APLICACIÓN A LA RED DE DATOS DE LA UNIVERSIDAD DEL CAUCA

ANEXO B – DESCRIPCIÓN DETALLADA DE LOS SERVICIOS DE INTERNET GESTIONADOS

Juan Manuel García V.

Héctor Fabio Polanco C.

Director:
Ing. Esp. Guefry Agredo Méndez.

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telecomunicaciones
Grupo de I+D en Nuevas Tecnologías en Telecomunicaciones
Línea Gestión Integrada de Redes, Servicios y Arquitecturas de Telecomunicaciones

Popayán, Noviembre de 2004

TABLA DE CONTENIDOS

INDICE DE FIGURAS

1. SERVICIO HTTP

1.1. Introducción

El Protocolo de Transferencia de Hiper-texto *(HTTP - Hypertext Transfer Protocol)* es un protocolo del nivel de aplicación para sistemas colaborativos de información distribuida e hiper-textual. La primera versión de HTTP (HTTP/0.9) era un protocolo simple para transferencia básica a través de Internet. HTTP/1.0 definido en el RFC 1945, mejoró el protocolo permitiendo mensajes en formato MIME.

1.2. Funcionamiento general

HTTP es un protocolo de petición/respuesta. Un cliente envía una petición a un servidor en forma de un método de petición, un Identificador Uniforme de Recurso y una versión del protocolo seguidos de un mensaje tipo MIME que contiene modificadores de la petición, información del cliente y posible contenido del mensaje sobre una conexión con el servidor.

El servidor responde con una línea de estado, incluyendo la versión del protocolo del mensaje y un código de éxito o error seguido de un mensaje tipo MIME con información del servidor, meta-información de la entidad y posible contenido. La mayoría de comunicaciones HTTP son iniciadas por el agente del usuario y consisten de una petición a ser aplicada a un recurso de algún tipo en el servidor. En el caso más simple, esto se realiza sobre una conexión (V) entre el agente del usuario (UA) y el servidor origen (S)

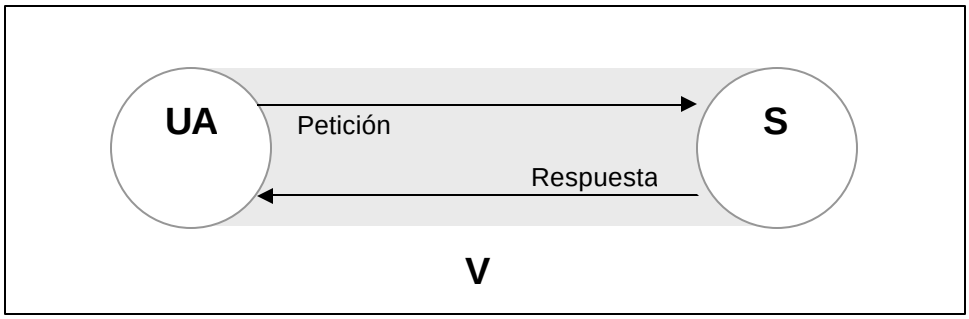

Figura 1. Comunicación entre el agente del usuario (UA) y el servidor (S)

1.3. Parametros del protocolo

1.3.1. URI (Identificador Uniforme de Recursos)

Los Identificadores Uniformes de Recursos *(URI - Uniform Resource Identifiers)* se conocen por diferentes nombres: Direcciones WWW, Identificadores Universales de Recursos (URI), Ubicadores Uniformes de de Recursos *(URL - Uniform Resource Locators)*, por ejemplo. Para HTTP, las URI's son simplemente cadenas de texto formateadas que identifican un recurso por su ubicación u otra característica.

1.3.1.1. Sintaxis general

URI's en HTTP pueden representarse de forma absoluta o relativa a un URI base conocido, dependiendo del contexto de su uso. Las dos formas se diferencian por el hecho de que los URI's absolutos siempre comienzan con un nombre de esquema seguido de un punto y coma. El protocolo HTTP no pone ningún límite al tamaño de un URI. Los servidores deben estar en capacidad de manipular el URI de cualquier recurso sin importar su ancho.

1.3.1.2. URL de HTTP

El esquema HTTP se usa para ubicar recursos de red sobre el protocolo HTTP. La siguiente es la sintaxis de las URL's de HTTP:

http_URL = "http:" "//" host [":" port] [abs_path ["?" query]]

SI el puerto no es dado, se asume el valor 80. Las semánticas indican que el recurso identificado está ubicado en el servidor que está escuchando conexiones TCP en el puerto indicado, y la Petición URI para el recurso es el parámetro "abs_path". El uso de direcciones IP en las URL's debería ser evitado siempre que sea posible. Si éste no está presente en la URL, debe darse como un "/" cuando se usa como Petición URI para un recurso.

1.3.1.3. Comparaciones de URI's

Cuando se comparan dos URI's para decidir si son iguales, el cliente debe realizar un comparación octeto-a-octeto y hacer distinción entre mayúsculas y minúsculas, con las siguientes excepciones:

- Un puerto vació o no escrito es equivalente al puerto por defecto para esa Referencia URI.
- Las comparaciones de nombres de equipos no deben distinguir entre mayúsculas y minúsculas.
- La compasión de nombres de esquemas no debe distinguir entre mayúsculas y minúsculas.
- Un valor vació de 'abs_path' es equivalente a "/"

Por ejemplo, las siguientes URI's son equivalentes:

- http://abc.com:80/~smith/home.html
- http://ABC.com/%7Esmith/home.html
- http://ABC.com:/%7esmith/home.html

1.4. Petición

Un mensaje de petición de un cliente a un servidor incluye, dentro de la primera línea, el método a ser aplicado al recurso, el identificador del recurso y la versión del protocolo en uso.

1.4.1. Línea de petición

La línea de petición comienza con un método, seguido de la Petición URI, la versión de protocolo y un retorno de carro con alimentación de línea al final *(CRLF – Carriage Return Line Feed)*. Los elementos son separados por espacios (SP). Sólo se permiten retornos de tipo Retorno de Carro *(CR – Carriage Return)* ó Alimentación de Línea *(LF – Line Feed)* excepto en el retorno CRLF de la secuencia final.

1.4.2. Métodos

El método indica la acción a realizar sobre el recurso identificado por la Petición URI. Los métodos no distinguen mayúsculas o minúsculas. Un método se dice que es seguro si no provocan ninguna otra acción que no sea la de devolver algo (no produce efectos laterales). Estos métodos son el método GET y el método HEAD. Para realizar acciones inseguras (las que afectan a otras acciones) se pueden usar los métodos POST, PUT y DELETE. Aunque esto está definido así, no se puede asegurar que un método seguro no produzca efectos laterales, porque depende de la implementación del servidor.

Un método es idempotente si los efectos laterales para N peticiones son los mismos que para una sola petición. Los métodos idempotentes son los métodos GET, HEAD, PUT y DELETE.

1.4.2.1. Método OPTIONS

Este método representa una petición de información sobre las opciones de comunicación disponibles en la cadena petición-respuesta identificada por el URI de la petición. Esto permite al cliente conocer las opciones y requisitos asociados con un recurso o las capacidades del servidor. La respuesta sólo debe incluir información sobre las opciones de comunicación.

Si el URI es ``*'', entonces la petición se aplica al servidor como un conjunto. Es decir, contesta características opcionales definidas por el servidor, extensiones del protocolo.

1.4.2.2. Método GET

El método GET requiere la devolución de información al cliente identificada por el URI. Si el URI se refiere a un proceso que produce información, se devuelve la información y no la fuente del proceso.

El método GET pasa a ser un GET condicional si la petición incluye las cabeceras *If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match* o *If-Range*. Estas cabeceras hacen que el contenido de la respuesta se transmita sólo si se cumplen unas condiciones determinadas por esas cabeceras. Esto se hico para reducir el tráfico en las redes.

También hay un método GET parcial, con el que se envía sólo parte del contenido del recurso requerido. Esto ocurre cuando la petición tiene una cabecera *"Range".* Al igual que el método GET condicional, el método GET parcial se creó para reducir el tráfico en la red.

1.4.2.3. Método HEAD

El método HEAD es igual que el método GET, salvo que el servidor no tiene que devolver el contenido, sólo las cabeceras. Estas cabeceras que se devuelven en el método HEAD deberían ser las mismas que las que se devolverían si fuese una petición GET.

Este método se puede usar para obtener información sobre el contenido que se va a devolver en respuesta a la petición. Se suele usar también para chequear la validez de links, accesibilidad y modificaciones recientes.

1.4.2.4. Método POST

El método POST se usa para hacer peticiones en las que el servidor destino acepta el contenido de la petición como un nuevo subordinado del recurso pedido. El método POST se creó para cubrir funciones como la de enviar un mensaje a grupos de usuarios, dar un bloque de datos como resultado de un formulario a un proceso de datos, añadir nuevos datos a una base de datos.

La función llevada a cabo por el método POST está determinada por el servidor y suele depender dell URI de la petición. El resultado de la acción realizada por el método POST puede ser un recurso que no sea identificable mediante una URI.

1.4.2.5. Método PUT

El método PUT permite guardar el contenido de la petición en el servidor bajo el URI de la petición. Si esta URI ya existe, entonces el servidor considera que esta petición proporciona una versión actualizada del recurso. Si el URI indicado no existe y es válido para definir un nuevo recurso, el servidor puede crear el recurso con esa URI. Si se crea un nuevo recurso, debe responder con un código 201 (creado), si se modifica se contesta con un código 200 (OK) o 204 (sin contenido). En caso de que no se pueda crear el recurso se devuelve un mensaje con el código de error apropiado.

La principal diferencia entre POST y PUT se encuentra en el significado del URI. En el caso del método POST, el URI identifica el recurso que va a manejar en contenido, mientras que en el PUT identifica el contenido. Un recurso puede tener distintas URI.

1.4.2.6. Método DELETE

Este método se usa para que el servidor borre el recurso indicado por el URI de la petición. No se garantiza al cliente que la operación se lleve a cabo aunque la respuesta sea satisfactoria.

1.4.2.7. Método TRACE

Este método se usa para saber si existe el receptor del mensaje y usar la información para hacer un diagnóstico. En las cabeceras el campo *"Via"* sirve para obtener la ruta que sigue el mensaje. Mediante el campo *"Max-Forwards"* se limita el número de pasos intermedios que puede tomar. La petición con el método TRACE no tiene contenido.

1.4.3. Petición URI

La Petición URI identifica el recurso al cual se va a aplicar la petición

Request-URI = "*" | absoluteURI | abs_path | authority

Las cuatro opciones para las Peticiones URI dependen de la naturaleza de la petición. El asterisco (*) significa que la petición no aplica a un recurso en particular, sino al servidor, y solo se permite cuando el método usado no necesariamente se aplica a un recurso. Por ejemplo:

OPTIONS * HTTP/1.1

La forma más común de Petición URI es aquella usada para identificar un recurso en un servidor. En este caso, la ruta absoluta del URI debe ser transmitida como la Petición URI, y la ubicación de red del URI debe ser transmitida en un campo del encabezado del equipo. Por ejemplo, un cliente que desee traerse el anterior recurso directamente del servidor origen crearía una conexión TCP al puerto 80 del equipo 'www.w3.org' y enviar las líneas:

GET /pub/WWW/TheProject.html HTTP/1.1
Host: www.w3.org

Seguidas de el resto de la Petición. Nótese que la ruta absoluta no puede estar vacía; si no está presente en el URI original, debe darse como un "/" (La raíz del servidor).

1.5. Respuesta

Después de recibir e interpretar un mensaje de petición, un servidor responde con un mensaje de respuesta HTTP

1.5.1. Línea de estado

La primera línea de un mensaje de respuesta es la Línea de Estado, que consiste de la versión del protocolo seguida de un código de estado numérico y su frase asociada, cada elemento separado por un espacio (SP). No se permiten caracteres CR o LF excepto en la secuencia CRLF final.

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

1.5.2. Código de estado y frase de explicación

En el protocolo HTTP, toda respuesta de un servidor Web debe incluir un código de estado que indica el resultado de la solicitud HTTP del cliente (navegador).

Los códigos de estado HTTP están compuestos por un código de tres dígitos seguidos por una descripción corta. La siguiente es la clasificación de los códigos de estado HTTP 1.1:

- **100 - 199 Informativo**
- **200 - 299 Exitoso**
- **300 - 399 Redirección**
- **400 - 499 Error del Cliente**
- **500 - 599 Error del Servidor**

1.5.2.1. 100 - 199 Informativo

- **100 Continue**
  El cliente (navegador) debe seguir con la solicitud.
- **101 Switching Protocols**
  El servidor entiende y desea cumplir con la solicitud del cliente.

1.5.2.2. 200 - 299 Exitoso

Un código de estado 2xx significa que la solicitud del cliente fue recibida, entendida y aceptada con éxito. Salvo el código "204 No Content".

1.5.2.3. 300 - 399 Redirección

Una respuesta 3xx del servidor indica que el recurso solicitado debe obtenerse en una URL diferente.

1.5.2.4. 400 - 499 Error del cliente

Con este tipo de códigos, el servidor indica que piensa que el cliente se equivocó, como por ejemplo solicitar una página que no existe en el servidor o tratar de obtener acceso a un área protegida con contraseña sin presentar las credenciales correctas.

- **400 Bad Request**

  El servidor Web no pudo entender la solicitud debido a errores de sintaxis.

- **401 Unauthorized**

  Se requiere la autenticación del usuario.

- **402 Payment Required**

  Código reservado para uso futuro.

- **403 Forbidden**

  El servidor entendió la solicitud pero no la va a atender. Un caso típico de este error sería cuando la URL solicitada corresponde a un directorio y no a un archivo, y el servidor no permite listar directorios.

- **404 Not Found**

  El servidor no ha encontrado el recurso correspondiente a la URL solicitada. Este error puede deberse a un error de ortografía o de sintaxis en la URL o que se trate de un recurso que ya no existe en el servidor.

  Un error común que produce un código 404 en los navegadores es incluir la dirección de un archivo local ú omitir el protocolo en una URL absoluta.

- **405 Method Not Allowed**

  El método especificado en la solicitud no se permite para el recurso solicitado. Este error ocurre por ejemplo cuando se hace una petición POST y el servidor Web solamente permite peticiones GET para el tipo de archivo correspondiente a la URL solicitada.

