

Agentes de transmisión de correo

Trabajo de grado de la Licenciatura en Informática



BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.





Universidad Nacional de La Plata (UNLP)

Facultad de Ciencias Exactas

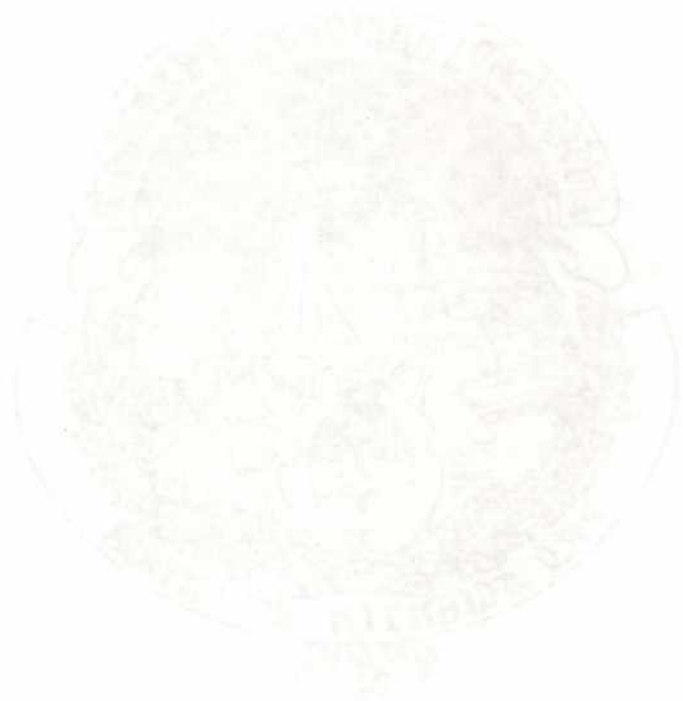
Miguel Angel Luengo

Juan Carlos Marra

Director: Licenciado Francisco Javier Díaz

<p>TES 95/1 DIF-01906 SALA</p>	<p> UNIVERSIDAD NACIONAL DE LA PLATA FACULTAD DE INFORMÁTICA Biblioteca 50 y 120 La Plata catalogo.info.unlp.edu.ar biblioteca@info.unlp.edu.ar</p> <p> DIF-01906</p>
--	---

44



DONACION.....

TES
95/1

\$.....

Fecha..... 16/8/05

Inv. E..... Inv. B..... 1906



Indice

PREFACIO	5
OBJETIVO	7
ALCANCE DEL TRABAJO PROPUESTO	9
ASPECTOS CONCEPTUALES.....	13
EL CORREO ELECTRÓNICO (E-MAIL).....	13
MODELO PARA EL MANEJO DE MENSAJES.....	13
<i>Descripción del modelo</i>	14
<i>Formato de los mensajes</i>	15
EL U.A. (AGENTE USUARIO).....	16
<i>Funcionalidad del U.A.</i>	16
MEDIOS DE TRANSPORTE	18
INTRODUCCIÓN	18
ARQUITECTURA DE REDES.....	18
MODELO DE REFERENCIA OSI	20
EL INTERNET SUITE.....	22
UUCP (UNIX TO UNIX COPY).....	24
DIRECCIONES.....	25
DIRECCIONES TCP/IP.....	25
<i>Subredes (RFC-877)</i>	27
<i>Nombres y direcciones TCP/IP</i>	28
<i>Asignación de nombres</i>	28
<i>DNS (Domain Name Service)</i>	30
DIRECCIONES UUCP.....	31
ESTRUCTURA DE UN MENSAJE.....	32
RFC 822.....	32
<i>Definición Formal</i>	32
<i>Estructura de los valores de los headers</i>	33
MIME (MULTIPURPOSE INTERNET MAIL EXTENTIONS - RFC 1521).....	34
TRANSPORTE DE MAIL.....	38
SIMPLE MAIL TRANSFER PROTOCOL (SMTP) (RFC-821).....	39
<i>Interacciones del protocolo</i>	40
EXTENDED SIMPLE MAIL TRANSFER PROTOCOL. (ESMTP) (RFC-1651).....	41
<i>Servicio de Extensión SMTP para transporte 8bit-MIME</i>	42
<i>Delivery Status Notifications (DSN)</i>	43
<i>Servicio de Extensión SMTP para transmisión de Mensajes Extensos y Mensajes Binarios</i>	46
UUCP	48
TRANSPORTE DE ARTÍCULOS NEWS.....	49
<i>NNTP (Network News Transfer Protocol)</i>	50
SERVICIO DE MAILBOX.....	52
POST OFFICE PROTOCOL (POP).....	52
<i>Funcionamiento básico del POP</i>	53
<i>El modelo Split-U.A.</i>	53

<i>Criterio de diseño</i>	55
INTERFAZ WINDOWS SOCKETS	56
INTRODUCCIÓN	56
CONCEPTOS BÁSICOS	56
MODELO CLIENTE-SERVIDOR	57
IMPLEMENTACIÓN	61
POP3	61
<i>DoUSER()</i>	61
<i>DoPASS()</i>	62
<i>DoSTAT()</i>	63
<i>DoLIST()</i>	64
<i>DoRETR()</i>	65
<i>DoLAST()</i>	66
<i>DoDELETE()</i>	67
<i>DoNOOP()</i>	68
<i>DoRESET()</i>	69
<i>DoTOP()</i>	70
<i>DoQUIT()</i>	71
<i>DoGREET()</i>	72
SMTP/ESMTP	73
<i>DoHELO()</i>	73
<i>DoEHLO()</i>	74
<i>DoMAIL()</i>	75
<i>DoRCPT()</i>	76
<i>DoNOOP()</i>	77
<i>DoRESET()</i>	78
<i>DoVRFY()</i>	79
<i>DoDATA()</i>	80
<i>DoQUIT()</i>	81
<i>DoEXPN()</i>	82
NNTP	83
<i>DoARTICLE()</i>	83
<i>DoGROUP()</i>	84
<i>DoHELP()</i>	85
<i>DoIHave()</i>	86
<i>DoLAST() / DoNEXT()</i>	87
<i>DoLIST()</i>	88
<i>DoNEWGROUPS()</i>	89
<i>DoNEWNEWS()</i>	90
<i>DoSLAVE()</i>	91
<i>DoPOST()</i>	92
<i>DoQUIT()</i>	93
INTERFAZ CON LOS PROTOCOLOS DE TRANSPORTE	95
RECUPERACIÓN DE MENSAJES	96
ENVÍO DE MAIL	96
FUNCIONES PROPUESTAS	97
CONCLUSIÓN	105
ANEXO A	109
GRAMÁTICA BACKUS NAUR FORM (BNF)	109
DEFINICIÓN FORMAL DE UN MENSAJE (SEGÚN RFC 822)	110
MIME (RFC 1521)	112

ANEXO B.....	117
SMTP COMANDOS Y CÓDIGOS DE RESPUESTAS	117
POP. COMANDOS Y CÓDIGOS DE RESPUESTAS.....	119
GLOSARIO.....	121
BIBLIOGRAFIA:	123

Prefacio

Las comunicaciones de datos entre computadoras han evolucionado internacionalmente y la incorporación de la Universidad Nacional de La Plata a la red Internet abre numerosas posibilidades de actualización tecnológica, a través de los distintos servicios que esta brinda. En particular el que a nosotros nos interesa para desarrollar el proyecto es el servicio de mail o correo electrónico.

Si bien el correo electrónico es el medio más elemental que un servicio de red puede ofrecer, es también el que más difusión ha tenido, ya que ha permitido la comunicación entre “mundos” distintos, ya sea a nivel de hardware y a nivel de sistemas operativos, alrededor de todo el mundo.

Otro punto a favor del correo electrónico, es que actualmente existen posibilidades de utilizar información en diversos formatos (binarios, gráficos, animaciones, sonido, fax, etc.). Además el intercambio de mail entre distintos protocolos de transporte (como lo es SMTP sobre TCP/IP o bien X.400 sobre ISO/OSI), permiten mayores posibilidades de uso y extiende el campo de acción, ya sea a nivel académico, comercial, militar, etc.

Por otra parte la aparición de nuevos sistemas operativos, que entre otras cosas, se presentan con un entorno mucho más amigable al usuario final, permitió de alguna manera definir nuevas interfaces de usuarios para los sistemas de mail, cooperando aún más a su difusión. También nuevas tecnologías en los medios de comunicación permitieron enlaces más rápidos y seguros para mayores volúmenes de información.

Sin embargo todo este desarrollo tecnológico, que brinda una amplia gama de servicios que hasta no hace mucho tiempo atrás estaba fuera del alcance del correo electrónico, hubo que incorporarlo a lo que ya existía, además no se debe olvidar los nuevos niveles de seguridad y control necesarios con dicho avance, provocando una avalancha de definiciones de extensiones a los ya existentes.

Dicho de otra manera, el correo electrónico desde el punto de vista conceptual es un servicio tradicional, pero su implementación no lo es tanto, ya que es un servicio que evoluciona constantemente, lo que obliga a definir normas de convivencia de lo antiguo con lo nuevo, las cuales tienen mayores exigencias de procesamiento, no resultando particularmente simple.

Por lo anteriormente expresado y teniendo en cuenta el interés de nuestro proyecto, el informe estará dividido en tres partes. La primera intenta brindar un enfoque general, dando una visión de un servicio de mail, sus aspectos conceptuales y el entorno en el cual trabaja. También daremos una introducción a los conceptos que sobresalen en este proyecto, como los protocolos de mail (POP, SMTP, ESMTP y NNTP) y la interfaz para el transporte de red como son los Winsock. Este último punto

considera la adaptación para Windows de los sockets introducidos en la versión 4.1 del BSD Unix (Winsocks). La segunda parte trata sobre la implementación de un conjunto de primitivas que abarca los protocolos anteriormente citados. Por último se darán las conclusiones de lo realizado en la segunda parte y se analizarán nuevas extensiones para futuras implementaciones.