- **406 Not Acceptable**

El recurso solicitado tiene características de contenido (idioma, conjunto de caracteres, etc.) que no son aceptables de acuerdo con el encabezado "Accept" de la solicitud.

- **407 PROXY Authentication Required**

  Similar al 401, pero esta vez la autenticación debe hacerse a través de un PROXY.

- **408 Request Timeout**

  Ocurrió un problema de comunicación de la red mientras se revisaba el vínculo. Revisa manualmente este vínculo o repite la revisión.

- **410 Gone**

  El recurso solicitado ya no está disponible y no se le conoce nueva dirección.

1.5.2.5. 500 - 599 Error del servidor

El servidor reconoce que no pudo llevar a cabo la acción solicitada.

- **500 Internal Server Error**

  El servidor web encontró una situación inesperada que le impidió cumplir con la solicitud hecha desde el cliente.

  Este error es causado generalmente por errores en scripts CGI, permisos incorrectos para un archivo/script ejecutable o problemas de configuración/operativos en el servidor.

- **501 Not Implemented**

  El servidor no tiene la funcionalidad requerida para llevar a cabo la acción solicitada. Error poco común.

- **502 Bad Gateway**

  El servidor, actuando como puerta de enlace o PROXY, recibió una respuesta no válida del servidor anterior en la cadena. Error no usual.

- **503 Service Unavailable**

  El servidor no puede efectuar la acción solicitada debido a una sobrecarga temporal o por estar en mantenimiento.

Este error ocurre típicamente cuando un sitio web hospedado ha sobrepasado su cuota de ancho de banda permitida para un periodo en particular.

- **504 Gateway Timeout**

  El servidor, actuando como puerta de enlace o PROXY, no recibió una respuesta válida a tiempo del servidor anterior en la cadena. Error no usual.

- **505 HTTP Version Not Supported**

  El servidor no soporta o rehúsa soportar la versión de protocolo HTTP que se usó en el mensaje de solicitud.

1.5.3. Campos de la cabecera de respuesta

Los campos de la cabecera de respuesta permiten al servidor pasar información adicional sobre la respuesta que no puede ser ubicada en la Línea de Estado. Estos campos dan información adicional sobre el servidor y sobre el futuro acceso al recurso identificado por la Petición URI.

- **Age:** estimación del tiempo transcurrido desde que se creó la respuesta.
- **Location:** se usa par a redirigir la petición a otra URI.
- **PROXY-Authenticate:** ante una respuesta con el código 407 (autentificación PROXY requerida), indica el esquema de autentificación.
- **Public:** da la lista de métodos soportados por el servidor.
- **Retry-After:** ante un servicio no disponible da una fecha para volver a intentarlo.
- **Server:** información sobre el servidor que maneja las peticiones.
- **Vary:** indica que hay varias respuestas y el servidor ha escogido una.
- **Warning:** usada para aportar información adicional sobre el estado de la respuesta.
- **WWW-Authenticate:** indica el esquema de autentificación y los parámetros aplicables a la URI.

2. SERVICIO FTP

2.1. Introducción

En un entorno de red, es natural que se tenga la necesidad de copiar archivos entre distintas computadoras. ¿Por qué a veces es tan complicado? Los fabricantes de computadoras han ideado cientos de sistemas de archivos, estos sistemas difieren en docenas de detalles menores y también en varios detalles mayores. El problema no radica sólo en que haya múltiples fabricantes, a veces resulta difícil copiar archivos entre dos tipos diferentes de computadores del mismo fabricante.

Cuando se trabaja en un entorno multisistema pueden aparecer entre otros los siguientes problemas:

- Convenciones diferentes para nombrar archivos.
- Reglas diferentes para recorrer los sistemas de directorios.
- Restricciones de acceso a archivos.
- Formas diferentes de representar texto y datos dentro de los archivos.

Los diseñadores de conjunto de protocolos TCP/IP no trataron de crear una solución general muy complicada que resolviera cualquier problema de transferencia de archivos. En lugar de ello, crearon un **Protocolo de Transferencia de Archivos** ***(FTP – File Transfer Protocol)*** bastante básico, pero elegante, útil y fácil de usar.

Los objetivos de FTP son:

- Promocionar el uso compartido de archivos (programas y/o datos).
- Animar al uso indirecto o implícito (a través de programas) de servidores remotos
- Hacer transparente al usuario las variaciones entre la forma de almacenar archivos en diferentes computadores
- Transferir datos fiable y eficientemente. El FTP, aunque puede ser utilizado directamente por un usuario en un terminal, está diseñado principalmente para ser usado por programas.

2.2. Historia, terminología y modelo FTP

En esta parte se conocerá un poco acerca de la historia, la terminología y el modelo FTP. Los términos que se definen son sólo aquellos que tienen un significado especial en el FTP

2.2.1. Historia

El FTP ha pasado por una larga evolución a través de los años. La primera propuesta de mecanismos para transferencia de archivos de se remonta al año 1971, la cual se desarrolló para su uso en servidores del Instituto de Tecnología de Massachussets *(MIT – Massachussets Institute of Technology)* bajo la Petición de Comentarios *(RFC – Request for Comments)* número 114.

El RFC 172 proporcionó un protocolo orientado al nivel de usuario para transferir archivos entre computadores. Una revisión de éste, el RFC 265, inició una revisión adicional, mientras que el RFC 281 sugirió más cambios. El uso de una transacción para elegir el tipo de datos se propuso en el RFC 294 en Enero de 1982.

El RFC 354 dejó obsoletos los RFC's 264 y 265. FTP se definió en este momento como un protocolo para transferencia de archivos entre computadores conectados a la red *ARPANET*, señalando como función principal la transferencia de archivos fiable y eficiente entre computadores, permitiendo el uso adecuado de las características de almacenamiento remotas. El RFC 385 siguió comentando errores, enfatizó algunos puntos y añadió características al protocolo,

mientras que el RFC 414 fue un informe sobre los servidores FTP que estaban funcionando. El RFC 430, que salió a la luz en 1973, presentó más comentarios sobre el FTP. Finalmente, se publicó un documento "oficial" sobre el FTP: el RFC 454.

Hacia julio de 1973, se hicieron considerables cambios a las últimas versiones del FTP, pero la estructura general permaneció igual. El RFC 542 se publicó como una nueva especificación "oficial" para reflejar esos cambios. Sin embargo, muchas implementaciones basadas en anteriores especificaciones no se actualizaron.

En 1974, los RFC's 607 y 614 continuaron los comentarios sobre el FTP. El RFC 624 propuso más cambios en el diseño y pequeñas modificaciones. En 1975, el RFC 686, titulado *"Leaving Well Enough Alone"*, trató las diferencias entre todas las primeras y las últimas versiones del FTP. El RFC 691 presentó una pequeña revisión del RFC 686, referente al tema de archivos para imprimir.

Motivado por la transición del Protocolo de Control de Red *(NCP – Network Control Protocol)* al Protocolo de Control de Transferencias *(TCP – Transfer Control Protocol)* como protocolo subyacente, nació un símbolo de todos los anteriores esfuerzos en el RFC 765 como la especificación de FTP para uso sobre TCP. Esta edición de la especificación FTP está dirigida a la corrección de pequeños errores en la documentación, a mejorar la explicación de algunas características del protocolo, y a añadir algunas nuevas órdenes opcionales. En particular, se incluyen las siguientes nuevas órdenes opcionales en esta edición de la especificación:

- CDUP - Cambiar al directorio padre *(Change to Parent Directory)*
- SMNT - Montar estructura *(Structure Mount)*
- STOU - Guardar con nombre único *(Store Unique)*
- RMD - Borrar directorio *(Remove Directory)*
- MKD - Crear directorio *(Make Directory)*
- PWD - Mostrar directorio actual *(Print Working Directory)*
- SYST - Sistema *(System)*

Esta especificación es compatible con la anterior edición. Un programa implementado conforme a la especificación anterior debería estar automáticamente en conformidad con esta especificación.

2.2.2. Terminología

2.2.2.1. ASCII

El conjunto de caracteres del Código Americano para el intercambio de información *(ASCII - American Standard Code for Information Interchange)* se considera tal y como está definido en el manual de protocolos de *ARPA-Internet*. En el FTP los caracteres ASCII se definen como la mitad inferior de un código de 8 bits (i.e., el bit más significativo es cero).

2.2.2.2. Controles de acceso

Los controles de acceso definen los privilegios de acceso del usuario para el uso de un sistema y de los archivos que hay en él. Son necesarios para evitar el uso no autorizado o accidental de archivos. Es una prerrogativa para un proceso servidor FTP utilizar los controles de acceso.

2.2.2.3. Tamaño de *byte*

Hay dos tamaños de byte interesantes en el FTP: el tamaño lógico de byte del archivo y el tamaño de byte usado para la transmisión de los datos. El tamaño de byte utilizado para la transmisión es siempre de 8 bits. Este tamaño no es necesariamente el tamaño de byte con el que se guardan los datos en un sistema ni el tamaño lógico de byte para la interpretación de la estructura de los datos.

2.2.2.4. Conexión de control

La ruta de comunicación entre el USER-PI y el SERVER-PI para el intercambio de órdenes y respuestas.

2.2.2.5. Conexión de datos

Una conexión bidireccional sobre la que se transfieren los datos en un modo y tipo especificados. Los datos transferidos pueden ser una parte de un archivo, un archivo entero o un cierto número de archivos. La conexión se puede establecer entre un servidor y un cliente o entre dos servidores.

2.2.2.6. Puerto de datos

El proceso de transferencia pasiva de datos "escucha" en el puerto de datos hasta que recibe una conexión del proceso de transferencia activa para abrir la conexión de datos.

2.2.2.7. DTP (Proceso de Transferencia de Datos)

El Proceso de Transferencia de Datos *(DTP - Data Transfer Process)* establece y maneja la conexión de datos. Puede ser activo o pasivo.

2.2.2.8. EOL (Fin de Línea)

La secuencia Fin-De-Línea *(EOL - End-Of-Line)* define la separación entre líneas. La secuencia consiste en un retorno de carro seguido de un carácter de nueva línea.

2.2.2.9. EOF (Fin-de-Archivo)

La condición fin-de-archivo que define el final de un archivo en proceso de transmisión.

2.2.2.10. EOR (Fin de Registro)

La condición Fin-de-Registro *(EOR – End-Of-Record)* que define el final de un registro en proceso de transferencia.

2.2.2.11. Recuperación de errores

Un procedimiento que permite al usuario recuperar el control a partir de ciertas condiciones de error, como un fallo en el servidor o en el proceso de transferencia. En el FTP, la recuperación de errores puede implicar el reinicio de la transferencia de un archivo a partir de un cierto punto.

2.2.2.12. Órdenes FTP

Un conjunto de órdenes que abarcan la información de control que va desde el proceso usuario-FTP al proceso servidor-FTP.

2.2.2.13. Archivo

Un conjunto ordenado de datos del computador (incluyendo programas), de longitud arbitraria, definido de forma única por una ruta.

2.2.2.14. Modo

El modo en que se trasfieren datos por la conexión de datos. El modo define el formato de los datos durante la transferencia, incluyendo EOR y EOF. Los modos de transferencia definidos para FTP se describen en la sección Modos de Transmisión.

2.2.2.15. Página

Un archivo se puede estructurar como un conjunto de partes independientes llamadas páginas. El FTP soporta la transmisión de archivos discontinuos como páginas indexadas independientes.

2.2.2.16. Nombre de ruta

El nombre de ruta se define como una cadena de caracteres que el usuario pasa al sistema de archivos para identificar un archivo. La ruta normalmente contiene nombres de dispositivos y/o directorios y un nombre de archivo. El FTP no especifica aún un estándar para indicar una ruta. Cada usuario debe seguir las normas que dictan los sistemas de archivos implicados en la transferencia para nombrar los archivos.

2.2.2.17. PI (Intérprete de Protocolo)

El Intérprete del Protocolo *(PI - Protocol Interpreter)* se implementa como *User-PI* y *Server-PI* en la parte del usuario y del servidor respectivamente.

2.2.2.18. Registro

Un archivo secuencial se puede estructurar en un número de partes contiguas llamadas registros. El FTP soporta las estructuras de registro, pero un archivo no tiene por qué tener una estructura de registro.

2.2.2.19. Respuesta

Una respuesta es un asentimiento (positivo o negativo) enviada por el servidor al usuario a través de la conexión de control en respuesta a una orden FTP. La forma general de una respuesta es un código de terminación seguido de una cadena de texto. Los códigos son usados por los programas y el texto por las personas.

2.2.2.20. Server-DTP (Proceso de Transferencia de Datos del Servidor)

El proceso de transferencia de datos del servidor en su estado normal de "activo", establece una conexión de datos con el puerto de datos que está a la espera. Prepara los parámetros para transferir y almacenar y transfiere los datos cuando así se requiere a través de su PI. El DTP se puede poner en estado "pasivo" para esperar una conexión, en lugar de iniciarla.

2.2.2.21. Proceso *Server-FTP*

Un proceso o un conjunto de ellos que realizan la función de transferencia de archivos en conjunción con el proceso *User-FTP* y, posiblemente, otro servidor. Las funciones consisten en un intérprete de protocolo (PI) y un proceso de transferencia de datos (DTP).

2.2.2.22. Proceso *Server-PI*

El intérprete de protocolo del servidor "escucha" en el puerto “L” hasta que recibe una conexión de un *User-PI* y establece una conexión de comunicaciones para control. Recibe órdenes FTP estándar desde el *User-PI*, envía respuestas y controla el *Server-DTP*.

2.2.2.23. Tipo

El tipo de representación de datos usado para transferir y almacenar los datos. El tipo implica ciertas transformaciones a la hora de almacenar y enviar los datos. Los tipos de representación definidos en el FTP se describen en la sección titulada "Estableciendo conexiones de datos".

2.2.2.24. Usuario

Una persona o un proceso en su lugar que desea utilizar el servicio de transferencia de archivos. La persona puede interactuar directamente con el proceso *Server-FTP*, pero se prefiere el uso de un proceso *User-FTP* ya que el diseño del protocolo está orientado a su utilización por autómatas.

2.2.2.25. User-DTP (Proceso de Transferencia de Datos de Usuario)

El proceso de transferencia de datos del usuario espera a recibir una conexión en el puerto de datos desde el proceso *Server-FTP*. Si dos servidores están transfiriendo datos entre ellos, el *User-DTP* permanece inactivo.

2.2.2.26. Proceso *User-FTP*

Un conjunto de funciones incluyendo un intérprete de protocolo, un proceso de transferencia de datos y una interfaz de usuario que juntos realizan la función de transferir archivos en cooperación con un proceso *Server-FTP*. La interfaz de usuario permite usar un lenguaje local en el diálogo orden-respuesta con el usuario.

2.2.2.27. *User-PI*

El intérprete de protocolo de usuario inicia la conexión de control desde su puerto "U" hasta el proceso *Server-FTP*, envía órdenes FTP y controla el *User-DTP* si es necesario para la transferencia de archivos.

2.2.3. El modelo FTP

Teniendo en cuenta las definiciones anteriores, el siguiente modelo (mostrado en la Figura 1) representa un diagrama de un servicio FTP.

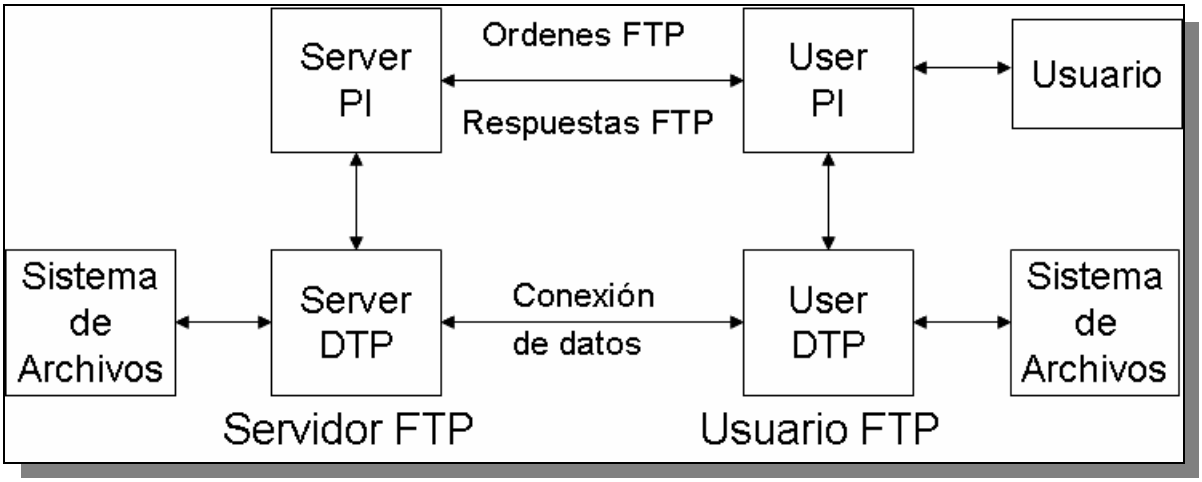

Figura 2. Modelo para el uso del FTP.

- La conexión de datos es bidireccional.
- La conexión de datos no tiene por qué existir todo el tiempo.

En el modelo descrito en la Figura 1, el intérprete de protocolo de usuario *(User-PI),* inicia la conexión de control. Las órdenes FTP estándar las genera el *User-PI* y se transmiten al proceso servidor a través de la conexión de control. (El usuario puede establecer una conexión de control directa con el *Server-FTP*, desde un terminal por ejemplo, y enviar órdenes FTP estándar, obviando así el proceso *User-FTP*). Las respuestas estándar se envían desde el *Server-PI* al *User-PI* por la conexión de control como contestación a las órdenes.

Las órdenes FTP especifican parámetros para la conexión de datos (puerto de datos, modo de transferencia, tipo de representación y estructura) y la naturaleza de la operación sobre el sistema de archivos (almacenar, recuperar, añadir, borrar, etc.). El *User-DTP* u otro proceso en su lugar, debe esperar a que el servidor inicie la conexión al puerto de datos especificado y transferir los datos en función de los parámetros que se hayan especificado. Se debe reseñar que el puerto de datos no tiene por qué estar en el mismo equipo que envía las órdenes FTP a través de la conexión de control, pero el usuario o el proceso *User-FTP* debe asegurarse de que se espera la conexión en el puerto de datos indicado. También hay que destacar que la conexión de datos se puede usar simultáneamente para enviar y para recibir.

En otra situación, un usuario puede querer transferir archivos entre dos computadores que no son locales. El usuario prepara las conexiones de control con los dos servidores y da las órdenes necesarias para crear una conexión de datos entre ellos. De esta forma, la información de control se envía al *User-PI* pero los datos se transfieren entre los procesos de transferencia de datos de los servidores. A continuación se ilustra en la Figura 2 un modelo de esta interacción servidor-servidor.

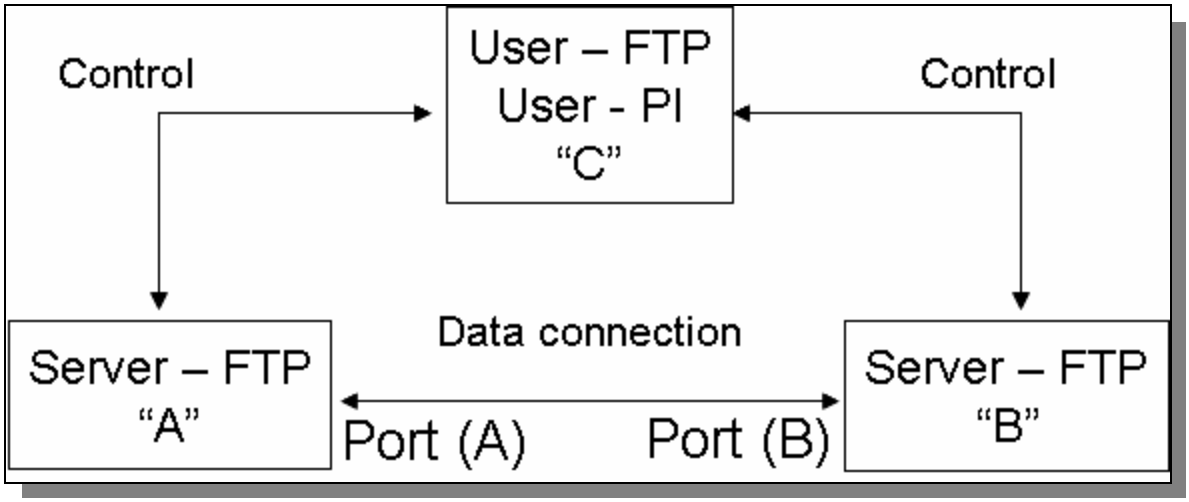

Figura 3. Modelo interacción servidor-servidor.

El protocolo requiere que las conexiones de control permanezcan abiertas mientras se transfieren datos. Es responsabilidad del usuario solicitar el cierre de las conexiones de control una vez termine de usar el servicio, mientras que el servidor es el encargado de cerrarlas. El servidor puede interrumpir la transferencia de datos si las conexiones de control se cierran sin previa solicitud.

2.3. Funciones de transferencia de datos

Los archivos sólo se transmiten por la conexión de datos. La conexión de control se usa para enviar órdenes, que describen la función a realizar, y las repuestas a estas órdenes. Varias órdenes se refieren a la transferencia de datos entre computadores. Estas órdenes incluyen

"MODE" (modo) que especifica cómo se transmiten los bits de datos y las órdenes "STRU" (*Structure* - Estructura) y TYPE, que se usan para definir la forma de representación de los datos. La transmisión y la representación son básicamente independientes, pero el modo de transmisión flujo depende de la estructura del archivo y si se especifica el modo de transmisión "comprimido", la naturaleza del byte de relleno depende del tipo de representación.

2.3.1. Representación de datos y almacenamiento

Los datos se transfieren desde un dispositivo de almacenamiento en el computador que envía los datos a otro en el computador que los recibe. A menudo es necesario realizar ciertas transformaciones en los datos porque la representación de los datos almacenados varía de un computador a otro. Por ejemplo los datos de la especificación ASCII de una Terminal Virtual de Red NVT-ASCII *(NTV – Network Virtual Terminal)* tiene diferentes representaciones de los datos almacenados en diferentes sistemas.

El sistema operativo "TOPS-20" de la Corporación de Equipos Digitales (DEC - *Digital Equipment Corporation)* generalmente almacenan NVT-ASCII como cinco caracteres ASCII de 7 bits, justificados a la izquierda en una palabra de 36 bits. Es conveniente convertir caracteres a la representación estándar NVT-ASCII cuando se transmite texto entre sistemas diferentes. Los computadores que intervienen en la transferencia deberían realizar las transformaciones necesarias entre la representación estándar y su representación interna.

Nos encontramos con un problema diferente cuando se transmiten datos en forma binaria (no caracteres) entre computadores con distinto tamaño de palabra. No siempre está claro cómo se deben enviar los datos y como se deben almacenar al recibirlos. Por ejemplo, cuando se transmiten bytes de 32 bits desde un sistema con tamaño de palabra de 32 bits a un sistema con tamaño de palabra de 36 bits, sería conveniente (por razones de eficiencia y utilidad) almacenar los bytes de 32 bits justificados a la derecha en una palabra de 36 bits en el sistema receptor. En cualquier caso, el usuario debería tener la opción de especificar la representación de los datos y las transformaciones realizadas en ellos. Hay que señalar que el FTP proporciona muy limitados tipos de representación. Las transformaciones necesarias más allá de esta limitada capacidad las deberá realizar el usuario directamente.

2.3.1.1. Tipos de datos

El usuario especifica las representaciones de datos que se manejarán en el FTP. Este tipo puede definirse de forma implícita como: Estándar Americano para el Intercambio de Datos (**ASCII** – American Standard for Information Interchange) ó Código Extendido de Intercambio Decimal Codificado en Binario (**EBCDIC** – Extended Binary Coded Decimal Interchange Code); o de forma explícita un tamaño de byte para la interpretación del denominado "tamaño de byte lógico". Hay que tener en cuenta que esto no tiene nada que ver con el tamaño de byte usado para la transmisión a través de la conexión de datos, llamado "tamaño de byte de transferencia", y no debería haber confusión entre ambos. Por ejemplo, NVT-ASCII tiene un tamaño de byte lógico de 8 bits. Si el tipo es tamaño de byte local entonces la orden “TYPE” tiene un segundo parámetro obligatorio especificando el tamaño de byte lógico. El tamaño del byte de transferencia es siempre de 8 bits.

2.3.1.2. Tipo ASCII

Este es el tipo por defecto y debe ser admitido por todas las implementaciones del FTP. Su principal propósito es transferir archivos de texto, excepto cuando ambos computadores encuentran el tipo EBCDIC más conveniente. El computador que envía los datos, los convierte de una representación de caracteres interna a la representación estándar NVT-ASCII de 8 bits. El receptor convertirá los datos del tipo estándar a su propia forma interna.

De acuerdo con el estándar NVT, la secuencia de línea de retorno de carácter “<CRLF>” *(Character Return Line Feed)* se debe usar donde sea necesario para indicar el final de una línea de texto. Usar la representación estándar NVT-ASCII implica que los datos se deben interpretar como bytes de 8 bits.

2.3.1.3. Tipo EBCDIC

El propósito de este tipo es la transferencia eficaz entre computadores que usan EBCDIC como su representación de carácter interna.

Para la transmisión, los datos están en forma de caracteres EBCDIC de 8 bits. El código de carácter es la única diferencia entre la especificación funcional de los tipos EBCDIC y ASCII.

2.3.1.4. Tipo imagen

Los datos se envían como bits contiguos que, para la transferencia, están empaquetados en bytes de transferencia de 8 bits. El receptor debe almacenar los datos como bits contiguos. La estructura del sistema de almacenamiento puede necesitar el rellenado del archivo (o de cada registro, para un archivo estructurado en registros) hasta el límite pertinente (byte, palabra o bloque). Este rellenado, que debe ser todo ceros, sólo puede tener lugar al final del archivo (o de cada registro) y debe haber una forma de identificar los bits de relleno para poder eliminarlos si se obtiene el archivo. La transformación de relleno debe ser conocida por el usuario para procesar el archivo en el lugar de recepción. El tipo imagen está indicado para el almacenamiento y obtención de archivos y la transferencia de datos binarios. Se recomienda que todas las implementaciones del FTP acepten este tipo.

2.3.1.5. Tipo local

Los datos se transfieren en bytes lógicos del tamaño especificado por el segundo parámetro obligatorio, tamaño de byte. El valor del tamaño de byte debe ser un entero decimal; no hay valor por defecto. El tamaño de byte lógico no es necesariamente el mismo que el tamaño de byte de transferencia. Si hay diferencia entre los tamaños de byte, entonces los bytes lógicos se deben empaquetar contiguamente, a pesar de los límites del byte de transferencia y con el relleno necesario al final.

Cuando los datos llegan al receptor, se transformarán de una manera independiente del tamaño de byte lógico y del computador en cuestión. Esta transformación debe ser invertible (i.e., se obtendrá el mismo archivo si se usan los mismos parámetros) y los desarrolladores de la implementación deben hacerla pública.

Por ejemplo, un usuario enviando números en coma flotante de 36 bits a un computador con tamaño de palabra de 32 bits puede enviar esos datos como byte local con un tamaño lógico de 36. El receptor debería almacenar los bytes lógicos de forma que sea fácil su manipulación; en este ejemplo, poner los bytes lógicos de 36 bits en palabras dobles de 64 bits sería suficiente.

En otro ejemplo, un par de computadores con tamaño de palabra de 36 bits, pueden enviarse datos en palabras usando "TYPE L 36". Los datos se enviarían en los bytes de transmisión de 8 bits y empaquetados para que 9 bytes de transmisión lleven dos palabras de 36 bits.