Objetivo

Todo sistema que se precie de brindar servicio de mail está compuesto básicamente por dos subsistemas. Uno que abstrae al usuario de todos los detalles de como se transportan los mensajes, como se encuentra el destino, y las transformaciones que pueden ser necesarias para alcanzar dicho destino, además también este subsistema debe ser capaz de crear un entorno de interacción para poder visualizar viejos mensajes, reenviarlos, manejar listas y alias de usuarios, entre otras cosas. Todo esto forma parte de lo que se conoce como User Agents (U.A.). El otro subsistema tiene por objetivo rutear y administrar los mensajes, realizando las transformaciones adecuadas para llegar a la dirección correcta, se lo conoce como Message Transfer Agents (MTA)

Estos subsistemas se relacionan mediante distintos protocolos que son básicamente el objetivo de nuestro proyecto, que de alguna manera son la última parte que queda por estudiar dentro del laboratorio al cual pertenecemos, ya que tanto se realizó un trabajo de grado evaluando distintos MTA's y en otro en el cual se realizó un U.A. usando UUCP como medio de transporte.

Si bien el proyecto define funcionalmente cada una de las partes que componen un sistema de mail, pone especial énfasis en los protocolos de entrega y recepción de mensajes ya que llegaremos a la implementación de algunos de ellos, como lo son: POP, y SMTP y a la definición formal para NNTP y ESMTP (MIME y DSN).

En el proyecto se fijaron dos objetivos primordiales, que son fundamentales para futuros desarrollos, esto es: la capacidad de extensión a otros medios de transporte, ganar independencia del medio de comunicación (por ej. tipo de línea, conexión y protocolo) y la posibilidad de incorporar nuevos servicios que le permitan al U.A. ampliar su funcionalidad. Para lograr estos objetivos, por un lado nos basamos en el poder de abstracción que nos dan los sockets como medio de comunicación entre procesos y por otro lado la gran ortogonalidad que presentan los diferentes protocolos de entrega y recepción de mensajes, sobre todo si vemos como se realizan las distintas extensiones para ir incorporando nuevas funcionalidades a los sistemas de mensajería.

El proyecto requirió el estudio y análisis de las definiciones de los diferentes protocolos, como también desarrollar una implementación de cada uno usando la funcionalidad elemental provista por los Winsocks.

Alcance del trabajo propuesto

Las técnicas de los sockets de Unix [COMER 93] son poderosas y requieren su comprensión para hacer un mejor manejo de las posibilidades de comunicación para aplicaciones Cliente-Servidor.

La implementación de esta técnica sobre Windows [WINSOCK1] posibilita el acceso estándar a estos servicios. La dificultad aquí, radica en el uso de esta capa por parte de las distintas aplicaciones. A fin de ganar compatibilidad se adoptó un estándar de Winsock con la intención de poder interactuar con diversos productos de base (esto fue también objetivo de testeo).

Por último, los protocolos de mensajería son variados y complejos. A fin de desacoplar las características de la interfaz de usuario del Agente de Usuario, se analizaron los estándares :

- SMTP (68 pág)
- ESMTP (50 pág)
- POP (15 pág.)
- NNTP (27 pág.)

Además, se determinó un conjunto de primitivas base para que el diseñador de la interfaz del User Agent pueda abstraerse del protocolo de mensajería en sus detalles de implementación.

PARTE I

CONCEPTOS

Aspectos Conceptuales

El correo electrónico (e-mail)

El correo electrónico provee un medio de comunicación entre usuarios que tienen un objetivo en común y es además, un servicio de entrega automática de objetos de información.

Aunque el correo electrónico puede verse como una forma particular de transferencia de archivos, tiene características peculiares. Los extremos emisores y receptores son personas y no máquinas, esto dio como resultado que los sistemas de correo electrónicos se dividan en dos partes, una relacionada con la interfaz humana y otra con el transporte de los mensajes. Otra diferencia es que los mensajes de mail son documentos estructurados. Parte de esa estructura está compuesta por campos con información de control que permiten el ruteo de los mensajes y la posterior interpretación.

Un sistema que ofrezca un servicio de mail debe exhibir cualidades que se expresan en el siguiente modelo de referencia.

Modelo para el manejo de mensajes

Plantaremos ahora un modelo conceptual genérico para correo electrónico [ROSE 93] (e-mail) :

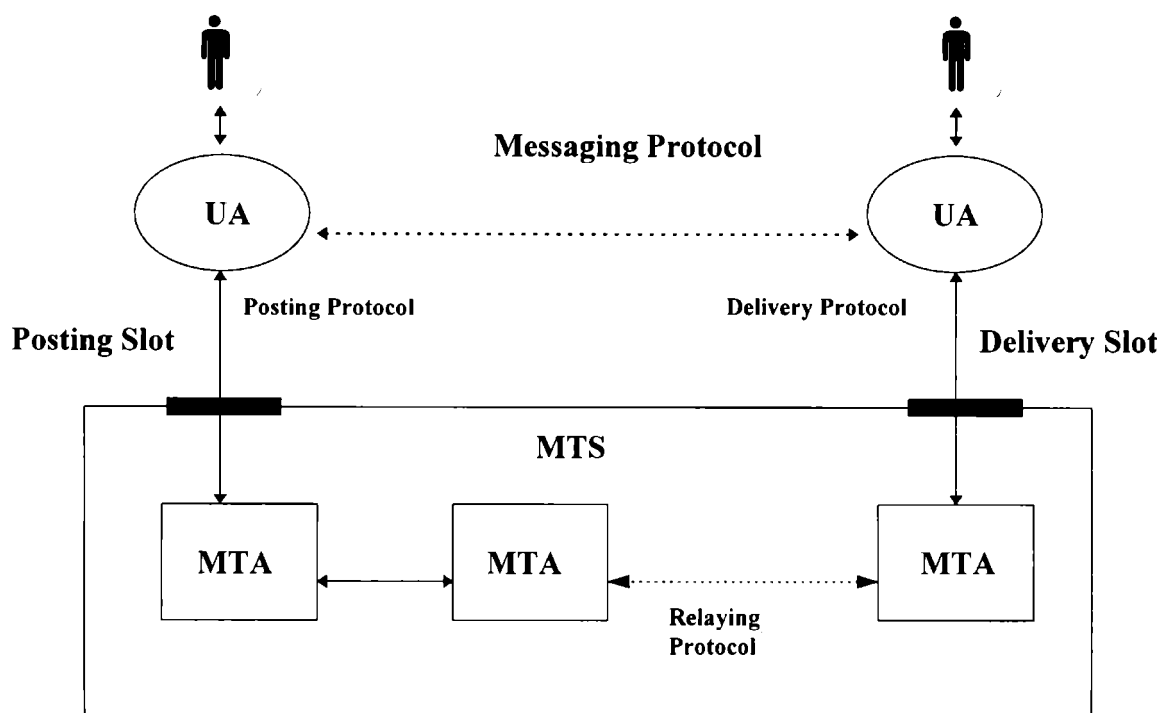


Figura 1

Antes de comenzar la descripción del modelo conceptual que muestra la figura 1 se definirán brevemente conceptos involucrados en el sistema de mensajería.

Alias

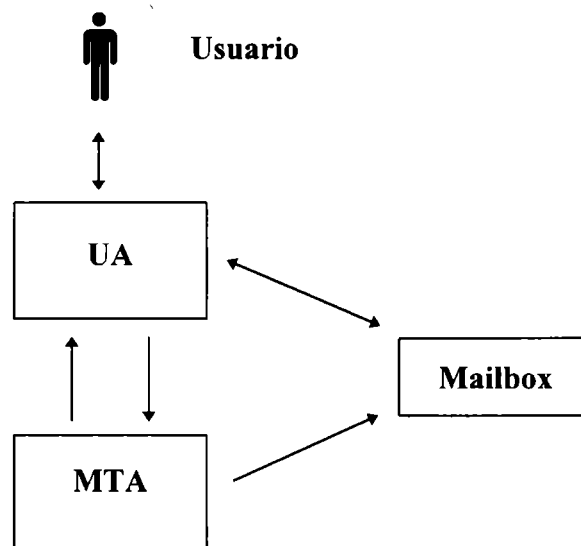
Es la forma de asignar nombres alternativos a usuarios individuales (nicknames), manejar listas de mail y retransmitir mensajes a otros hosts.

Los alias que definen nicknames pueden ser usados para manejar nombres usados frecuentemente o difíciles de escribir o recordar. Un alias puede tener un conjunto de direcciones de mail, formando una lista de usuarios que tienen intereses en común.

Mailbox

Es el equivalente a una casilla de correo en el sistema postal y permite recibir un mensaje aún cuando el usuario del servicio no este conectado.

A partir de la dirección destino de mail el sistema determinará el nombre del mailbox donde deberá depositar los mensajes.



Descripción del modelo

El modelo de la figura 1 se compone de una serie de *Message Transfer Agent* (MTA), los cuales conforman el *Message Transfer System* (MTS). En los extremos del sistema un *User Agent* (U.A.) actúa como una interfaz entre el usuario y el MTA local.

El MTA será encargado y responsable de que un mensaje introducido al sistema, llegue a destino. Para ello, deben elegir mecanismos adecuados, controlar el funcionamiento de los transportes que se han de usar para comunicarse y administrar los mensajes, ubicándolos en los mailbox correspondientes o bien direccionándolos a otros MTA's. El direccionamiento, en general, es la búsqueda de una serie de MTA's por medio de los cuales es posible llegar al mailbox del usuario destino. Para cualquier MTA el direccionamiento consiste en encontrar otro MTA al cual reenviar el mensaje, salvo que el destinatario sea local. [RUSSO 94]

Cuando un mensaje de correo electrónico (de ahora en más "mensaje") es enviado de un usuario a otro; el usuario emisor indica al U.A. la dirección del receptor y este coloca las direcciones destino y origen en el "*sobre*". Luego deposita el mensaje por el slot de ingreso (*posting slot*) al MTA (equivalente a poner una carta en un buzón en la oficina postal) utilizando un protocolo ad-hoc (*posting protocol*) que se encargará de validar las direcciones y el contenido del mensaje. Una vez realizada dicha validación, el MTA acepta la responsabilidad de entregar el mensaje, como también si este falla, informar al emisor con un reporte de error. Si el MTA detecta que puede entregar el mensaje en forma directa, lo hace a través del slot de entrega (*delivery slot*) al recipiente del U.A., usando para ello un protocolo de entrega (*delivery protocol*). Sino, se contacta con el MTA adyacente más cercano al receptor y negocia la transferencia del mensaje utilizando un protocolo de retransmisión (*relaying protocol*). Este proceso se repite hasta que el MTA sea capaz de entregar el mensaje o bien determine que no lo puede hacer.