2.3.1.6. Control de formato

Los tipos ASCII y EBCDIC llevan un segundo parámetro (opcional); este es para indicar qué tipo de control de formato vertical, si es que lo hay, se asocia con el archivo. Un archivo de texto se envía a un computador por uno de estos tres propósitos: para su impresión, para almacenarlo y posteriormente recuperarlo, o para procesarlo. Si se envía un archivo para imprimirlo, el receptor debe saber cómo está representado el control de formato vertical. En el segundo caso, debe ser posible almacenar el archivo y después recuperarlo exactamente igual. Por último, debería ser posible enviar un archivo de un computador a otro y procesarlo en el receptor sin problemas indebidos. Un formato ASCII o EBCDIC por sí solo no satisface estas tres condiciones. Por lo tanto, estos tipos tiene un segundo parámetro especificando uno de los siguientes tres formatos:

2.3.2. Estableciendo conexiones de datos

La mecánica de transferir datos consiste en preparar la conexión de datos en los puertos apropiados y elegir los parámetros para la transferencia. El usuario y los *Server-DTP's* tienen ambos un puerto por defecto. El puerto por defecto del proceso de usuario es el mismo que el puerto de la conexión de control (i.e., "U"). El puerto por defecto del proceso servidor es el puerto adyacente al puerto de la conexión de control (i.e., "L-1").

El tamaño de byte para la transferencia es de 8 bits. Este tamaño sólo es relevante para la transferencia de datos; no tiene nada que ver con la representación de los datos dentro del sistema de archivos de un computador.

El proceso de transferencia de datos pasivo (que puede ser un *User-DTP* o un segundo *Server-DTP*) deberá "escuchar" en el puerto de datos antes de enviar una orden de petición de transferencia. Esta orden determina el sentido de la transferencia de los datos. El servidor, una vez recibida la petición de transferencia, iniciará la conexión de datos al puerto indicado. Una vez establecida la conexión, la transferencia de datos comienza entre los DTP's y el *Server-PI* envía

una respuesta de confirmación al *User-PI*. Todas las implementaciones del FTP deben soportar el uso de los puertos de datos por defecto y sólo el *User-PI* puede solicitar un cambio a un puerto diferente.

Usando la orden "PORT", el usuario puede especificar un puerto alternativo. Puede que el usuario quiera volcar un archivo en una impresora de líneas o que se obtenga el archivo de un tercer computador. En este último caso, el *User-PI* establece las conexiones de control con ambos computadores. Luego dice a un servidor (mediante una orden FTP) que "escuche" hasta recibir una conexión que el otro iniciará. El *User-PI* envía aun *Server-PI* una orden "PORT" indicando el puerto de datos del otro. Finalmente, se envían a ambos las órdenes apropiadas de transferencia. La secuencia exacta de órdenes y respuestas entre el usuario y los servidores está en la sección sobre Respuestas FTP. En general, es responsabilidad del servidor mantener la conexión de datos (abrirla y cerrarla). La excepción a esto se produce cuando el *User-DTP* envía los datos en un modo de transferencia que requiere cerrar la conexión para indicar EOF. El servidor debe cerrar la conexión de datos bajo las siguientes circunstancias:

- El servidor ha terminado de enviar datos en un modo de transferencia que requiere un cierre para indicar EOF.
- El servidor recibe una orden "ABORT" del usuario.
- Una orden del usuario especifica un nuevo puerto.
- La conexión de control se cierra correctamente o de cualquier otro modo.
- Se produce una condición de error irrecuperable.

En cualquier caso, el cierre es una opción del servidor que se debe indicar al proceso de usuario sólo mediante una respuesta 250 ó 226.

2.3.3. Manejo de la conexión de datos

Puertos de la conexión de datos por defecto: Todas las implementaciones del FTP deben soportar el uso de los puertos de datos por defecto y sólo el *User-PI* puede iniciar el uso de otros puertos.

Negociando puertos de datos distintos a los que hay por defecto: el *User-PI* puede especificar un puerto de datos diferente por su parte con la orden "PORT". El *User-PI* puede solicitar al servidor la localización de un puerto en el propio servidor con la orden "PASV". Como una conexión está definida por el par de direcciones, cualquiera de estas acciones es suficiente para tener una conexión de datos diferente, pero se permite usar ambas órdenes para utilizar nuevos puertos en los dos lados de la conexión.

Reutilización de la conexión de datos: Cuando se usa el modo flujo para transferencia de datos el final de archivo se indica cerrando la conexión. Esta causa un problema si se van a transferir varios archivos en la sesión, debido a la necesidad que tiene el TCP de mantener el registro de la conexión por un tiempo de espera para garantizar una comunicación fiable. Por eso la conexión no puede reabrirse inmediatamente. Hay dos soluciones a este problema. La primera es negociar un puerto diferente al que hay por defecto. La segunda es usar otro modo de transmisión.

Un comentario sobre los modos de transmisión. El modo flujo de transferencia es implícitamente poco fiable porque no se puede determinar si la conexión se ha cerrado prematuramente o no. Los otros modos de transferencia (de bloque, comprimido) no cierran la conexión para indicar el final de archivo. Los datos que envía el FTP a través de ellos hacen que se pueda determinar el final de archivo. Por eso, usando estos modos, se puede dejar la conexión de datos abierta para transferir más de un archivo.

2.3.4. Modos de transmisión

Otro aspecto a considerar en la transferencia de datos es la elección apropiada del modo de transmisión. Hay tres modos: uno que formatea los datos y permite reiniciar la transmisión; otro que además comprime los datos para una transferencia eficaz; y otro que pasa los datos sin apenas procesarlos. En este último caso, el modo interactúa con el atributo de estructura para determinar el tipo de proceso. En el modo comprimido, el tipo de representación determina el byte de relleno.

Todas las transferencia de datos se deben complementar con un EOF que se puede indicar explícita o implícitamente cerrando la conexión de datos. Para archivos con estructura de registro, todos los indicadores de fin-de-registro (EOR) son explícitos, incluyendo el último. Para archivos

transmitidos con estructura de página, se usa una página con el indicador de última página activado.

Para estandarizar la transferencia, el computador que envía los datos trasladará sus caracteres de fin de línea o de fin de registro a la representación indicada por el modo de transferencia y la estructura del archivo y el receptor realizará la traducción inversa a su representación interna. Un campo de contador de registro de un *"Mainframe"* de IBM puede no ser reconocido por otro computador, por eso, la información de fin-de-registro se puede transferir como un código de control de dos bytes en modo flujo o como un bit activado en un descriptor de modo bloque o comprimido. El fin-de-línea en un archivo ASCII o EBCDIC no estructurado en registros se debería indicar con "<CRLF>" o "<NL>" respectivamente. Debido a que estas transformaciones implican un trabajo extra para algunos sistemas, sistemas idénticos que transfieren entre ellos un archivo de texto no estructurado en registros pueden preferir usar una representación binaria y modo flujo para la transferencia.

Se definen los siguientes modos de transmisión en el FTP:

2.3.4.1. Modo flujo

Los datos se transmiten como un flujo de bytes. No hay ninguna restricción en el tipo de representación usado; se permiten estructuras de registro.

En un archivo estructurado en registros, EOR y EOF se indicarán por un código de control de dos bytes. El primer byte del código de control consistirá en todo unos, el carácter de escape. El segundo tendrá el bit de orden más bajo activado y ceros en el resto para EOR y el segundo bit de orden más bajo activado para EOF; esto es, el byte valdrá 1 para EOR y 2 para EOF. EOF y EOR se pueden indicar conjuntamente en el último byte transmitido activando los dos bits más bajos (i.e., el valor 3). Si se pretende transmitir un byte con todo unos como dato, se debería repetir em el segundo byte del código de control.

2.3.4.2. Modo bloque

El archivo se transmite como una serie de bloques de datos precedidos por uno o más bytes cabecera. Los bytes de la cabecera contienen un campo contador y un código descriptor. El campo

contador indica la longitud total del bloque de datos en bytes, indicando así el inicio del siguiente bloque de datos (no hay bits de relleno). El código descriptor define: último bloque del archivo (EOF), último bloque del registro (EOR), indicador de reinicio (vea la sección sobre Recuperación de Errores y Reinicio) o datos sospechosos (i.e., los datos transferidos pueden contener errores y no son fiables). Este último código no es para controlar errores desde el propio FTP. Su uso está motivado por el deseo de ciertos sitios que intercambian cierto tipo de información (por ejemplo, datos sísmicos o meteorológicos) enviando y recibiendo todos los datos a pesar de que pueda haber errores locales (como errores leyendo una cinta magnética). Se permiten estructuras de registro y se puede usar cualquier tipo de representación.

La cabecera consiste en tres bytes. De los 24 bits de información, los 16 más bajos representan un contador de bytes y los 8 más altos representan códigos de descripción como se muestra en la figura 3.

Figura 4. Cabecera de bloque.

Los códigos de descripción se indican mediante bits en el byte de descripción. Hay asignados 4 códigos, donde cada número de código es el valor decimal del correspondiente bit en el byte.

Códigos y sus significados

128: El final del bloque de datos es EOR

64: El final del bloque de datos es EOF

32: Puede haber errores en el bloque

16: El bloque de datos es un marcador de reinicio

Con esta codificación, más de una condición codificada puede ocurrir para un bloque en particular. Se pueden activar tantos bits como sea necesario. El indicador de reinicio está incluido en el flujo de datos como un número de 8 bits representando caracteres imprimibles en el lenguaje usado en la conexión de control (por ejemplo, NVT-ASCII). <SP> (*Space* - Espacio) no se debe usar dentro de un indicador de reinicio. Por ejemplo, para transmitir un indicador de seis caracteres, se debería enviar lo siguiente:

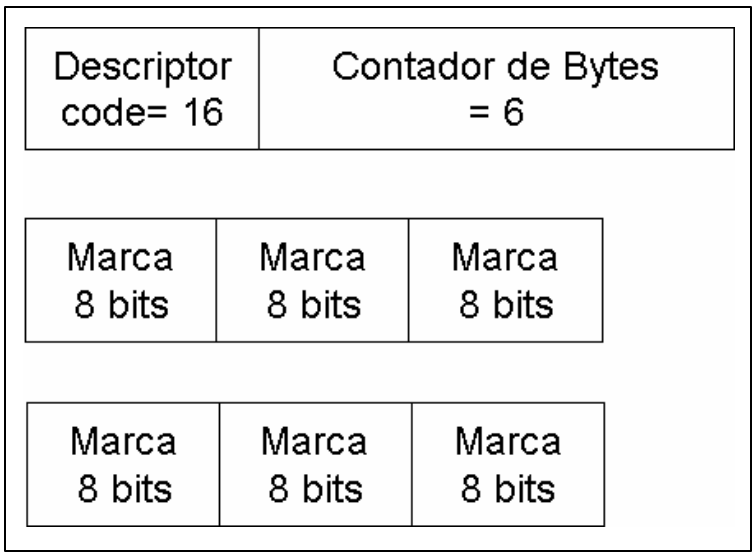

Figura 5. Transmisión de indicador con 6 caracteres.

2.3.4.3. Modo comprimido

Hay tres clases de información a enviar: datos normales, enviados en una cadena de bytes; datos comprimidos, formados por repeticiones o relleno; e información de control, enviada en una secuencia de escape de dos bytes. Si se envía n>0 bytes (hasta 127) de datos normales, estos n bytes van precedidos de un byte con el bit más a la izquierda a cero y los 7 bits más a la derecha contienen el número n.

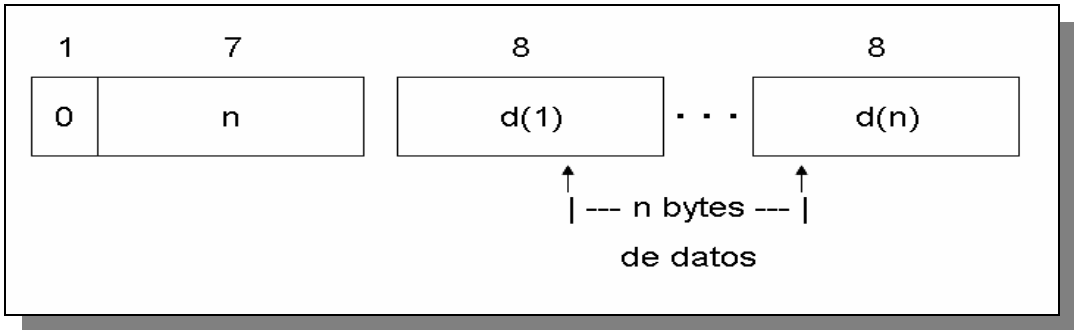

Figura 6. Cadena de bytes

- Cadena de n bytes de datos d(1),..., d(n)
- El contador n debe ser positivo.

Para comprimir una cadena de n repeticiones del byte de datos d, se envían los 2 siguientes bytes:

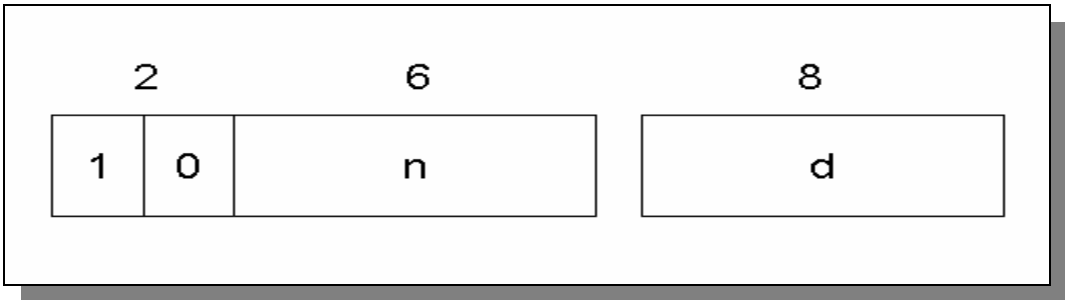

Figura 7. Ejemplo de byte repetido

Una cadena de n bytes de relleno se puede comprimir en un solo byte, donde el byte de relleno varía según el tipo de representación. Si el tipo es ASCII o EBCDIC, el byte de relleno es <SP> (espacio, código ASCII 32, código EBCDIC 64). Si el tipo es imagen o byte local, el relleno es un byte cero.

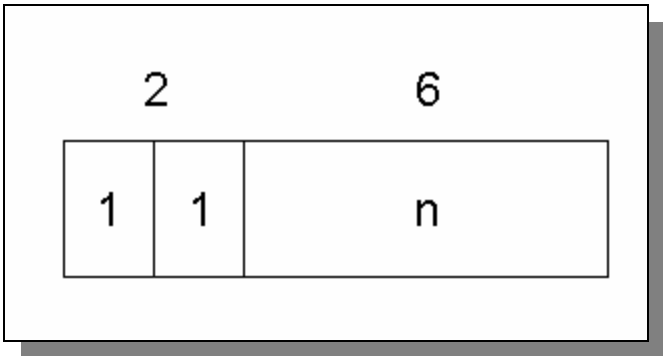

Figura 8. Cadena de relleno

La secuencia de escape está formada por dos bytes, el primero de los cuales es el byte de escape (todo ceros) y el segundo contiene códigos de descripción según lo definido para el modo bloque. Los códigos tienen el mismo significado que anteriormente y se aplican a las cadenas de bytes transmitidas con éxito.

EL modo comprimido es útil para obtener un ancho de banda adicional en transmisiones muy largas con un costo extra de uso de la unidad central de proceso *(CPU – Central Process Unit).* Se puede usar de forma muy efectiva para reducir el tamaño de archivos para imprimir como los generados por computadores RJE.

2.3.5. Recuperación de errores y reinicio

No hay nada que detecte bits perdidos o alterados en las transmisiones de datos; este nivel de control de errores lo maneja el TCP. Sin embargo, se proporciona un medio para recomenzar transferencia para proteger a los usuarios de fallos graves en el sistema (incluyendo fallos de un computador, un proceso FTP o de la red usada).

Sólo está definido el procedimiento de reinicio para los modos de transferencia de bloque y comprimido. Requiere que el que envía datos inserte un código marcador especial en el flujo de datos con información que indique por dónde vamos. Esta información sólo tiene significado para el computador que envía los datos, pero debe consistir en caracteres imprimibles en el lenguaje negociado o por defecto de la conexión de control (ASCII o EBCDIC). El indicador puede representar una cuenta de los bits, los registros o cualquier otra información por la que un sistema

pueda identificar un punto de control. El receptor de los datos, si implementa el procedimiento de reinicio, debería guardar la correspondiente posición de este marcador en el sistema receptor y devolver esta información al usuario.

En el caso de un fallo del sistema, el usuario puede reiniciar la transferencia de datos identificando el último marcador recibido con el procedimiento de recomenzar del FTP. El siguiente ejemplo ilustra el uso de este procedimiento:

El que envía datos inserta un bloque marcador apropiado en el flujo de datos en un punto conveniente. El que recibe marca el correspondiente punto en su sistema de archivos y proporciona al usuario información del último marcador recibido, ya sea directamente o por la conexión de control en una respuesta “110” (dependiendo de quién envíe el archivo). Si ocurre un error del sistema, el usuario o proceso reinicia el servidor en el último punto que este indicó enviando la orden recomenzar con el marcador del servidor como argumento. La orden se transmite por la conexión de control y es seguida inmediatamente por la orden (ya sea “RETR”, “STOR” o “LIST”) que se estaba ejecutado cuando ocurrió el error.

2.4. Funciones de transferencia de archivos

El canal de comunicación entre el *User-PI* y el *Server-PI* se establece como una conexión TCP desde el usuario al puerto estándar. El intérprete de protocolo de usuario es responsable de enviar órdenes FTP e interpretar las respuestas recibidas; el *Server-PI* interpreta las órdenes, envía respuestas y controla su DTP para establecer la conexión de datos y transferirlos. Si el proceso de transferencia pasiva es el *User-DTP*, entonces está controlado a través del protocolo interno del computador donde se ejecuta el *User-DTP*; si es otro server-DTP, está controlado a través de órdenes recibidas en su PI desde el *User-PI*.

Por tanto, el formato para respuestas multilínea consiste en que la primera línea empieza con el código de respuesta requerido seguido inmediatamente de un guión, "-" (también es el signo menos), seguido de el texto. La última línea empezará con el mismo código, seguido

inmediatamente por un espacio <SP>, opcionalmente texto, y el código "TELNET" de fin de línea. Cada tres dígitos de la respuesta tienen un significado especial.

Se pretende permitir un rango de respuestas desde muy simple a muy sofisticado para el proceso de usuario. El primer dígito denota si la respuesta es buena, mala o incompleta. Un proceso de usuario poco sofisticado podrá determinar su próxima acción simplemente examinando el primer dígito. Un proceso de usuario que quiera conocer aproximadamente el tipo de error ocurrido (por ejemplo, error del sistema de archivos o error de sintaxis) puede examinar el segundo dígito, reservando el tercero para una mayor precisión en la información (por ejemplo, orden "RNTO" sin ser precedida por una "RNFR").

Existen cinco valores para el primer dígito del código de respuesta:

- 1yz Respuesta preliminar positiva. Se ha iniciado la acción requerida; se espera otra respuesta antes de seguir con una nueva orden. (El proceso de usuario que envía otra orden antes de que la respuesta de finalización llegue, estará violando el protocolo, pero el proceso servidor debería encolar cualquier orden que reciba mientras está ejecutando una orden anterior.) Este tipo de respuesta se puede usar para indicar que se ha aceptado la orden y que el proceso de usuario debería ocuparse ahora de la conexión de datos, en implementaciones donde controlar ambas conexiones a la vez es difícil. El proceso servidor puede enviar, a lo sumo, una respuesta 1yz por orden recibida.
- 2yz Respuesta de finalización positiva. La acción requerida se ha completado satisfactoriamente. Se puede iniciar una nueva orden.
- 3yz Respuesta intermedia positiva. La orden se ha aceptado, pero se está pendiente de recibir más información para completarla. El usuario debería enviar otra orden indicando esta información. Esta respuesta se utiliza en órdenes que deben ir en secuencia.
- 4yz Repuesta de finalización negativa transitoria. La orden no se ha aceptado y la acción requerida no se ha llevado a cabo, pero la condición de error es temporal y se puede solicitar la acción de nuevo. El usuario debería volver al principio de la secuencia de comandos, si es que la hay. Es difícil dar un significado a "transitoria", particularmente cuando dos sistemas diferentes (el servidor y el usuario) deben estar de acuerdo en la interpretación. Cada respuesta la categoría 4yz puede tener un valor diferente, pero se

pretende con ella que el proceso de usuario reintente la operación. Una regla para determinar si una respuesta pertenece a la categoría 4yz o a la 5yz (permanente negativa) es que las respuestas son 4yz si la orden se puede repetir sin cambios en la forma o en las propiedades del usuario o del servidor (por ejemplo, la orden y sus argumentos son los mismos; el usuario no cambia su cuenta ni su nombre; el servidor no lo interpreta de diferente forma.

- 5yz Respuesta de finalización negativa permanente. La orden no se ha aceptado y la acción requerida no ha tenido lugar. El proceso de usuario no debería repetir la misma petición (en la misma secuencia). Incluso algunas condiciones de error "permanente" se pueden corregir, por eso el usuario (la persona) puede hacer posteriormente que su proceso de usuario repita posteriormente la orden (por ejemplo, se corrige el argumento, o el usuario cambia el estado de su directorio.)

Las siguientes agrupaciones de funciones se codifican en el segundo dígito:

- x0z Sintaxis. Estas respuestas se refieren a errores de sintaxis, órdenes correctas sintácticamente pero que no encajan en ninguna otra categoría, órdenes no implementadas o superfluas.
- x1z Información. Estas son respuestas a solicitudes de información como STATUS o HELP.
- x2z Conexiones. Respuestas referidas a las conexiones de control y de datos.
- x3z Autenticación y cuenta. Respuestas para el proceso de entrada al sistema y procedimientos de cuenta.
- x4z Sin especificar aún.
- x5z Sistema de archivos. Estas respuestas indican el estado del sistema de archivos en el servidor según se realizan transferencias u otras acciones sobre el sistema de archivos. El tercer dígito afina más en el significado de cada una de las categorías indicadas por el segundo. La lista de respuestas de la siguiente sección mostrará esto. El texto asociado con cada respuesta es sólo una recomendación, no una obligación, y puede incluso cambiar según la orden asociada. Los códigos de respuesta, por otra parte, deben seguir estrictamente las especificaciones; es decir, las implementaciones de servidores no deben

inventarse nuevos códigos para situaciones que son ligeramente diferentes a las descritas aquí, sino que deberían adaptarse a los códigos ya definidos. Una orden como "TYPE" o "ALLO" cuya correcta ejecución no proporciona al usuario ninguna nueva información debería devolver un código "200". Si la orden no se ha implementado porque no tiene sentido para un determinado sistema, por ejemplo "ALLO" en un TOPS-20, una respuesta de finalización positiva es conveniente para no confundir al proceso de usuario. En este caso se usa una respuesta "202" con, por ejemplo, el siguiente texto: "No es necesario solicitar espacio para el archivo." Si, por otra parte, la orden no es específica a ningún sistema y no está implementada, la respuesta debe ser "502". Un caso particular de esto es la respuesta "504" para una orden que está implementada pero que se solicita con un argumento no implementado.

Códigos de respuesta por grupos funcionales

- 200 Orden correcta.
- 500 Error de sintaxis, comando no reconocido. Esto puede incluir errores como línea de orden demasiado larga.
- 501 Error de sintaxis en parámetros o argumentos.
- 202 Orden no implementada, no necesaria en este sistema.
- 502 Orden no implementada.
- 503 Secuencia de órdenes incorrecta.
- 504 Orden no implementada para ese parámetro.
- 110 Respuesta de marcador de reinicio. En este caso, el texto debe ser: MARK yyyy = mmmm Donde yyyy es el marcador del flujo de datos en el proceso de usuario y mmmm es el equivalente en el servidor (atención a los espacios entre los marcadores y el "=").
- 211 Estado del sistema o respuesta de ayuda del sistema.
- 212 Estado del directorio.
- 213 Estado del archivo.

- 214 Mensaje de ayuda. Sobre como usar el servidor o el significado de una orden particular no estándar. Esta respuesta sólo es útil para una persona.
- 215 NOMBRE system type.. Donde NOMBRE es un nombre de sistema oficial de la lista que hay en el documento Números Asignados.
- 120 El servicio estará en funcionamiento en nnn minutos.
- 220 Servicio preparado para nuevo usuario.
- 221 Cerrando la conexión de control. Desconectado si procede.
- 421 Servicio no disponible, cerrando la conexión de control. Esta puede ser la respuesta a cualquier comando si el servidor sabe que debe finalizar.
- 125 La conexión de datos ya está abierta; comenzando transferencia.
- 225 Conexión de datos abierta; no hay transferencia en proceso.
- 425 No se puede abrir la conexión de datos.
- 226 Cerrando la conexión de datos. La acción sobre archivo requerida ha sido correcta (por ejemplo, una transferencia o interrupción).
- 426 Conexión cerrada; transferencia interrumpida.
- 227 Iniciando modo pasivo (h1,h2,h3,h4,p1,p2).
- 230 Usuario conectado, continúe.
- 530 No está conectado.
- 331 Usuario OK, necesita contraseña.
- 332 Necesita una cuenta para entrar en el sistema.
- 532 Necesita una cuenta para almacenar archivos.
- 150 Estado del archivo correcto; va a abrirse la conexión de datos.
- 250 La acción sobre archivo solicitado finalizó correctamente.
- 257 "NOMBRERUTA" creada.
- 350 La acción requiere más información

- 450 Acción no realizada. archivo no disponible (por ejemplo, archivo bloqueado).
- 550 Acción no realizada, archivo no disponible (por ejemplo, archivo no existe, no se tiene acceso al mismo).
- 451 Acción interrumpida. Error local.
- 551 Acción interrumpida. Tipo de página desconocido.
- 452 Acción no realizada. Falta de espacio en el sistema de archivos.
- 552 Acción interrumpida. Se ha sobrepasado el espacio disponible de almacenamiento (para el directorio actual).
- 553 Acción no realizada. Nombre de archivo no permitido.

2.5. Escenario Típico del FTP

El usuario del equipo U quiere transferir archivos a/desde el equipo S: Generalmente, el usuario se comunicará con el servidor a través del proceso User-FTP. La figura 8 ilustra un escenario típico. Lo que muestra el use-FTP está entre paréntesis, '---->' representa órdenes de U a S y '<----' representa respuestas de S a U.

```
ORDENES DEL USUARIO                ACCIONES QUE IMPLICAN
ftp (host) multics<CR>             Conectarse a S en el puerto L
                                   estableciendo conexiones de control.
                                   <---- 220 Servicio preparado <CRLF>.
username Doe <CR>                  USER Doe<CRLF>---->
                                   <---- 331 Nombre de usuario OK,
                                             necesita contraseña<CRLF>.
password mumble <CR>               PASS mumble<CRLF>---->
                                   <---- 230 Usuario aceptado<CRLF>.
retrieve (local type) ASCII<CR>
(local pathname) test 1 <CR>       El user-FTP abre el fichero local en
                                   modo ASCII.
(for. pathname) test.pll<CR>       RETR test.pll<CRLF> ---->
                                   <---- 150 Fichero correcto;
                                         se va a abrir la conexión
                                         de datos<CRLF>.
                                   El servidor crea la conexión de datos
                                   al puerto U.
                                   <---- 226 Cerrando conexión de datos,
                                       transferencia completada<CRLF>.
type Image<CR>                     TYPE I<CRLF> ---->
                                   <---- 200 Orden OK<CRLF>
store (local type) image<CR>
(local pathname) file dump<CR>     El user-FTP abre el fichero local en
                                   modo binario.
(for.pathname) >udd>cn>fd<CR>      STOR >udd>cn>fd<CRLF> ---->
                                   <---- 550 Acceso denegado<CRLF>
terminate                          QUIT <CRLF> ---->
                                   El servidor cierra todas las
                                   conexiones.
```

Figura 9. Escenario típico de FTP

3. SERVICIO PROXY

3.1. Generalidades

PROXY es un servicio de redes de computadoras que permite a otros equipos clientes hacer conexiones de red indirectas a otros servicios de red. Un cliente se conecta al servidor PROXY, solicita una conexión, archivo o cualquier otro recurso disponible en otro servidor. El PROXY provee el recurso al cliente, ya sea conectándose directamente al servidor especificado, o recuperándolo de su CACHE. EN algunos casos, el PROXY puede alterar la petición del cliente o la respuesta del servidor para diferentes propósitos.