También se describe en el gráfico un protocolo general entre los U.A.; el *messaging protocol*, el cual es una abstracción de los demás protocolos referenciados en el modelo.

La transferencia de mensajes es principalmente, *store-and-forward* (almacena y reenvía) y además no necesita que los UA's origen y destino estén simultáneamente on-line. Se debe destacar que el estándar SMTP es directo, excepto que use mail exchange, cuando para alcanzar el destino debe utilizar otros nodos intermedios.

Formato de los mensajes

Ahora consideraremos el *messaging protocol* que une dos U.A.

La unidad mínima de información que maneja el e-mail es el mensaje, el cual contiene un cuerpo de estructura arbitraria que transporta "algo" con significado entre el emisor y el receptor. Formalmente, los mensajes de mail tienen dos tipos de información: Los encabezamientos (*headers*), que son usados para transportar información de control y el cuerpo (*body*) que es utilizado para transportar datos. La sintaxis básica de estos mensajes está definida en la **RFC-822** (Para el Internet Suite) y en **X400 84/88** (Para el OSI Suite). Esto se verá en más detalles en secciones posteriores.

El U.A. (Agente Usuario)

Pondremos mayor énfasis en esta parte del modelo dado que la finalidad del proyecto es proveer un servicio de entrega y recuperación de mensajes hacia y desde el MTS. El U.A. provee básicamente una interfaz con el usuario para el ingreso y extracción de mensajes del mailbox y además se comunica con el MTS para la entrega y aceptación de los mismos.

La interfaz de usuario provee una presentación que permite ingresar comandos relacionados con el envío, la composición y recepción de mensajes. En algunos, dicha interfaz es gráfica, iconizada y con menús estilizados.

Funcionalidad del U.A.

La funcionalidad que debe proporcionar un U.A. está claramente dividida en 3 categorías :

Generación : el mensaje es creado y entregado al MTA local

Hay cuatro actividades de generación básicas:

1. **Composición** : en la cual un nuevo mensaje de mail es compuesto y enviado.
2. **Redirección** : en la cual un mensaje recibido es enviado palabra por palabra a recipientes adicionales.
3. **Transmisión** (Forwarding) : En la cual uno o más mensajes recibidos son combinados en uno nuevo y enviado a nuevos recipientes, pudiendo agregar preguntas o comentarios..
4. **Contestación** (Replying) : Se genera una respuesta destinada al emisor del mensaje y eventualmente a otros receptores, conteniendo total o parcialmente el mensaje original indentado, posibilitando distinguir las anotaciones que se agregan.

Examinación : Permite examinar el mensaje por parte del usuario después que ha sido entregado al U.A.

Existen dos facilidades de examinación :

1. **List**: Provee un resumen del contenido del header del mensaje al usuario.
2. **Display** : Donde un mensaje completo es mostrado al usuario.

Procesamiento: El mensaje recibido es manipulado.

Esta funcionalidad provee dos facilidades, las cuales se basan en las características del encabezamiento como ser fecha de emisión, recepción, origen, tema o prioridad :

1. **Búsqueda**: Permite buscar mensajes según algún criterio de búsqueda.

2. **Ordenamiento:** Permite ordenar los mensajes según algún criterio.

También deben considerarse funcionalidades adicionales presentes en todo U.A. tal como :

- **Impresión**
- **Guardar los mensajes en carpetas históricas (Folders).**
- **Manejo de nicknames:** definición de apodos, alias y listas de distribución personal.

Medios de Transporte

Introducción

Los MTA's una vez que aceptan la responsabilidad de un mensaje, deben transportarlo en la red hasta llegar a su destinatario o bien hasta que puedan detectar que no lo pueden entregar. Los MTA's establecen la ruta usando otros MTA's intermedios, para ello utilizan distintos protocolos de comunicación de red. Dichos protocolos han evolucionado debido a los avances tecnológicos en las comunicaciones y redes de computadoras. Los protocolos de red más comunes para el transporte de mail son : **TCP/IP** y **UUCP** los cuales serán tratados en secciones posteriores.

Para comenzar haremos una introducción a algunos conceptos básicos de redes.

Arquitectura de redes

Una arquitectura de red es un conjunto de reglas y especificaciones para el diseño de las comunicaciones, y la distribución y performance de funciones dentro de la red. Una arquitectura de red incluye hardware, software, topología y protocolos.

Las funciones de red pueden ser diseñadas en módulos o capas. El número, nombre y función de cada capa varía de red en red y cada una se construye sobre su predecesora.

La capa n de una máquina dialoga con la capa n de la otra máquina (*peer-to-peer*). Las reglas y convenciones utilizadas en este dialogo se denomina protocolo de la capa n . En realidad no existe una transferencia directa de datos entre las dos capas mencionadas; sino que cada capa pasa datos e información de control a la capa inferior hasta llegar a la capa más baja de la estructura. Debajo de la capa 1 está el medio físico, a través del cual se realiza la comunicación real. Este modelo es mostrado por la Figura 2.

Entre cada par de capas adyacentes existe una interfaz, la cual define los servicios y operaciones que la capa inferior ofrece a la superior.

Cuando un mensaje es enviado de una máquina a otra, sufre transformaciones al pasar a través de cada interfaz. Estas transformaciones se producen por la adición de información de control (headers) propia del protocolo de cada capa (ver figura 3). Se debe destacar que una capa no define un único protocolo, sino una función de comunicación de datos, que quizás es realizada por varios protocolos.

A continuación veremos dos modelos de los denominados abiertos : el Open Systems Interconnection (OSI) suite como modelo de referencia y el Internet suite (comúnmente conocido como TCP/IP) como estándar de facto.

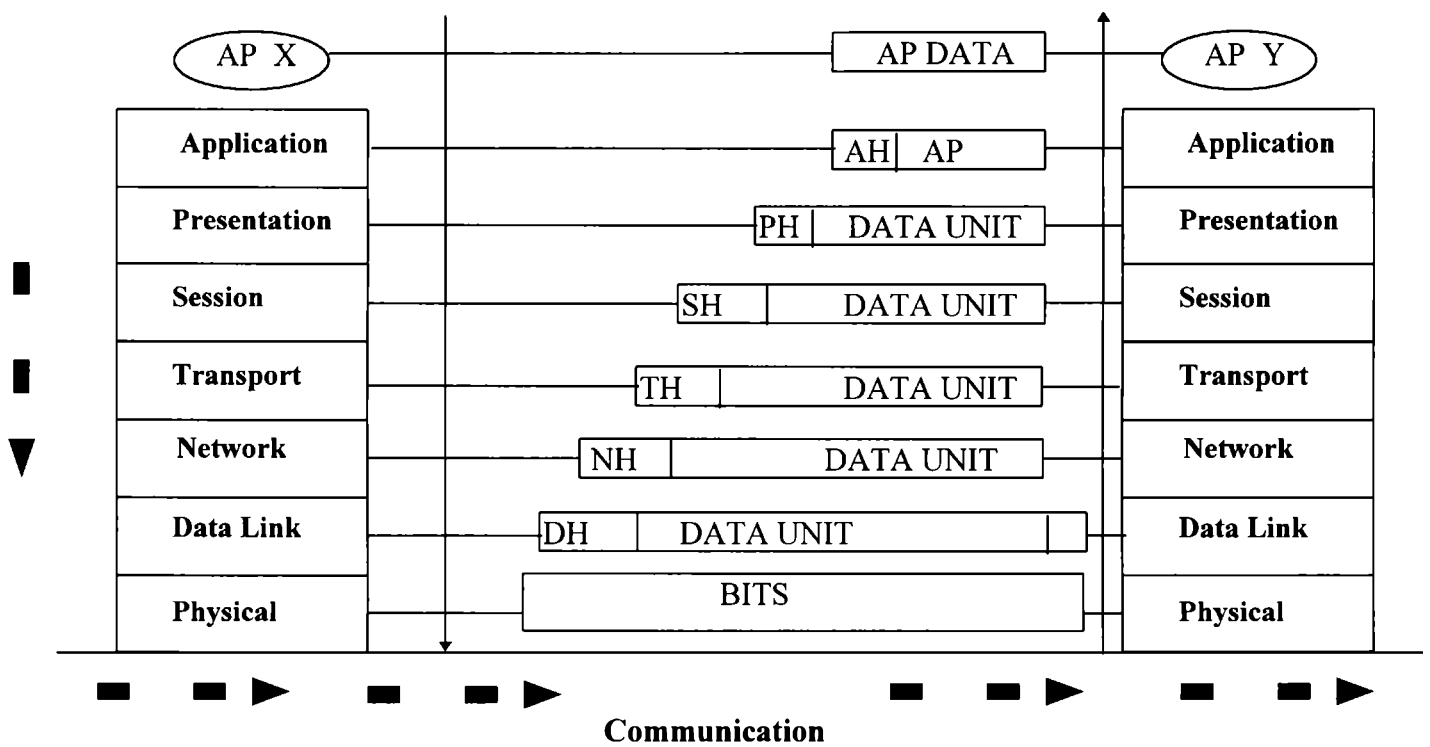
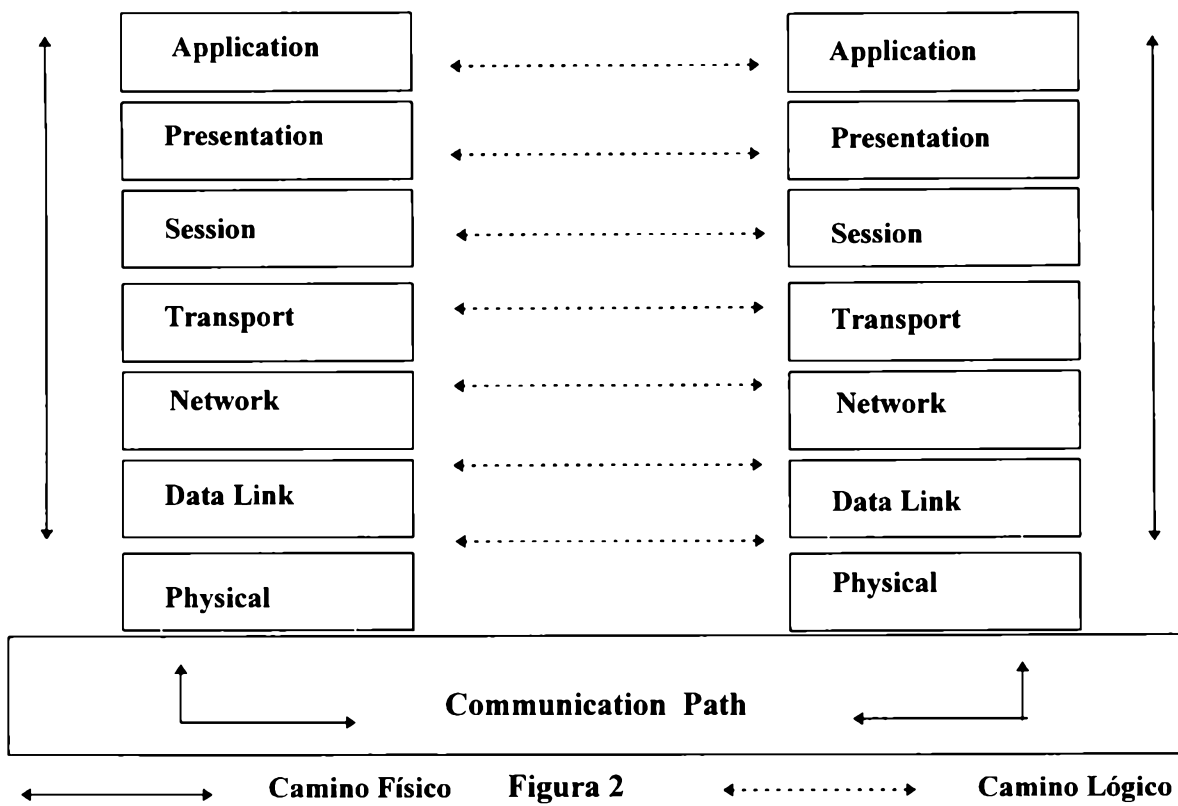


Figura 3

Modelo de Referencia OSI

Para proveer una manera uniforme de describir que función realiza cada capa, la **ISO** (*International Standardization Organization*) propuso el modelo de capas denominado **OSI** (*Open System Interconection*). Este modelo, es generalmente utilizado como un framework sobre el cual se describen las características y funcionalidad de los protocolos [TANEN 91]. La siguiente figura muestra las capas del modelo :

• 7	Application Layer
• 6	Presentation Layer
• 5	Session Layer
• 4	Transporte Layer
• 3	Network Layer
• 2	Data Link Layer
• 1	Physical Layer

Figura 4

Physical Layer: Provee la forma de comunicación sobre el medio físico y determina la característica de la transmisión . (Cosas tales como nivel de voltaje o número y ubicación de los pines de la interfaz) Ejemplo de esta capa son las interfaces X.21, RS-232.

Data Link Layer: La tarea de esta capa es asegurar la confiabilidad de la transmisión de unidades de información (packets o blocks) y direccionar a las estaciones conectadas al medio de transmisión. Debe encargarse de “fragmentar” los datos de entrada en unidades de información y reconstruirlos en el destino, por lo cual debe reconocer los límites de dichas unidades y además debe realizar los tests de integridad. Ejemplos de estos protocolos son : HDLC y CSMA/CD

Network Layer: Se encarga de establecer caminos virtuales entre las estaciones de la red para el intercambio de mensajes. También es función de esta capa la segmentación y bloqueo de mensajes y tareas, tales como, el control de congestión de la red. Ejemplos de esta capa son : X.25.

Transporte Layer: La mayor función de esta capa es asegurar la calidad y la entrega de los mensajes. Esto incluye secuenciación de mensajes, control de flujo, establecimiento y cerrado de conexión. Ejemplo de esta capa es el IPX .

Session Layer: El propósito de esta capa es organizar y sincronizar el diálogo y el flujo de datos entre entidades de presentación en el envío y recepción de datos. Las sesiones permiten que el tráfico vaya en una o ambas direcciones en un momento dado.

Presentation Layer : Provee servicios de codificación y traslación del conjunto de caracteres y formateo de datos. El servicio de formateo de datos es utilizado por la

capa de aplicación y por varios dispositivos de hardware asociados con los programas de aplicación. La compresión de datos puede realizarse a este nivel.

Application Layer : Esta capa provee la interfaz que permite a los programas de usuarios acceder a la red. Normalmente los proveedores de sistemas incluyen aplicaciones que realizan servicios estándar tales como: servicio de terminal virtual, file transfer y e-mail.

El Internet Suite

El Internet Suite, conocido popularmente como TCP/IP, difiere en su arquitectura del modelo de referencia OSI. La cantidad de capas y la funcionalidad de cada una de las mismas son distintas. La estructura del Internet Suite está definida en cuatro capas (layer) como muestra la siguiente figura :

• 4	Application Layer
• 3	Transporte Layer
• 2	Internet Layer
• 1	Subnetwork Layer

Figura 5

Subnetwork layer : describe la capa física y la tecnología de unión de datos para realizar la transmisión sobre el hardware. Esta capa a menudo es ignorada por los usuarios, dado que el diseño del TCP/IP oculta los detalles de las capas inferiores.

Las funciones realizadas a este nivel incluyen la encapsulación de datagramas IP en frames a transmitirse por la red y el mapeo de direcciones IP en direcciones físicas usadas por la misma. Por ejemplo, el ARP (Address Resolution Protocol) definido por la RFC 826, especifica como los datagramas IP son encapsulados para transmisión sobre redes Ethernet.

Internet layer: Describe la tecnología de interconexión de redes . El Internet Protocol (RFC 791) es el corazón del TCP/IP y demás protocolos importantes. Todos los protocolos de las capas superiores e inferiores utilizan IP para la transmisión de datos . Entre las funcionalidades del IP se encuentran

- Definición de datagramas, los cuales son la unidad básica de transmisión en la Internet.
- Definición del esquema de direccionamiento
- Ruteo de datagramas a host remotos
- Fragmentación y reensamblado de datos.

Como parte integral del IP , ICMP (Internet Control Message Protocol) definido en la RFC 792 ,es quien envía los mensajes de control y reportes de error ocurridos en toda la red y sus nodos al TCP/IP

Transport layer: Describe la tecnología usada para realizar una comunicación confiable con detección y corrección de errores entre hosts. Los protocolos más importantes son el TCP (RFC 793) y UDP (RFC 768) y se encargan de entregar los datos entre la capa de aplicación y la capa internet. El primero ofrece un servicio orientado a conexión, con detección y corrección de errores de los datos entregados (Testea que los datos lleguen íntegros y en orden , por lo cual se dice que es “confiable”). El UDP es un protocolo de transporte sin conexión (connectionless) y no provee técnicas de verificación de datos en la entrega (Se dice que “no es confiable”).

Application layer: describe la tecnología usada para proveer diferentes servicios a los usuarios. Esta capa concentrará la mayor atención, dado que sobre ella se ubican los protocolos de aplicación involucrados en el manejo de mensajes. Hay varios protocolos de aplicación disponibles; los específicos de mail son :

- **Simple Mail Transfer Protocol (SMTP)** el cual provee servicio store-and-forward para mensajes de textos y la RFC-822 define los formatos de estos mensajes.
- **Extended Simple Mail Transfer Protocol (ESMTP)** como extensión de los servicios de SMTP, que permite eliminar algunas de las restricciones por él impuestas al transporte de mail (por ej. envío de objetos binarios sin codificación previa).
- **Post Office Protocol (POP)** que provee un servicio de recuperación de mensajes del mailbox bajo demanda.
- **Interactive Mail Access Protocol (IMAP)** que permite una comunicación asincrónica entre cliente y servidor para intercambiar mensajes sin solicitud (en forma interactiva).
- **Network News Transfer Protocol (NNTP)** provee servicio store-and-forward para la entrega de artículos a grupos de usuarios.
- **Domain Name System (DNS)** provee un servicio de transformación de nombre de host a direcciones de red.

La siguiente figura muestra los protocolos en las capas de Internet Suite y muestra su equivalencia con las capas del modelo de referencia OSI

Protocolos en el TCP/IP	Modelo OSI
TELNET	Application
FTP	
TFTP	
SMTP	
DNS	
NNTP	
POP	
TCP -UDP	Transport
IP	Network
SUBNETWORK	Data Link
	Physical

Figura 6

En términos de los protocolos genéricos mencionados anteriormente en el modelo (Fig. 1), la RFC-822 se corresponde con el messaging protocol y el SMTP con el relaying protocol. En el Internet Suite los submission y delivery protocols son cuestiones locales, aunque SMTP y POP pueden proveer esa funcionalidad.

UUCP (Unix to Unix copy)

Es un conjunto de programas originalmente diseñado para interconexión de sistemas Unix. Una implementación mínima debe permitir: transferencia de archivos entre sistemas (uucp), ejecución de comandos remotos (uux), envío de mail a usuarios en sistemas remotos (rmail).

UUCP no requiere hardware y software especializado, ya que está diseñado para funcionar con cables seriales, modems y líneas telefónicas. La comunicación es establecida entre sistemas solamente cuando los usuarios la requieren. Cada sistema en una red UUCP tiene archivos de configuración que describen a los sistemas directamente conectados a él y el tipo de enlace que está disponible.

Los comandos de ejecución remota son limitados por razones de seguridad.

UUCP es una red store-and-forward, o sea, los requerimientos de ejecución remota de comandos o transferencia de archivos no son ejecutados inmediatamente, pero si son almacenados para cuando la comunicación se establezca entre dos sistemas (inmediata o no).



Cuando un usuario invoca los comandos uucp, uux o envía un mail; dos eventos son necesarios:

1. Crear un archivo de control conteniendo información tal como: los nombres de los archivos fuentes y destinos, las opciones de uucp y uux y el tipo de requerimiento.
2. Invocar al programa uucico para realizar la transferencia.

El archivo de trabajo no contiene información de cuando y como hacer la transferencia, esta está contenida en otros archivos de configuración.

En UUCP podemos distinguir dos capas :

La capa de Aplicación , que esta compuesta por todos los programas que constituyen las funcionalidades antes mencionadas (uucp, uux, uucico, uuto).

La capa de transporte , la cual es utilizada para realizar las transferencias. En sus inicios el único medio de transmisión fue con modems utilizando líneas telefónicas. En la actualidad , se puede usar como medio de transporte TCP/IP y X.25.