3.2. Funcionamiento

La aplicación más común de los PROXIES es el Web PROXY. Este provee un caché local de sitios Web y archivos disponibles en servidores Web remotos, permitiendo a los clientes locales accederlos de forma rápida y confiable.

Cuando el Web PROXY recibe una petición de un recurso especificado por un URL, lo busca en su caché local. Si está almacenado, retorna el documento inmediatamente. De lo contrario, lo trae del servidor remoto, guarda una copia en el caché y lo entrega al solicitante. El caché usualmente posee un algoritmo de expiración que elimina sus contenidos de acuerdo a la edad, tamaño e historial de accesos. Dos simples algoritmos son LRU y LFU. El algoritmo LRU *(Least Recently Used)* elimina los archivos menos recientes y el LFU *(Least Frequently Used)* elimina los archivos menos usados.

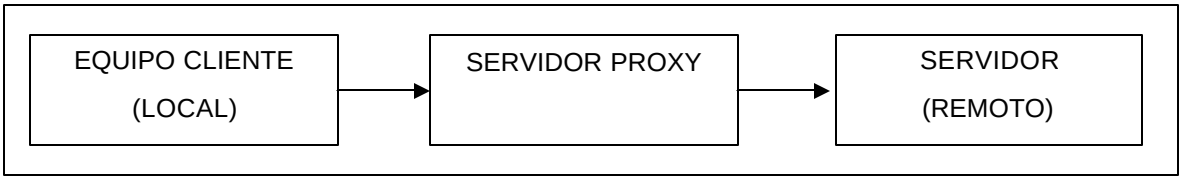

Figura 10. Esquema de funcionamiento de un servidor PROXY.

Los Web PROXIES pueden filtrar el contenido de las páginas Web servidas. También pueden ser instaladas aplicaciones de censura que intentan bloquear el contenido Web ofensivo. Los operadores de red pueden también configurar los PROXIES para interceptar virus de computadores y todo tipo de contenido hostil que esté alojado en otros servidores.

3.3. Caché Web

El Caché Web *(Web Caching)* es el cacheo de recursos Web (páginas, imágenes, etc.) con el fin de reducir el uso del ancho de banda y el tiempo de acceso a los sitios Web. Un Caché Web almacena copias de los recursos solicitados por los usuarios. Peticiones subsecuentes pueden ser completadas desde el caché si se cumplen ciertas condiciones. Los Caché Web generalmente alcanzan metas del 30% al 50% y se vuelven más efectivos cuando la población aumenta.

HTTP es un conjunto de funcionalidades relativamente complejas que pueden ser usadas por los agentes de usuario y los servidores para controlar si los recursos son almacenados en el caché o no, y cuándo las copias del caché pueden ser re-usadas.

Hay dos tipos de Caché Web: del lado del cliente *(client-side)* y del lado del servidor *(server-side)*. Los cachés del lado del cliente se denominan a menudo “cachés normales” y sirven a clientes locales. Son usados por los proveedores de servicios de Internet *(ISP – Internet Service Providers)*, instituciones educativas y corporaciones. Los Cachés del lado del servidor también se conocen como “cachés inversos” o “aceleradores Web” y se ubican delante de los servidores para reducir su carga.

Los principales Sitios Web, que reciben millones de peticiones al día, requieren alguna forma de Caché Web. Algunos ejemplos de Cachés Web son: memcached, Akamai, Squid cache.

Los Cachés Web también realizan tareas relacionadas como la autenticación y el filtro de contenidos.

3.4. NAT Proxies y TransProxies

Los PROXIES también pueden operar a un nivel inferior en la torre de protocolos. La traducción de direcciones de red *(NAT – Network Adress Translation)* es un método para intercambiar conexiones TCP y datagramas UDP mediante el servidor PROXY. NAT también es conocido como "enmascaramiento IP".

Muchas organizaciones usan los servidores PROXY para reforzar sus redes y proveer políticas de uso y seguridad, además de fomentar el uso del caché. Un Web PROXY típico no es transparente para los clientes, éstos deben estar configurados para usar el servidor PROXY, ya sea de forma manual o mediante la ejecución de comando automatizados (SCRIPTS). Un PROXY transparente (TRANSPROXY) combina un servidor PROXY con NAT de manera que las conexiones son enrutadas hacia el PROXY sin necesidad de configuraciones del lado del cliente.

Tanto los NAT como los TRANSPROXY han generado controversia en la comunidad técnica de Internet, pues violan el principio punto a punto *(end-to-end)* sobre el cual fue construido TCP/IP.

3.5. Proxies abiertos.

Tanto los PROXIES de Web como de red son abusados por *"spammers"* y otros usuarios de red malintencionados. Un PROXY abierto es un servidor PROXY que acepta conexiones de cualquier dirección IP y hace conexiones a un recurso de Internet. El abuso de PROXIES abiertos se debe

usualmente a casos de envío de correos no solicitado. Los "*spammers*" usualmente instalan PROXIES abiertos en equipos vulnerables en forma de virus diseñados para este fin.

Debido a que los PROXIES se ven implicados en casos de abuso, los administradores del sistema han desarrollado múltiples métodos para denegar el servicio a PROXIES abiertos. Redes de colaboradores escanean continuamente la red en busca de PROXIES abiertos. Así mismo, otras redes publican la lista de direcciones IP de los PROXIES abiertos conocidos

3.6. Servidores Proxy más populares

- **Squid cache** es un servidor PROXY popular en los sistemas LINUX/UNIX
- El servidor **Apache HTTP** puede ser configurado para actuar como servidor PROXY.
- **Wingate** es un pequeño servidor PROXY que puede usarse para redireccionar todo tipo de tréfico en un equipo WIndows. Otro servidor PROXY gratuito es **AnalogX**.
- El kernel de **LINUX** contiene módulos que proven la funcionalidad NAT como parte de los sistemas de filtrado de paquetes "ipchains" y "iptables"
- **Packet Filter** de OpenBSD es una interfaz muy flexible para inspeccionar el tráfico de redes, la cual puede proveer, entre otras, funcionalidad de NAT
- **Elhttp Server** es un PROXY HTTP liviano para sistemas UNIX y Windows

3.7. Squid Proxy

Squid es el software para servidor Proxy más popular y extendido entre los sistemas operativos basados sobre UNIX. Es muy confiable, robusto y versátil. Al ser software libre, además de estar disponible el código fuente, está libre del pago de licencias por uso o con restricción a un uso con determinado número de usuarios.

Squid puede actuar como PROXY y Caché con los protocolos HTTP, FTP, GOPHER y WAIS, Proxy de SSL, Caché transparente, WWCP, Aceleración HTTP, Caché de consultas DNS y otras como filtración de contenido y control de acceso por IP y por usuario.

Squid no puede funcionar como PROXY para servicios como SMTP, POP3, TELNET, SSH, etc. Si se requiere hacer PROXY para cualquier cosa distinta a HTTP, HTTPS, FTP, GOPHER y WAIS se requerirá implementar enmascaramiento de IP a través de un NAT.

3.7.1. Configuración básica.

Squid utiliza el archivo de configuración **squid.conf**, y requiere de configurar los siguientes parámetros básicos:

- **http_port**

  El servidor PROXY por defecto utilizará el puerto 3128 para atender peticiones, sin embargo se puede especificar que lo haga en cualquier otro puerto. Es importante recordar que los servidores HTTP, como Apache, utilizan el puerto 80 así que no se recomienda utilizarlo como puerto para el servicio PROXY. Regularmente algunos programas utilizados comúnmente por los usuarios suelen traer por defecto el puerto 8080 -servicio de cacheo WWW- para utilizarse al configurar que servidor PROXY utilizar. Para aprovechar esto y evitar configuraciones extras en los usuarios, se puede especificar este puerto para el servicio PROXY.

- **cache_mem**

  El parámetro cache_mem establece la cantidad ideal de memoria para los objetos en tránsito, objetos hot y objetos negativamente almacenados en el caché. Los datos de estos objetos se almacenan en bloques de 4 Kb. El parámetro cache_mem especifica un límite máximo en el tamaño total de bloques acomodados, donde los objetos en tránsito tienen mayor prioridad. Sin embargo los objetos Hot y aquellos negativamente almacenados en el caché podrán utilizar la memoria no utilizada hasta que esta sea requerida. De ser necesario, si un objeto en tránsito es mayor a la cantidad de memoria especificada, el servidor PROXY excederá lo que sea necesario para satisfacer la petición. Por defecto se establecen 8 MB. Puede especificarse una cantidad mayor si así se considera necesario,

dependiendo esto de los hábitos de los usuarios o necesidades establecidas por el administrador. Por ejemplo, si se posee un servidor con al menos 128 MB de RAM, se puede establecer 16 MB como valor para este parámetro.

- **cache_access_log**

  cache_access_log nombra el archivo donde se almacenan los registros de peticiones del servicio PROXY. Cada solicitud HTTP es registrada en este archivo.

- **cache_log**

  cache_log nombra el archivo donde se almacena toda la información general del servicio PROXY.

- **cache_store_log**

  cache_log nombra el archivo donde se registran las actividades del administrador de almacenamiento. Muestra qué objetos son sacados del cache y cuales almacenados y por cuánto tiempo. Para deshabilitarlo escriba 'none'.

- **cache_mgr**

  Dirección de correo electrónico del administrador del servicio PROXY.

- **cache_effective_user**

  Si el servicio PROXY es ejecutado como ROOT, se cambiará su ID efectiva/real a el ID especificada en esta directiva. Por defecto se debe cambiar el ID a 'nobody'. Si no es ejecutado como ROOT, por defecto se mantiene el ID actual.

- **cache_effective_group**

  Si el servicio PROXY es ejecutado como ROOT, se cambiará su ID efectivo/real al ID especificado en esta directiva. Por defecto se debe cambiar el ID a 'nobody'. Si no es ejecutado como ROOT, por defecto se mantiene el ID actual.

- **logfile_rotate**

  Especifica el número de rotaciones a efectuar cuando se ejecute 'squid -k rotate'. El valor por defecto es 10 el cual rotará el archivo con extensiones del 0 al 9. El valor 0 deshabilita

la rotación, pero los archivos de registro se siguen cerrando y re-abriendo. Esto le permitirá renombrar los archivos de registros usted mismo antes de enviar la señal de rotación.

- **error_directory**

  Permite crear sus propias versiones de los archivos de error por defecto (en Inglés), ya sea personalizándolos para ajustarse a su lenguaje u organización. Copie los archivos de la plantilla en inglés a otro directorio y especifique la nueva ruta en esta directiva.

4. SERVICIO RAS

4.1. Introducción

La administración de múltiples líneas seriales y MODEMS dispersos para grandes números de usuarios pueden crear la necesidad de obtener soporte administrativo de forma significante. Los grupos MODEMS son, por definición, un enlace con el mundo exterior. Requieren una cuidadosa atención en lo referente a seguridad, autorizaciones y cuentas de usuario. La mejor forma de lograrlo es a través de la administración de una simple base de datos de usuarios, la cual permite autenticarlos (verificando nombre de usuario y contraseña) y mantener la información de la configuración que detalla el tipo de servicio que será ofrecido al usuario (por ejemplo SLIP, PPP, TELNET, RLOGIN, etc.).

4.2. Características

4.2.1. Modelo Cliente/ Servidor

Los servidores de acceso remoto *(RAS – Remote Access Servers)* operan como clientes del servicio RADIUS. El cliente es responsable de pasar la información del usuario a los servidores específicos de RADIUS, y actuar de acuerdo a la respuesta obtenida. Los servidores RADIUS son responsables de recibir peticiones de conexión del usuario, autenticarlo, y retornar toda la información de configuración necesaria al cliente para entregarle el servicio. Un servidor RADIUS puede actuar como cliente apoderado (PROXY) de otros servidores RADIUS y de otras clases de servidores de autenticación.

4.2.2. Seguridad de red

Las transacciones entre el cliente y el servidor RADIUS son autenticadas por medio del uso de un "secreto compartido", el cuál nunca se envía a través de la red. Además, cualquier contraseña de usuario se envía cifrada entre el cliente y el servidor RADIUS para eliminar la posibilidad de que alguien que monitoree la red pueda determinar la contraseña de un usuario.

4.2.3. Mecanismos flexibles de autenticación

El servidor RADIUS puede soportar una variedad de métodos para autenticar a un usuario. Cuando el usuario proporciona el nombre del usuario y la contraseña original, puede proveer mecanismos de autenticación como PPP PAP, CHAP y UNIX LOGIN entre otros.

4.3. Terminología

- **Servicio:** El Servidor de Acceso Remoto (RAS – Remote Access Server) provee un servicio al usuario llamante, tal como el protocolo punto a punto *(PPP – Point To Point Protocol)* o TELNET.
- **Sesión:** Cada servicio que provee el RAS al usuario llamante constituye una sesión. El comienzo de la sesión se define como el momento inicial donde se proporciona el servicio. El final de sesión se define como el punto donde se termina el servicio. Un usuario puede tener varias sesiones en serie o en paralelo si el RAS se encuentra configurado así.
- **Descarte silencioso:** Esto significa que la implementación descarta el paquete sin procesarlo más allá. La implementación debe proveer la capacidad de registrar el error, incluyendo el contenido del paquete descartado, y debe a su vez registrar el evento en un contador de estadísticas.

4.4. Operación

Cuando un cliente es configurado para usar RADIUS, sus usuarios le presentan la información de autenticación. Esto puede ser a través de una pantalla de bienvenida donde el usuario ingresa su identificación y contraseña. Alternativamente, el usuario puede usar un enlace enmarcado en un

protocolo, tal como el protocolo punto-a-punto (PPP), el cual tiene paquetes de autenticación que llevan esta información. Una vez el cliente ha obtenido dicha información, puede ser autenticado por RADIUS. Para esto, el cliente crea una "petición de acceso" que contiene el nombre de usuario, la contraseña, el ID del cliente y el ID del puerto a través del cual está accediendo. Cuando existe una contraseña, ésta se oculta mediante algún método de encriptación.