Direcciones

Como mencionamos anteriormente un mensaje de mail tiene un emisor y uno o varios receptores. Cada uno de ellos (emisor y receptor/es) tiene asociada un dirección que lo identifica en la red. Debe notarse la diferencia entre la dirección del host que actúa como servidor de mail (Equipo donde recibe los mensajes) y la dirección de mail (Dirección que lo identifica en el e-mail). La primera identifica al host unívocamente en la red y la segunda contiene, además de la identificación de host, el nombre que identifica al usuario en dicho host . Las direcciones tienen una estructura que depende del transporte. En los siguientes párrafos describiremos los formatos de las direcciones TCP/IP y UUCP

Direcciones TCP/IP

Para transmitir datos entre dos host en una red TCP/IP es necesario conocer la dirección correcta. El direccionamiento de una comunicación entre pares, como lo son programas de aplicación, requiere cuatro direcciones diferentes cuando se atraviesan las cuatro capas del modelo :

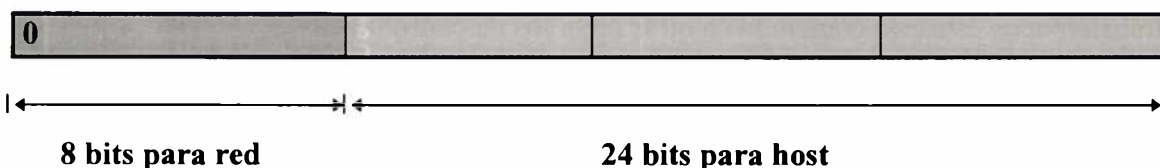
1. Una dirección a nivel subred (Por ejemplo una dirección Ethernet)
2. Una dirección Internet
3. Una dirección del protocolo de transporte
4. Un número de port

La más importantes para nuestro proyecto, es la dirección Internet (IP) , la cual tiene 32 bits y es representada como cuatro números separados por puntos (por ejemplo 163.10.0.105) . Es la más importante porque cada nodo de la Internet tiene una

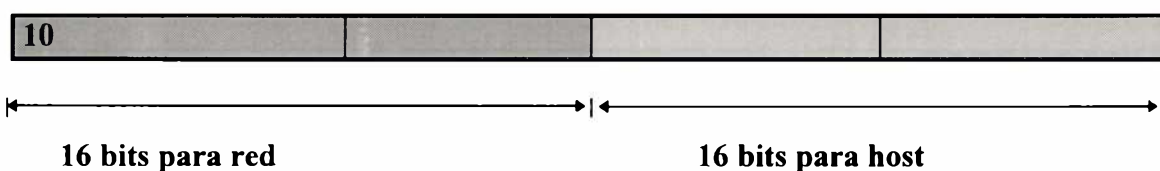
única dirección IP y forma parte del header de los datagramas cuando son transmitidos a través de la red.

La dirección IP contiene una parte de identificación de *red* y una parte identificación de *host*. Dependiendo de estas, las direcciones se clasifican como : Clase A, Clase B y Clase C

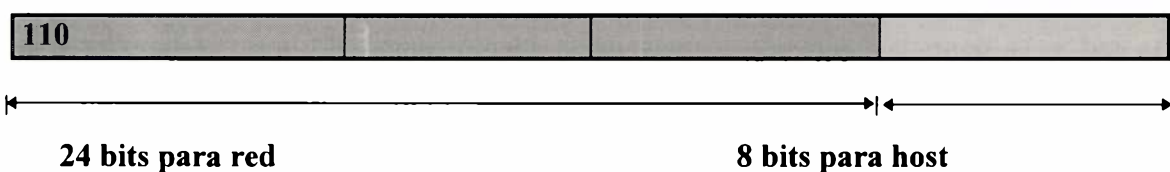
Clase A



Clase B



Clase C



Podemos notar que la dirección se forma por cuatro números decimales , cada uno en el rango de 0-255 separados por puntos. Con el primer valor podemos determinar la clase :

- menor que 128 indica clase A
- entre 128 y 191 indica clase B
- entre 192 y 223 indica clase C
- mayor de 223 indica direcciones reservadas

El porqué de las clases de direcciones es simple : cuando ARPANET comenzó, había solo redes de clase A, asumiendo que solo habría una pocas redes grandes. Con la introducción de LAN'S en muchas organizaciones se debió reorganizar y así se crearon las clases B y C. Actualmente, se sobrelleva una falta de direcciones, lo que desencadenó un proceso de definición de nuevas direcciones denominadas **IP ng**, las cuales tendrán 128 bit de longitud.

Las direcciones oficiales IP deben obtenerse del NIC (Network Information Center) por la necesidad de unicidad a nivel mundial.

Subredes (RFC-877)

De lo anteriormente expresado se deduce que el identificador de red se corresponde con una red física, o sea, que si tuviéramos varias redes físicas en un determinado lugar, necesitaríamos un identificador de red IP por cada una de ellas.

Adoptar esta solución en cada uno de los nodos de la Internet nos llevaría a un agotamiento de las direcciones, y además se tendría información semánticamente redundante en las tablas de ruteo.

La solución es introducir un tercer nivel en la jerarquía de direccionamiento (teníamos identificador de red e identificador de hosts), subdividiendo el identificador de host.

La idea general es definir una segmentación de la red en subredes. Esto permite, además, una administración descentralizada .

Identificador de Red	Identificador de Host	
⋮	⋮	⋮
Identificador de Red	Número de subred	Número de host

La idea de subred es que hacia afuera del nodo se usa la dirección IP con identificador de red e identificador de host y hacia adentro se utiliza un número de subred y un número de host, donde el número de subred se refiere a una red física particular dentro del nodo [MILLER 90].

Una subred se define aplicando una “máscara de red “ a la dirección IP. La máscara es interpretada bit a bit de la siguiente forma: si es 0 se toma como bit de red y si es 1 como bit de host. Luego se realiza un AND lógico entre la dirección y la máscara. De esta manera se debe agregar un nuevo valor a la tabla de ruteo: la máscara de red.

Como ejemplo, la UNLP tiene una red clase B, cuya dirección IP es 163.10.0.0. Como mencionamos anteriormente, de esa dirección los dos primeros bytes se corresponden con la identificación de red y los dos últimos con identificación de host. Al introducir subredes, se tomaron de los dos últimos bytes, 10 bits para subredes y 6 bits para hosts, con lo cual se obtiene 1024 subredes de 64 host cada una. La máscara utilizada es 255.255.255.192, pues tendrá en uno todos los bits, excepto los seis últimos (con valor 0).

Nombres y direcciones TCP/IP

Como mencionamos anteriormente todo host conectado a una red TCP/IP es identificado por un único número de 32 bits. Un nombre (nombre de host) puede ser asignado a cualquier host que tenga una dirección IP. La ventaja de asignar un nombre es que es más fácil de recordar que un número. El software no requiere de esos nombres, pero ellos ayudan a los usuarios en el uso de la red.

En muchos casos, el número y el nombre pueden usarse indistintamente, por ejemplo, usando telnet para servicio de terminal virtual podemos poner :

telnet 163.10.0.66 ó
telnet isis.unlp.edu.ar

Sin embargo, la conexión de red siempre trabajará con el número de la dirección IP. El sistema transforma el nombre en dirección antes de que la conexión se realice. El administrador de sistema debe ser el responsable de asignar los nombres a direcciones y mantener la base de datos utilizada para la conversión. La *traducción* de nombre a direcciones es un problema local y existen dos maneras de hacerlo: el viejo método que simplemente busca en una tabla (*host table*) la equivalencia dirección-nombre y la otra técnica, que usa una base de datos distribuida, llamada **DNS** (Domain Name Service). Se debe destacar que DNS tiene además, la equivalencia nombre-dirección (reverse-address).

Asignación de nombres

Un aspecto importante a tener en cuenta es la forma de identificación de entidades dentro de una red. En el caso particular de la comunidad Internet, una dirección de mail es utilizada para identificar a un usuario. El formato de tal dirección es:

local@domain donde **domain** es una autoridad administrativa responsable de dar nombres a las entidades y **local** es un string con significado dentro de un dominio.

Una buena asignación de nombres debe proveer un excelente soporte para la asignación distribuida, ejecución y mantenimiento de los mismos. Estos objetivos se logran utilizando una estructura jerárquica.

Conceptualmente, el espacio de representación de un nombre de dominio es un árbol; esto significa que todos los nombres inmediatamente subordinados a un nodo deben ser únicos. Lo anterior indica que un label para un nodo particular no podrá ser igual para otro del mismo nivel.

Un nombre de dominio se escribe como uno o más labels separados por punto (“.”). Por convención, la raíz se representa con el label vacío.

Bajo el root domain están los dominios de alto nivel (*top level domain*), de los cuales hay dos tipos básicos : geográficos y organizacionales.

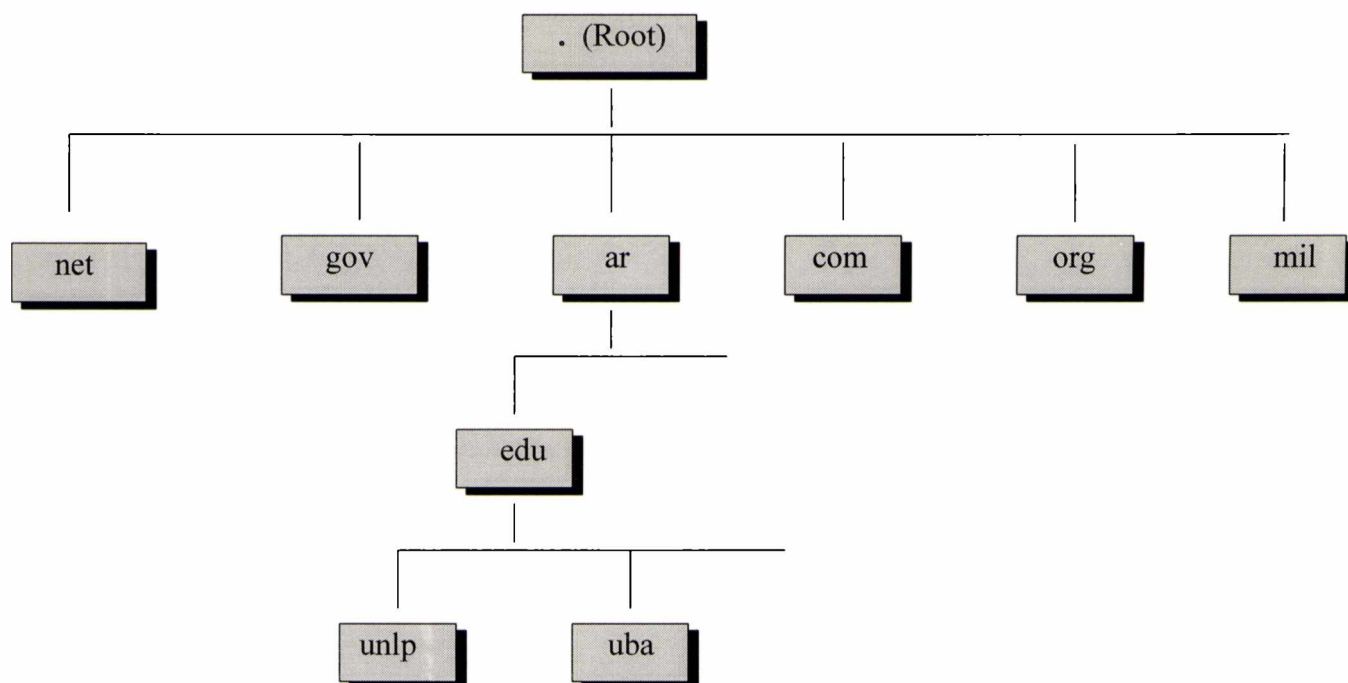


Fig. 7

La figura anterior muestra un ejemplo de una jerarquía de dominios. Los Dominios geográficos están organizados por país, cuyos nombre están codificados con dos letras. Por ejemplo: AR (Argentina), US (EE.UU.), JP (Japón) . La creación de los dominios y subdominios de primer nivel está a cargo del NIC (Network Information Center).

DNS (Domain Name Service)

DNS tiene dos grandes ventajas sobre la host table

- DNS es escalable . No es una gran tabla, sino un único sistema de bases de datos distribuida .
- DNS garantiza que la nueva información será diseminada al resto de la red cuando sea necesario (Bajo requerimiento).

La información no solo es automáticamente diseminada, sino que también es distribuida a quienes están interesados. Si DNS recibe un requerimiento de un host del cual no tiene información, se lo pasa a un servidor responsable de más alto nivel quién delegará explícitamente en un host más específico. Este servidor, es cualquier servidor autorizado de mantener información segura acerca del dominio que está siendo consultado. Cuando este servidor responde, el host local salva (usando memoria cache) la respuesta para futuros usos.

Bajo DNS, no hay una base de datos central con información de todos los host de la Internet. La información está distribuida entre cientos de servidores de nombres, organizados de forma jerárquica. Existe un dominio raíz (*root domain*) en la parte más alta de la jerarquía, que es atendido por un grupo de servidores denominados *root servers*.

Direcciones UUCP.

Hemos mencionado previamente la interconexión de redes con UUCP y su gran comunidad. Ahora veremos el formato de las direcciones de mail.

Hay dos clases importantes de direcciones de mail utilizadas en el mundo UUCP:

1. El formato `a!b!c!user` usada por las viejas versiones para rutear explícitamente el mensaje a su destino. Si un usuario U1 del host H1 debía entregar un mensaje al usuario U2 del host H2 pasando por el host H3, la dirección era `H3!H2!U2`. El `!` (bang) es el separador utilizado por UUCP.
2. La sintaxis `user@domain` usada conforme a la RFC 822 y utiliza tablas de ruteo.

Una diferencia importante entre los dos formatos es que la primera forma no permite que el sistema de transporte elija la ruta según condiciones dinámicas.

Sin embargo existe la posibilidad de expresar direcciones en una sintaxis híbrida, tal como:

`a!b@c.d`

En la red UUCP, cada nodo conoce sus nodos adyacentes y para transmitir un mensaje de mail ejecuta el comando `rmail` en el nodo remoto y sus argumentos son la dirección y el contenido del mensaje.

Estructura de un mensaje

RFC 822.

En general un mensaje es un objeto de información, esto es, contiene un cuerpo arbitrariamente estructurado que transporta información con significado entre el emisor y el receptor.

En la comunidad Internet un mensaje de e-mail contiene información de dos tipos:

- *headers*: usados para transportar información de control y usualmente es descripto como el *sobre*..
- *body*: utilizado para transportar los datos del mensaje (contenido de la carta).

Para el alcance de este proyecto la sintaxis de los mensajes de correo electrónico está definida en la RFC-822.

Similarmente a todos los protocolos Internet, un mensaje contiene caracteres del repertorio NVT ASCII, el cual es utilizado para el intercambio de información. El formato de los mensaje de mail establece pautas para dicho intercambio y no como la información es representada o almacenada localmente. Todo esto puede verse muy limitado dada la variedad de las representaciones de la información. Sin embargo posteriormente veremos como podemos extender las definiciones para transportar objetos binarios arbitrariamente estructurados (MIME).

Los headers son identificados por una palabra clave y un valor estructurado. En contraste, el cuerpo (*body*) es visto como texto sin estructura.

Definición Formal.

Una especificación formal de un mensaje de mail en alto nivel utilizando **BNF** sería:

```

message      ::= 1*field * (CRLF *text)
field        ::= name ":" [value] CRLF
name         ::= 1 * <cualquier caracter, excepto SP o ": " >
value        ::= text * (LWSP text)
text         ::= <cualquier caracter, incluido CR y LF , pero no incluyendo CRLF>
LWSP         ::= CRLF 1 * SP
CRLF         ::= < carriage-return seguido por line feed>
SP           ::= < SPACE o TAB>

```

- La sintaxis para la regla **value** depende de la palabra clave utilizada para identificar al header.

De la especificación anterior se puede deducir lo siguiente :

- Un mensaje de mail comienza con uno o más headers.
- Cada header es identificado por una palabra clave seguida de un valor.
- El valor puede tener múltiples líneas. La primera siempre debe estar precedida por la palabra clave.
- El cuerpo comienza después de la primer línea en blanco.

Estructura de los valores de los headers.

Con la idea de permitir un procesamiento automático por parte del U.A., es necesario definir los valores de los headers en forma estructurada. Para construir dichas estructuras se definen los siguientes bloques :

comments: Información textual con significado para humanos, delimitada por paréntesis, puede ser multilínea y pueden estar en cualquier lugar del mensaje.

domain-literal: Información de direccionamiento en forma nativa, usada cuando se produce un error en la translación de dirección a nombre. Se delimita con corchete (“[”,”]”). Pueden ser multilínea.

quoted-string: Conjunto de caracteres tratados como una unidad. Utilizado cuando el valor tiene caracteres en blanco o separadores. Puede ser multilínea.

atom: Es usado cuando las únicas componentes del valor son caracteres alfanuméricos sin separadores.

word: Está compuesta por un **atom** o un **quoted-string**.

special: Incluyen caracteres separadores.

La representación en BNF (manteniendo la sintaxis original) de lo anteriormente expresado está descrita en el **Anexo A**.

A continuación describiremos a modo de ejemplo algunos headers:

- **From:** Identifica a quien ha iniciado el mensaje.
- **Received:** Una copia de este campo es incorporada por cada MTA, por donde el mensaje pasa. Es útil para rastrear problemas de transporte.
- **To :** Contiene la dirección de la persona a la que queremos enviar el mensaje.
- **CC:** Contiene una dirección a la cual se desea enviar una copia del mensaje
- **Sender:** Contiene la identidad de quien envía el mensaje.
- **Subject :** Provee un resumen del contenido del mensaje
- **In-Reply-To:** Indica las direcciones en las cuales se quieren recibir las respuesta.

Ejemplos de headers en mensaje de mail :

```
From jdiaz@unlp.unlp.edu.ar Mon Oct 23 12:29:09 1995
Return-Path: jdiaz@unlp.unlp.edu.ar
Received: from unlp.unlp.edu.ar (unlp.unlp.edu.ar [163.10.0.67]) by isis.unlp.edu.ar
(8.6.11/8.6.9) with SMTP id MAA01542; Mon, 23 Oct 1995 12:29:09 -0200
Received: from amon-ra.unlp.edu.ar by unlp.unlp.edu.ar with SMTP id AA10351
(5.65c/IDA-1.4.4); Mon, 23 Oct 1995 12:21:58 -0300
Date: Mon, 23 Oct 1995 12:21:58 -0300
Message-Id: <199510231521.AA10351@unlp.unlp.edu.ar>
X-Sender: jdiaz@unlp
X-Mailer: Windows Eudora Version 1.4.4
Mime-Version: 1.0
Content-Type: text/plain; charset="us-ascii"
To: Miguel Luengo <mluengo@isis.unlp.edu.ar>
From: jdiaz@unlp.unlp.edu.ar (Javier Diaz)
Subject: Re: Recordatorio
Cc: jmarra@isis.unlp.edu.ar
Status: RO
```

<texto del mensaje.....>

MIME (Multipurpose Internet Mail Extentions - RFC 1521)

Introducción

La RFC 822 intento especificar un formato para mensaje de texto. Por lo que mensajes no textuales, tales como mensajes multimediales que pueden incluir audio, imágenes, tablas o formatos de procesadores de texto simplemente no son mencionados.

Aun en el caso de texto, la RFC 822 es inadecuada para mensajes que usan un conjunto de caracteres extendidos (lenguajes asiáticos y latinos por ejemplo).

Una de las mas notables limitaciones de los sistemas de mail basados en la RFC-822/821 restringen el contenido de los mensajes de mail a líneas relativamente cortas de ASCII 7 bits. Esto fuerza a los usuarios a convertir los datos no textuales en caracteres imprimibles codificados en 7 bits. Ejemplo de tal codificación en la comunidad Internet son uuencode, base64 y Andrew ToolKit Representation entre otras. Esto si bien resuelve el problema, hacen más complejo el envío y recuperación de mensajes.

Descripción

MIME describe varios mecanismos, que combinados, resuelven muchos de esos problemas sin introducir incompatibilidades con el mundo del mail de la RFC 822. En particular describe :

- Un campo que describe la versión MIME, para indica que el mensaje cumple con esta especificación. La versión actual es la 1.0. La presencia de este header asegura que el mensaje ha sido desarrollado cumpliendo con la RFC 1521(MIME).
- Un header que define el tipo de contenido (**Content-Type Header**), cuyo valor indica la clase de contenido en el cuerpo del mensaje.