La petición de acceso es remitida al servidor de RADIUS a través de la red. Si no hay una respuesta en un intervalo definido de tiempo, la petición es re-enviada un número de veces igualmente definido. El usuario puede también enviar peticiones a uno o varios servidores alternativos en caso de que el servidor primario este sin servicio o fuera de alcance. El servidor alternativo puede ser usado después de un determinado número de intentos fallidos con el servidor primario, o con una base de datos de búsqueda cíclica. Una vez el servidor de RADIUS recibe la petición, este valida al cliente que la envía. Una petición de un cliente con el cual RADIUS no tiene un secreto compartido, debe ser descartada silenciosamente. Si el cliente es valido, el servidor RADIUS consulta una base de datos de usuarios para encontrar aquel usuario cuyo nombre coincide con el de la petición. El registro del usuario en la base de datos contiene una lista de requerimientos los cuales deben ser diligenciados para permitirle el ingreso. Esto siempre incluye la verificación de la contraseña, pero puede también especificar el (los) cliente(s) o puerto(s) a los cuales tiene acceso. El servidor de RADIUS puede también consultar otros servidores para satisfacer la petición, en cuyo caso actúa como cliente.

Si no se cumple con las condiciones, el servidor RADIUS envía una respuesta de "rechazo de acceso" indicando que la petición de este usuario es invalida. El servidor puede también incluir un mensaje de texto en el rechazo de acceso el cual puede ser mostrado por el cliente al usuario.

Si todas las condiciones se cumplen y el servidor RADIUS desea emitir una prueba que el usuario debe responder, el servidor RADIUS envía una respuesta de "prueba de acceso". Esta puede incluir un mensaje de texto para ser mostrado al usuario solicitando una respuesta a la prueba, y puede incluir también el atributo de estado.

Si el cliente recibe una prueba de acceso y tiene habilitada la configuración de prueba/respuesta, puede mostrar el mensaje de texto al usuario (si existiera) y luego solicitarle una respuesta. El cliente luego re-envía la petición de acceso original con un nuevo ID de petición, reemplazando el

atributo de usuario-contraseña con la respuesta (encriptada) e incluyendo el atributo de estado de la prueba de acceso, si existiera. Solo se permiten las instancias 0 ó 1 para el atributo de estado en una petición. El servidor puede responder a esta nueva petición de acceso, ya sea con una aceptación de acceso, un rechazo de acceso u otro tipo de prueba de acceso.

Si todas las condiciones se cumplen, la lista de valores de configuración para el usuario es puesta dentro de una respuesta de "aceptación de acceso". Estos valores incluyen el tipo de servicio y aquellos valores necesarios para entregar el servicio deseado. En el caso de que el servicio sea el Protocolo de Internet para Línea Serial *(SLIP – Serial Line Interface Protocol)* ó PPP, se pueden incluir valores como la dirección IP, mascara de subred, máxima unidad de transferencia *(MTU – Maximum Transfer Unit)*, compresión deseada y los identificadores de filtro de paquetes.

4.5. Formato de paquetes

Un paquete de RADIUS se encapsula en un paquete de protocolo de data-grama de usuario (UDP – User Datagram Protocol) donde el puerto de destino es el 1812. Cuando una respuesta es generada, los puertos fuente y destino son invertidos.

Un resumen del formato de datos de RADIUS se muestra a continuación. Los campos son transmitidos de izquierda a derecha.

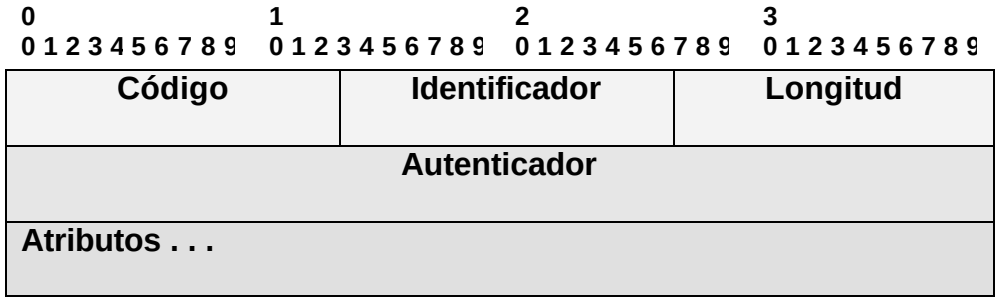

Figura 11. Formato de datos de RADIUS

4.5.1. Código

El campo de código es un octeto e identifica el tipo de paquete RADIUS. Cuando un paquete es recibido con un campo de código invalido, este es silenciosamente descartado.

Los códigos de RADIUS son especificados así:

1 Access-Request
2 Access-Accept
3 Access-Reject
4 Accounting-Request
5 Accounting-Response
11 Access-Challenge
12 Status-Server (experimental)
13 Status-Client (experimental)
255 Reserved

Los códigos 12 y 13 están reservados para un posible uso.

4.5.2. Identificador

El campo identificador es un octeto y ayuda a comparar las peticiones y las respuestas. El servidor RADIUS puede detectar una petición duplicada si ésta tiene como fuente la misma dirección IP, puerto de origen UDP o identificador de tiempo.

4.5.3. Longitud

El campo de longitud abarca dos octetos. Este indica la longitud del paquete incluyendo el código, identificador, longitud, autenticador y atributos. Los octetos fuera del rango del campo de longitud deben ser ignorados en la recepción. Si el paquete es mas pequeño de lo indicado en el campo de longitud, este debe ser descartado silenciosamente. La longitud mínima del paquete es de 20 Bytes y la máxima de 4096 Bytes.

4.5.4. Autenticador

El campo autenticador abarca 16 octetos. El octeto más significante es transmitido en primer lugar. Este valor es usado para autenticar la respuesta desde el servidor de RADIUS y es usado en el algoritmo para ocultar la contraseña.

4.5.4.1. Autenticador de petición

En los paquetes de petición de acceso, el valor del campo del autenticador es un numero aleatorio de 16 octetos llamado "autenticador de petición", el valor debe ser único e impredecible a lo largo del tiempo de vida de un secreto (la contraseña compartida entre el cliente y el servidor RADIUS), ya que la repetición de un valor de petición, junto con el secreto pueden permitir a un atacante responder con un mensaje previamente interceptado. Ya que se espera que el mismo secreto pueda sea usado para autenticarse en servidores ubicados en distintas ubicaciones geográficas, el campo de autenticación de petición debe ser único global y temporalmente.

El valor de autenticador de petición en un paquete de petición de acceso también debe ser impredecible, evitando que un atacante engañe al servidor con una respuesta predicha y después la use para enmascarar futuras peticiones de acceso a este servidor.

Los protocolos usados en RADIUS no protegen las sesiones autenticadas de ser escuchadas mediante ataques activos en tiempo real sobre la línea telefónica, pero esto puede ser evitado mediante la generación de peticiones únicas e impredecibles.

4.5.4.2. Autenticador de respuesta

El valor del campo del autenticador en los paquetes de aceptación de acceso, rechazo de acceso y prueba de acceso se conoce como "autenticador de respuesta" y contiene una llave de encripción calculada sobre una cadena de octetos que consta de: el paquete RADIUS, el comienzo del campo de código, el identificador, la longitud, el campo autenticador de petición del paquete de petición de acceso y los atributos de respuesta, seguidos por el secreto compartido.

4.5.5. Nota administrativa

El secreto (Contraseña compartida entre el cliente y el servidor de RADIUS) debe ser bien escogido así como tener una longitud adecuada e igualmente insospechable. Es preferible que la contraseña sea de al menos 16 octetos, esto con el fin de asegurar un rango lo suficientemente largo al secreto para darle de protección contra ataques que intenten descifrarlo. El secreto no debe estar vacío (longitud cero) ya que podría permitir que los paquetes sean forzados. Un servidor de RADIUS debe usar la dirección IP origen del paquete UDP RADIUS para decidir qué secreto compartido usar.

4.6. Tipos de paquetes

El tipo de paquetes de RADIUS esta determinado por el campo de código en el primer octeto del paquete.

4.6.1. Petición de acceso.

Los paquetes de petición de acceso son enviados a un servidor RADIUS y llevan información que permite determinar si un usuario esta autorizado para acceder a un RAS especifico y qué servicio especial requiere. Una implementación que desee autenticar a un usuario debe transmitir un paquete RADIUS con el campo de código en valor “1” (petición de acceso). Así mismo, al recibir una petición de acceso de un cliente válido, se debe transmitir la respuesta apropiada.

Una petición de acceso debe contener el atributo “nombre de usuario”. Este debe contener un atributo de dirección IP-RAS o un atributo de identificador del NAS (o ambos). Una petición de acceso debe contener una contraseña de usuario, una contraseña CHAP o un estado. Una petición de acceso no debe contener una contraseña de usuario y una contraseña CHAP al mismo tiempo. Una petición de acceso debe contener un puerto NAS, un tipo de atributo de puerto NAS, o ambos; a menos que el tipo de acceso solicitado no implique un puerto o el NAS o no distinga entre sus puertos.

Con base en la Figura 11, se define el formato de los paquetes de petición de acceso

- **Código:** Debe ser 1 para la petición de acceso.
- **Identificador:** El campo de identificador debe ser cambiado cada vez que el contenido del campo de atributos cambie y siempre que una respuesta válida sea recibida de una petición previa. Para el caso de re-transmisiones, el identificador debe mantenerse.
- **Autenticador de petición:** El valor del autenticador de petición debe cambiar cada vez que un nuevo identificador es usado.
- **Atributos:** El campo de atributos es variable en longitud, y contiene la lista de atributos que son requeridos para el tipo de servicio a ofrecer, así como cualquier otro atributo opcional.

4.6.2. Aceptación de acceso

Los paquetes de aceptación de acceso son enviados a través del servidor RADIUS y proveen la información de la configuración específica necesaria para entregar el servicio al usuario. Si todos los valores recibidos en una petición de acceso son válidos, entonces la implementación del RADIUS debe transmitir un paquete con el valor "2" en el campo de código (aceptación de acceso).

AL recibir una aceptación de acceso, el campo del identificador se compara con una petición de acceso pendiente. El campo de autenticador de respuesta debe contener la respuesta correcta para la petición de acceso pendiente. Los paquetes inválidos son silenciosamente descartados.

Con base en la Figura 11, se define el formato de los paquetes de aceptación de petición

- **Código:** Debe ser 2 para aceptación de acceso.
- **Identificador:** El campo de identificador es una copia del campo de identificador de la petición de acceso que causó esta aceptación de acceso.
- **Autenticador de respuesta:** El valor del campo del autenticador de respuesta es calculado a partir del valor del campo en la petición de acceso.
- **Atributos:** El campo de atributos en variable en longitud y contiene una lista de cero o mas atributos.

4.6.3. Rechazo de acceso

Si cualquier valor de los atributos recibidos no es aceptable, entonces el servidor de RADIUS debe transmitir un paquete con el valor "2" en el campo de código (rechazo de acceso). Este puede incluir uno o más atributos de mensajes de respuesta con un mensaje de texto que el RAS puede mostrar al usuario.

Con base en la Figura 11, se define el formato de los paquetes de rechazo de acceso

- **Código:** Debe colocarse en 3 para rechazo de acceso.
- **Identificador:** El campo de identificador es una copia del mismo campo en la petición de acceso que causó este rechazo de acceso.
- **Autenticador de respuesta:** El valor del autenticador de respuesta es calculado a partir del valor de la petición de acceso.
- **Atributos:** El campo de atributos es variable en longitud y contiene una lista de cero o más atributos.

4.6.4. Prueba de acceso

Si el servidor de RADIUS desea enviar al usuario una prueba que requiera de una respuesta, este servidor debe responder a la petición de acceso transmitiendo un paquete con el valor "11" en el campo de código (prueba de acceso). El campo de atributos puede tener uno o más atributos de mensajes de respuesta, y puede tener un atributo de estado simple o ninguno.

Al recibir una prueba de acceso, el campo de identificador debe coincidir con una petición de acceso pendiente. Adicionalmente, el campo de autenticador de respuesta debe contener la respuesta correcta para la petición de acceso pendiente. Los paquetes inválidos son silenciosamente descartados.

Si el RAS no soporta la configuración de pruebas/respuestas, debe intentar una prueba de acceso como si hubiera recibido un rechazo de acceso a cambio. Si el RAS soporta pruebas/respuestas, la recepción de una prueba de acceso indica que una nueva petición de acceso debe ser enviada. El RAS puede mostrar un mensaje de texto al usuario y pedirle una respuesta. Luego este envía su

petición de acceso original con una nueva ID de solicitud y un nuevo autenticador de petición con el atributo de contraseña reemplazado por la respuesta del usuario (encriptada) e incluyendo el atributo de estado de la requisición de acceso, si existiera. Las instancias 0 ó 1 del atributo de estado pueden estar presentes en una solicitud de acceso.

Con base en la Figura 11, se define el formato de los paquetes de prueba de acceso.

- **Código:** El valor debe ser 11 para prueba de acceso.
- **Identificador:** El campo de identificador es una copia del campo identificador de la petición de acceso la cual causa esta prueba de acceso.
- **Autenticador de respuesta:** El valor del autenticador de respuesta es calculado en base al valor de la petición de acceso.
- **Atributos:** El campo de atributos es variable en longitud y contiene una lista de cero o más atributos.

4.7. Atributos.

Los atributos de RADIUS transportan la autenticación específica, autorización, información y detalles de configuración para la petición y la respuesta. El final de la lista de atributos esta indicada por la longitud del paquete de RADIUS. Si múltiples atributos del mismo tipo se presentan, el orden de atributos debe ser conservado. El orden de los atributos de diferentes tipos no requiere ser conservado. Un cliente o servidor RADIUS no debe depender del orden de los atributos de diferentes tipos.