Un **Content-Type** está identificado por:

1. Un tipo, el cual da una guía de los recursos requeridos para procesar el contenido,
2. Un subtipo el cual redefine el contenido
3. Y cero o mas parámetros que permiten personalizar el contenido.

Para la utilización de tipos y subtipos no estándar deben definirse los valores del campo con un string "X-" al comienzo de los mismos.

Los tipos predefinidos para el Content-Type son:

text, el cual es usado para representar información textual.

Hay dos subtipos:

plain: indica que el contenido es texto sin estructura.

richtext: indica que el contenido es texto con directivas para un lenguaje de formateo del mismo.

multipart, usado para combinar varias partes de cuerpos, posiblemente de diferente tipo, en un único mensaje. Cada parte es separada por un delimitador y es estructurada de una manera similar a un mensaje de mail.

Hay cuatro subtipos:

mixed: las distintas partes del cuerpo deben ser procesadas secuencialmente.

paralell: la partes del cuerpo deben ser procesadas en forma paralela.

digest: indica que las partes del cuerpo son mensajes de mail.

alternative: indica que hay múltiple partes con igual contenido semántico.

applications, es particularmente usada para datos a ser procesados por programas de aplicación basados en mail. Esta información debe ser procesada antes de ser presentadas o ser usada por un usuario. Los subtipos definidos son:

octet-stream: el contenido son datos binarios.

postscript: indica que el contenido es un programa PostScript.

message, permite encapsular otros mensajes de mail.

Hay tres subtipos:

rfc822: indica que el contenido es simplemente un mensaje de mail.

partial: indica que el contenido es un mensaje fragmentado.

external-body: indica que los datos del cuerpo no están incluidos, pero si referenciados, en este caso los parámetros describen los mecanismos para acceder a los datos.

image, es utilizado para transportar imágenes fijas, los subtipos son: **gif** y **jpeg**

audio, indica que el contenido son datos que representan sonido digitalizado. El único subtipo definido es **basic**.

video, es usado para indicar que el contenido son imágenes con movimiento. El subtipo definido es **mpeg**.

- Un header que indica la codificación del contenido para la transferencia (**Content-Transfer-Encoding Header**), el cual puede ser utilizado para especificar una codificación auxiliar, que se aplica a los datos para permitirle pasar a través de los mecanismos de transporte de mail, los cuales pueden tener un conjunto de caracteres limitados (ver transporte de mail)

Muchos tipos de contenidos son representados como datos binarios o caracteres en 8 bits. Tales datos no pueden ser transmitidos sobre algunos protocolos de transporte de mail. Por ejemplo la RFC 821 (SMTP) restringe los mensajes de mail a datos en 7 bits, cuya longitud de línea no supere los 1000 caracteres. En dicho caso es necesario definir algún mecanismo estándar para recodificar esos datos en líneas cortas de 7 bits.

La proliferación de valores para el Content-Transfer-Encoding no es deseable y además innecesario. Sin embargo establecer un único mecanismo no ha sido posible.

La necesidad de una codificación para grandes contenidos en formato binario, por un lado, y la necesidad de una codificación para datos en los cuales la mayoría de sus caracteres, pero no todos, están en 7 bits, generó dos tipos de codificación una “densa” y otra “legible”.

Se debe destacar que no existe una relación, a priori, entre el tipo de contenido y el tipo de codificación a utilizar, aunque algunos contenidos se adaptan mejor a cierto tipo de codificación. Los Content-Types **multipart** y **message** rechazan otro tipo de codificación diferente de 7bits.

Las actividades de codificación y descodificación debe verse separadas del procesamiento del contenido (Por ejemplo, imprimir un postscript o ejecutar una animación). Todo mensaje entrante debe ser descodificado a su forma nativa antes de ser procesado como un tipo particular de contenido.

Algunos de los tipos de codificación son:

quoted-printable: este esquema de codificación es utilizado cuando el contenido esta en su mayoría en NVT ASCII. Es muy útil cuando sólo un pequeño porcentaje de los caracteres tienen seteado el octavo bit.

base64: este esquema de codificación analiza el string de octetos cada 24 bits tomándolos en cuatro grupos de 6 bits. Cada valor de 6 bits es usado como un índice en una tabla de 64 caracteres para tomar el valor asociado. Si el stream de entrada no es múltiplo de 24 se completa con el carácter “=”.

7bit, 8bit y binary: todos significan que no se ha realizado codificación. Sin embargo es útil como indicación de la clase de datos contenidos en el objeto y así seleccionar la clase de codificación que puede ser necesaria para la transmisión en un sistema de transporte dado. En particular:

7bit: es el mecanismo de codificación por default e indica que el contenido esta en líneas cortas conforme al repertorio NVT ASCII.

8bit: significa que las líneas son cortas pero pueden existir caracteres cuyo octavo bit este seteado.

binary: significa que no solamente puede haber caracteres con el octavo bit seteado sino también que las líneas no son suficientemente cortas para el transporte SMTP.

- Dos headers adicionales (**Content-ID** y **Content-Descriptions Headers**) que permiten una mayor descripción de los datos del cuerpo de un mensaje.

Transporte de Mail

Como se ha mostrado en el modelo conceptual (Figura 1), en un sistema de mail conviven varios protocolos, como lo son el posting, el delivery y el relaying protocol. El delivery y el submission son tratados localmente y definen el transporte entre el U.A. y el MTA, mientras que el relaying protocol describe el transporte entre MTA's.

En la actualidad, dentro de la comunidad Internet, el submission y el relaying, es generalmente ofrecido por el estándar Simple Mail Transfer Protocol (SMTP) y el delivery es provisto por otro standard, el Post Office Protocol (POP).

Desde el punto de vista del e-mail los protocolos mencionados proveen toda la funcionalidad necesaria para el transporte de mensajería, razón por la cual las nuevas se basan en extensiones y no en el desarrollo de protocolos alternativos.

SMTP fue por muchos años el protocolo de transporte estándar, pero dada sus restricciones debió extenderse para permitir transportar mensajes cuyo cuerpo no esté compuesto sólo de caracteres del conjunto NVT ASCII.

Las restricciones impuestas por los servidores SMTP es que las líneas (conjunto de caracteres entre dos CR-LF) no deben superar los 1000 octetos, y los caracteres dentro de ellas deben estar en US-ASCII (7 bit). De haber caracteres en 8 bits, el bit de mayor orden es puesto en 0.

Los sistemas de mail basados en los RFC 822/821 funcionan correctamente aun con estas restricciones, pues la RFC-822 define el cuerpo de los mensajes como unas pocas líneas de caracteres US-ASCII (7 bits). La necesidad de manejar otro tipo de información en el cuerpo de los mensajes, llevó a la definición de MIME (RFC- 1521), lo que permitió incorporar en el cuerpo de los mensajes, información no textual, tales como gráficos, vídeo o sonido.

Aunque esa definición extendió el formato de los mensajes, SMTP mantenía las limitaciones mencionados anteriormente. Esto obligó a incorporar extensiones. Al protocolo SMTP más sus extensiones se lo denomina ESMTP (Extended SMTP). Se debe destacar que existen varias extensiones, tales como **MIME8**, **SIZE**, **EXPN**, **CHUNKING** y **MIMEBINARY** (estos 2 últimos experimentales aún) y los servidores SMTP pueden tener implementados todas ó algunas, razón por la cual antes de utilizar alguna de ellas, se debe consultar al servidor si la soporta. Esto se hace por medio del comando **EHLO** de ESMTP, que de estar implementado, retornará una lista de las extensiones soportadas por ese servidor SMTP.

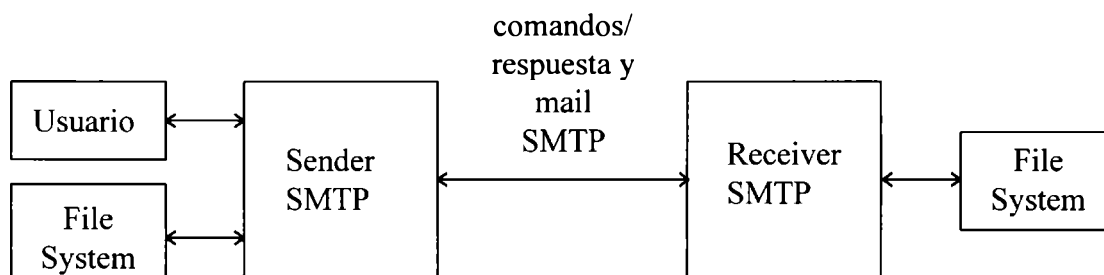
Volviendo a la relación que existe entre el contenido del cuerpo del mensaje a transmitir, y las condiciones impuesta por el transporte, podemos decir que si el contenido del cuerpo tiene caracteres fuera del repertorio NVT ASCII, depende entonces de las limitaciones del transporte. Si el transporte soporta sólo caracteres NVT ASCII se

debe codificar la información a 7 bits mediante algún algoritmo de codificación tal como **uuencode**, **quoted-printable** o **base64**. Esta codificación produce un gran overhead en el transporte.

Si el transporte soporta la extensión **MIME8**, entonces se pueden enviar caracteres ASCII con valores superiores al 127. Esto solucionará parcialmente el problema, dado que la longitud de línea sigue limitada a unos 1000 octetos. Si se está transmitiendo un objeto binario, la longitud de líneas puede superar largamente lo permitido por SMTP, por lo tanto deberá también ser codificado. Por tal motivo dos nuevas extensiones de SMTP que funcionan conjuntamente se definieron (en forma experimental hasta Agosto de 95) para transportar información binaria: el **CHUNKING** y el **MIMEBINARY**. Estos permiten enviar objetos binarios en “trozos” sin necesidad de codificación.

A continuación se detallaran los protocolos más importantes del mail Internet y algunas de sus extensiones.

Simple Mail Transfer Protocol (SMTP) (RFC-821)



El SMTP es el standard de transporte de mail en la comunidad Internet y fue definido por la RFC-821. Este protocolo transporta tanto el sobre como el contenido del mensaje. El sobre SMTP deberá contener:

- La dirección de mail del emisor que se corresponde, si existe, con el campo Sender en el header.
- Una o más direcciones de destino junto con una indicación del modo de entrega.