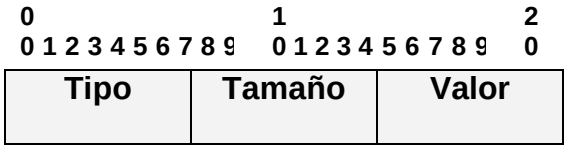

Figura 12. Resumen del formato de los atributos

4.7.1. Resumen de atributos

1 User-Name
2 User-Password
3 CHAP-Password
4 NAS-IP-Address
5 NAS-Port
6 Service-Type
7 Framed-Protocol
8 Framed-IP-Address
9 Framed-IP-Netmask
10 Framed-Routing
11 Filter-Id
12 Framed-MTU
13 Framed-Compression
14 Login-IP-Host
15 Login-Service
16 Login-TCP-Port
17 (unassigned)
18 Reply-Message
19 Callback-Number
20 Callback-Id
21 (unassigned)
22 Framed-Route
23 Framed-IPX-Network
24 State
25 Class
26 Vendor-Specific
27 Session-Timeout
28 Idle-Timeout
29 Termination-Action
30 Called-Station-Id
31 Calling-Station-Id
32 NAS-Identifier
33 PROXY-State
34 Login-LAT-Service
35 Login-LAT-Node
36 Login-LAT-Group
37 Framed-AppleTalk-Link

38 Framed-AppleTalk-Network
39 Framed-AppleTalk-Zone
40- (reserved for accounting)
60 CHAP-Challenge
61 NAS-Port-Type
62 Port-Limit
63 Login-LAT-Port

4.7.2. Tipo

El campo tipo es un octeto. Los valores 192 a 223 están reservados para uso experimental. Los valores 224 a 240 están reservados para usos específicos de implementación. Los valores 241 a 255 están reservados y no se deben usar.

Un servidor RADIUS puede ignorar atributos de tipo desconocido. Un cliente RADIUS puede ignorar atributos de tipo desconocido.

4.7.3. Tamaño

El campo tamaño es un octeto e indica la longitud de este atributo incluyendo los campos Tipo, Tamaño y Valor. Si un atributo es recibido en una petición de acceso pero con un tamaño inválido, se transmite un rechazo de acceso. Si un atributo es recibido en un paquete de aceptación de acceso, rechazo de acceso o prueba de acceso con un tamaño inválido, el paquete será tratado como un rechazo de acceso o es descartado silenciosamente.

4.7.4. Valor

El campo valor puede ser de cero o más octetos y contiene información específica para el atributo. El formato y tamaño del campo valor es determinado por los campos Tipo y Tamaño.

5. SERVICIO SMTP

5.1. Introducción

El protocolo de transferencia simple de correo electrónico *(SMTP – Simple Mail Transfer Protocol)* permite enviar correo electrónico. Los mensajes salientes utilizan SMTP para pasar de la máquina del cliente al servidor, lugar desde el que se trasladan hasta el destino final. También dos servidores de correo que intentan transferir entre sí un mensaje utilizan SMTP para comunicarse, incluso si utilizan plataformas totalmente distintas.

5.2. Funcionamiento

SMTP usa el puerto 25 del servidor para comunicarse. Empieza un intercambio SMTP básico con el sistema conectado mediante la emisión del comando "MAIL From: <dirección de correo electrónico>" para iniciar el intercambio. El sistema que recibe el comando responde con un mensaje 250 para informar de que se ha recibido el primer comando. A continuación, el sistema conectado comunica las direcciones de correo electrónico para recibir el mensaje del sistema receptor, seguido de un mensaje con el comando "DATA". Este mensaje notifica al sistema receptor que la siguiente parte de la comunicación será el cuerpo real del mensaje de correo electrónico. Cuando el sistema conectado finaliza de enviar el mensaje de correo electrónico, coloca un punto sencillo (.) en una línea. A partir de ese momento, se considera que el mensaje se ha enviado.

El protocolo SMTP también permite gestionar el reenvío de mensajes entre sistemas si el sistema receptor sabe el destino al que tiene que enviar el mensaje. El protocolo puede verificar si determinados usuarios utilizan realmente un servidor de correo concreto (comando VRFY) o

ampliar una lista de distribución de correo (comando EXPN). También se puede retrasar el envío de correo electrónico entre dos servidores SMTP si en los dos sistemas se permite realizar esta actividad.

A diferencia de otros protocolos, SMTP no requiere autenticación en su forma más básica. Esto ha provocado mucho correo basura (spam), ya que un usuario no local puede utilizar el sistema de otro para enviar o transmitir el correo a listas completas de destinatarios con los recursos y ancho de banda del sistema. Las aplicaciones SMTP modernas han progresado enormemente al minimizar este comportamiento y restringir las transmisiones de modo que sólo los equipos conocidos envíen correo electrónico.

En las peticiones de comentarios 821 *(RFC – Request For Comments)* se describe el comportamiento básico del protocolo SMTP, aunque varias extensiones de SMTP, posibles gracias a RFC 1869, han agregado nuevas funciones al SMTP a lo largo de los años con nuevos comandos. Al iniciar una conversación con un servidor SMTP mediante un comando EHLO, en lugar de HELO, el servidor conectado puede identificarse a sí mismo como un servidor compatible con las extensiones SMTP. El servidor receptor contesta con una línea 250 que contiene las distintas extensiones SMPT compatibles. A continuación, el servidor conectado puede utilizar las extensiones compatibles como desee para obtener los objetivos de la comunicación.

Una de las extensiones que vale la pena destacar está relacionada con la incorporación de autenticación SMTP mediante el comando AUTH, tal y como se describe en el documento RFC 2554. Otra extensión SMTP muy utilizada se explica en detalle en el documento RFC 2034, que describe el uso entre aplicaciones SMTP de códigos de error estándar separados por puntos.

5.3. Software de correo electrónico

Hay tres tipos de programas de correo electrónico y todos ellos desempeñan un papel específico en el proceso de trasmitir y administrar mensajes de correo electrónico. Aunque la mayoría de los usuarios sólo conocen el programa de correo electrónico con el sistema para enviar y recibir

mensajes, cada uno de los tipos de programas siguientes es también importante para garantizar que los mensajes llegan al destino correcto.

5.3.1. Agente de usuario de correo

El agente de usuario de correo *(MUA – Mail User Agent)* es un programa que permite a un usuario, leer y escribir mensajes de correo electrónico. A un MUA se le denomina a menudo "cliente de correo". Lógicamente, hay muchos programas MUA que ofrecen al usuario muchas más funciones, entre las que se incluyen la recuperación de mensajes, la configuración de buzones de correo para almacenar los mensajes o ayuda para presentar los mensajes nuevos a un programa Agente de transferencia de correo *(MTA – Mail Transfer Agent)* que los enviará al destino final.

Los programas MUA pueden ser gráficos, como "Mozilla Mail", o pueden tener una interfaz basada en texto sencilla, como "Mutt" o "Pine".

5.3.2. Agente de transferencia de correo

Un programa agente de transferencia de correo *(MTA – Mail Transfer Agent)* transfiere los mensajes de correo electrónico entre máquinas que usan el protocolo SMTP. Un mensaje puede pasar por varios MTA hasta llegar al destino final. La mayoría de los usuarios desconocen la existencia de estos agentes, incluso si cada mensaje se envía a través de como mínimo un MTA.

Aunque el proceso de envío de mensajes entre las máquinas puede parecer bastante directo, todo el proceso de decidir si un agente MTA concreto puede o debe aceptar un mensaje para entregarlo a un equipo remoto es bastante complicado. Además, debido a los problemas de correo basura, el uso de un MTA concreto normalmente está limitado por la propia configuración del MTA o el acceso a la red del sistema que lo ejecuta.

Muchos de los agentes MUA de mayores dimensiones y complejidad también sirven para enviar correo. Sin embargo, no se debe confundir esta acción con las funciones propias y verdaderas de estos agentes. Para que los usuarios que no ejecutan un agente MTA propio puedan transmitir los mensajes salientes a una máquina remota para su envío, deben utilizar una capacidad en el MUA capaz de transferir el mensaje a un MTA para el que tengan autorización de uso. Sin embargo, el

agente MUA no entrega directamente el mensaje al servidor de correo del destinatario final; esta función está reservada al agente MTA.

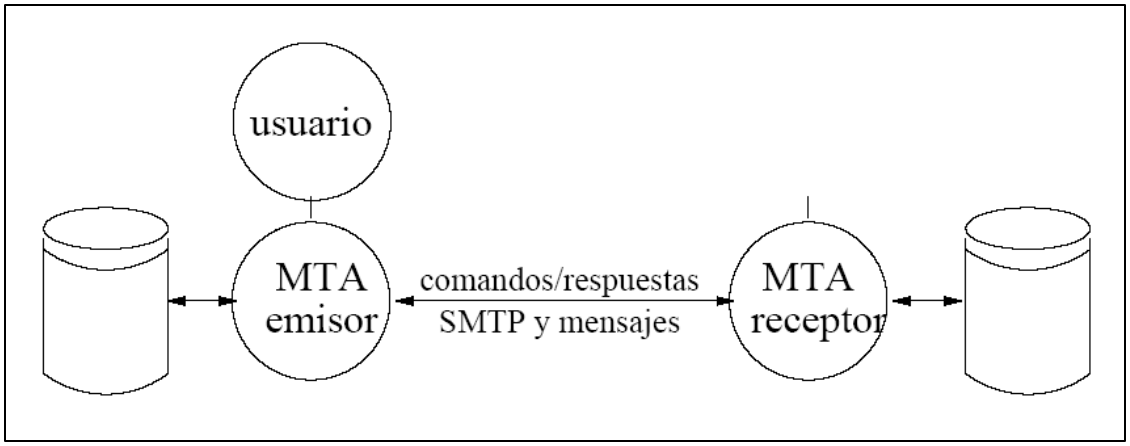

Figura 13. Modelo de SMTP

5.3.3. Sendmail

Muchas distribuciones de LINUX usan "Sendmail" como agente MTA para la entrega de los mensajes, ya estén destinados a usuarios del mismo sistema o a usuarios ubicados en destinos remotos. La mayoría de los administradores de sistema deciden usar Sendmail como agente MTA por su eficacia, escalabilidad y cumplimiento con los estándares de Internet más importantes, como el protocolo SMTP.

La función principal de Sendmail, al igual que la de otros agentes MTA, es transferir de forma segura los correos electrónicos entre los equipos, normalmente mediante el uso del protocolo SMTP. Sin embargo, Sendmail es altamente configurable y permite controlar casi cada aspecto de cómo se gestiona el correo, incluido el protocolo que se va a utilizar.

5.3.3.1. Historia

Las raíces de Sendmail se remontan al nacimiento del correo electrónico, una década antes de que naciese la red de investigación avanzada de proyectos *(ARPANET – Advanced Research Projects Network)*, el precursor de Internet. Por entonces, cada buzón de usuario era un archivo con derechos de sólo lectura y las aplicaciones de correo eran simplemente texto incorporado en ese archivo. La primera transferencia real de un archivo de mensaje de correo entre dos equipos no tuvo lugar hasta 1972, año en el que el correo electrónico empezó a transferirse por FTP a través del protocolo de control red *(NCP – Network Control Protocol).* Este método de comunicación más sencillo muy pronto se hizo popular, incluso hasta el punto de representar la mayor parte del tráfico de ARPANET en menos de un año.

Sin embargo, la falta de estándares entre los protocolos existentes convirtió al correo electrónico en más difícil de enviar desde algunos sistemas y así continuó hasta que ARPANET creó el estándar TCP/IP en 1982. Un nuevo protocolo, SMTP, se materializaba en el transporte de mensajes. Estos avances, en combinación con la sustitución de los archivos, permitieron que se materializasen los agentes MTA con funciones completas. Sendmail, que creció a partir de un precedente sistema de entrega de correo electrónico denominado "Delivermail", muy pronto se convirtió en estándar a medida que Internet comenzaba a expandirse y a utilizarse más ampliamente.

5.4. Ejemplo de envío de un correo electrónico utilizando SMTP

Alice, alice@alfa.proveedor.com, desea enviar un mensaje a Robert, robert@beta.proveedor.com.

1) Alice redacta el mensaje con su Agente de Usuario de Correo, (MUA). Especifica el destinatario en el campo To, el asunto del mensaje en el campo Subject, y luego escribe el texto del mensaje propiamente dicho

   To: Robert@beta
   Subject: comida

¿Te apetece una pizza?

2) Llegados a este punto, el MTA puede añadir campos de cabecera adicionales como la fecha (Date) y el identificador de mensaje (Message-Id) y modificar los valores que Alice ha introducido: por ejemplo, puede reemplazar Robert@beta por «Robert <Robert@beta.proveedor.com>». Después, el MUA inyecta el mensaje en el sistema de correo. Hay dos maneras de hacerlo: o bien ejecutar un programa que proporcione el sistema de correo con la finalidad de inyectar mensajes, o bien abrir una conexión al protocolo simple de transferencia de correo (SMTP), bien en el sistema local o en el servidor de correo remoto. Para este caso, el MUA utilizará un programa local de inyección para pasarle mensajes al agente de transporte de correo (MTA). Los detalles de los procesos de inyección varían según el MTA, pero en los sistemas UNIX el estándar de hecho es el método del "Sendmail". Con este método, el agente de usuario de correo (MUA) puede poner el encabezado y el cuerpo en un archivo, separados por una línea en blanco y pasar el archivo al programa "Sendmail".

3) Si la inyección tiene éxito, el mensaje es ya responsabilidad del agente de transporte de correo (MTA). Los detalles varían en gran medida según el MTA, pero generalmente el MTA de la máquina alfa examina el encabezado para determinar a dónde enviar el mensaje; luego abre una conexión SMTP a la máquina beta, y luego le pasa el mensaje al MTA del sistema beta. El diálogo SMTP precisa que los mensajes se envíen en dos partes: el «sobre» o «envoltorio», que especifica la dirección del destinatario (Robert@beta.proveedor.com) y la dirección de retorno (Alice@alpha.proveedor.com), y el mensaje propiamente dicho, que está compuesto por la cabecera y el cuerpo.

4) Si el MTA de la máquina beta rechaza el mensaje, por ejemplo por no existir un usuario Robert en el sistema, el agente de transporte de correo (MTA) de la máquina alfa envía un mensaje de «devolución» (del inglés bounce) a la dirección de retorno, Alice@alfa, para notificarle el problema.

5) Si el agente de transporte de correo de beta acepta el mensaje, entonces examina la dirección del destinatario, y determina si es una dirección local de beta o está en un sistema remoto. En este caso es local, de manera que el MTA entrega el mensaje él mismo o lo pasa a un agente de entrega de correo.

6) Si falla la entrega, quizás porque Robert se ha excedido en su cuota de correo, el MTA de la máquina beta envía un mensaje de devolución a la dirección de retorno que figura en el «sobre», Alice@alfa.

7) Si la entrega tiene éxito, el mensaje aguardará en el buzón de Bob hasta que su agente de usuario de correo (MUA) lo lea y lo muestre.