El modo de entrega es, generalmente, el mailbox (buzón) de usuario, aunque la posibilidad del envío directo a terminal es considerada también en la definición .

Cuando un MTA determina que no puede transmitir o entregar el mensaje a uno o más recipientes se genera un nuevo mensaje, llamado reporte de error. El sobre del mensaje contiene como dirección origen el string vacío y como dirección destino, la dirección origen del viejo mensaje. El cuerpo del mensaje es cualquier texto de diagnóstico. La falta de un estructura formal del contenido de un reporte de error a

menudo requiere ser interpretado por un humano (*). Es importante notar que nunca un mensaje de error debe generar otro mensaje de error.

Generalmente asumimos que la relación entre dirección de mailbox y la dirección en el mensaje es uno a uno, sin embargo no siempre es así. Cuando un MTA acepta la responsabilidad para una dirección local esta puede ser un alias, la cual puede resolverse en una o más direcciones de mail denominadas valor de alias. Cualquier número de estas direcciones de mail puede ser un recipiente local o podría ser alcanzado a través de un MTA remoto. El manejo de los alias es una cuestión local y esta relacionado con la configuración del MTA local.

Cuando un MTA encuentra un alias local como dirección de destino en un sobre, éste la reemplaza con la dirección de mail dada por el valor de ese alias. Esto es llamado expansión de alias (*alias expansion*). Algunos alias son especiales, ya que son listas de mail (*mailing list*), las cuales están formadas por un conjunto de una o más direcciones de mail con un administrador. Cuando se encuentra un alias correspondiente a un mailing list como dirección de recipiente en el sobre, entonces ocurre un *list explosion*. Se crea un nuevo sobre con idéntico contenido y el nuevo emisor pasa a ser la dirección del administrador de la mailing list. Las direcciones de los recipientes en el sobre son establecidas a los valores de alias (Subscriptores).

Los U.A. pueden proveer también facilidad de alias, sin embargo, estos alias serán expandidos antes de que los mensajes sean entregados al MTA.

(*) *Ver DSN*

Interacciones del protocolo

Una interacción SMTP es straight-forward. Un servidor SMTP espera en el port TCP 25 que un cliente establezca una conexión. El servidor “saluda” indicando el estado de su MTA local. Si el estado es aceptado por ambas partes, el cliente se identifica y luego inicia una o más transacciones SMTP (Comandos y respuestas). Cuando el cliente finaliza, emite un comando para liberar el servicio y espera que el servidor retorne una respuesta y cierre la conexión TCP. Luego el cliente cierra también la conexión.

Cada comando SMTP consiste de una palabra clave, que puede en algunos casos, estar seguido de un argumento. Algunos de estos argumentos son una dirección e-mail, la cual era especificada en forma completa, indicando el full path para llegar al destino. En la actualidad, esta modalidad ha sido reemplazada por la notación user@domain haciendo uso de las facilidades de DNS para el ruteo de mails.

Los códigos de respuestas SMTP fueron ideadas para asegurar la sincronización de los requerimientos y las acciones en los procesos de transferencia de mail y además, para garantizar que el “Sender” SMTP permita conocer el estado del “Receiver”. Todo comando deberá generar solo una respuesta, la cual consiste de tres dígitos numéricos seguidos por algún texto. El número está pensado para ser procesado automáticamente y

determina la secuencia de estados que se debe seguir, mientras que el texto se utiliza para interpretación por parte de los humanos.

Una respuesta está formada por un código, seguido de un espacio y una línea de texto, finalizado por un CRLF (Carriage Return-Line Feed). También la respuesta puede ser multilínea.

La sintaxis de los comandos y el análisis de los códigos de respuesta se expresan en detalle en Anexo B.

Extended Simple Mail Transfer Protocol. (ESMTP) (RFC-1651)

El SMTP ha sido por mucho tiempo un estable y efectivo protocolo de transmisión entre MTA's. Sin embargo la necesidad de nuevas extensiones fue un echo evidente. Entre las necesidades están las de enviar mails que contengan gráficos, sonido o texto formateado; también la de recibir notificación que permitan procesamiento automático y no solo en el caso de errores.

Recordemos que SMTP transmite objetos de mail que contienen :

- Un “**sobre**” , que es de transmisión directa y está compuesto por una serie de unidades de protocolos, tal como una dirección de origen (a la cual los reportes de error deben ser dirigidos); un modo de entrega (por ej. mailbox o terminal); y una o más direcciones receptoras.
- Un “**contenido**” , que es enviado por la unidad de protocolo DATA y tiene dos partes: headers y cuerpo. Los headers son un conjunto de pares campo/valor, estructurados de acuerdo a la RFC-822, mientras que el cuerpo no lo es. El contenido es por naturaleza textual y se usa US-ASCII para su representación. El cuerpo puede estructurarse de acuerdo a extensiones tales como MIME, lo que permite, también, extender su representación más allá del US-ASCII.

Una extensión define:

1. Un nuevo comando (**EHLO**) , por medio del cual un cliente puede consultar que extensiones soporta un servidor SMTP.
2. Una registración en la IANA, organismo que se encarga de mantener el estándar. A cada nueva extensión autorizada, le asignará un valor de respuesta al comando EHLO.
3. Parámetros adicionales para los comandos **MAIL FROM** y **RCPT TO**.

Las extensiones analizadas en este proyecto serán **MIME8transport**, **DSN** (Delivery Status Notifications) y **CHUNKING**.

Servicio de Extensión SMTP para transporte 8bit-MIME

Introducción.

En párrafos anteriores se mencionaron las limitaciones de SMTP para transportar información en 8 bits. Se describirá un mecanismo que define una extensión, por la cual SMTP podrá intercambiar ese tipo de datos. Debe notarse que esta extensión no elimina la limitación de la longitud de línea a 1000 octetos, por tal razón no provee un medio para transferir datos binarios sin codificación vía SMTP.

Definición

1. La extensión se llamará **8bit-MIMEtransport** ;
2. El valor asociado con la extensión al comando EHLO, será **8BITMIME**;
3. Se agrega un nuevo parámetro BODY al comando MAIL FROM. El valor asociado a ese parámetro indica si es un mensaje 7bit o MIME:

valor ::= "7BIT" / "8BITMIME"

Cuando un cliente SMTP desea enviar el cuerpo de un mensaje conteniendo caracteres en 8 bits, primero debe transmitir el comando EHLO al servidor SMTP. Si el servidor responde con una respuesta positiva (código 250) y la respuesta contiene la palabra clave 8BITMIME, entonces el servidor está indicando que soporta la extensión y aceptará el mensaje MIME con caracteres en 8 bits. Si la respuesta al comando EHLO no es positiva o no incluye la palabra clave 8BITMIME, el cliente debe efectuar una transformación para convertir el mensaje a 7 bits o puede emitir un error permanente y manejarlo como una falla de entrega.

Ejemplo

```
C: EHLO dbc.mtview.ca.us
S: 250- isis.unlp.edu.ar say hello
S: 250 8BITMIME
C: MAIL FROM : <user@isis.unlp.edu.ar > BODY=8BITMIME
S: 250 <user@isis.unlp.edu.ar >... Sender and 8BITMIME ok
C: RCPT TO : <mrose@dbc.mtview.ca.us>
S: 250 <mrose@dbc.mtview.ca.us>... Recipient ok
C: DATA
S: 354 Send 8BITMIME message, ending in CRLF.CRLF

C: .
S: 250 ok
C: QUIT
S: 250 Goodbye
```

Delivery Status Notifications (DSN).

Introducción.

Una queja común entre las organizaciones que tienen que usar o evaluar el potencial uso del mail Internet como un vínculo confiable para mensajería, ha sido la falta de un mecanismo de notificación de entrega estandarizado, similar a lo que ofrecen todas las implementaciones X.400.

La Internet Engineering Task Force (IETF) responde a esta demanda creando el NOTARY Working Group a principio de 1994. El propósito de este grupo es identificar y desarrollar “ las mejores herramientas para la construcción de un sistema, donde el *sender* (emisor) del mensaje pueda saber que pasó con él” . Estas incluyen los requerimientos para un reporte formateado, basado en MIME, el cual puede ser usado para reportar entrega, no entrega y acuse de recibo.

El estado actual

Aunque la especificación original del protocolo SMTP, requiere del servidor SMTP receptor para proveer notificación de falla de entrega, no existe especificación para el formato de tal notificación. Esto significa que cada dominio tiene la libertad para crear su propio formato para texto y códigos de error. Generalmente estos son solo entendibles para los humanos, no apto para el procesamiento automático. El uso incremental de listas de distribución enfatizan la necesidad de procesar las notificaciones en forma automática. También los mensajes de falla de entrega son indistinguibles de un mensaje normal, lo cual complica aún más las cosas. Actualmente no hay forma de que el sender de un mensaje SMTP, pueda obtener una notificación de entrega positiva o a requerir la entrega del mensaje original junto con una notificación de que no ha sido entregado. Actualmente se retorna información del header o parte del texto del mensaje.

La nueva funcionalidad incluye lo siguiente:

- Un cliente ESMTP requiere a un servidor ESMTP que retorne notificaciones de estado de entregas (DSN).
- El servidor ESMTP retorna los DNS requeridos, empaquetados como un contenido MIME .

Como los DSN incluyen códigos de error extendidos para mensajes no entregados, estos son presentados en un formato estandarizado, el U.A. original o el gateway pueden ahora entender el DSN que es devuelto y opcionalmente, procesarlo automáticamente. Note que el nuevo formato puede también ser usado para retornar notificaciones de mensajes no entregados sobre el actual protocolo SMTP.

La especificación NOTARY tiene en cuenta que el mundo real está compuesto totalmente por gateway y trata de hacer la convivencia entre distintos sistemas lo más armonioso posible.

Los documentos NOTARY

Las nuevas funcionalidades han sido descritas en cuatro publicaciones:

- Extensión de servicios SMTP para notificación de estados de entrega.
- Un formato de mensaje extensible para notificación de estados de entrega de mensajes
- Tipo de contenido Multipart/Report para usar por un sistema de mail administrativo de mensajes.
- Extensión de los códigos de estado del sistema de mail.

La extensión de servicios SMTP para notificación de estados de entrega describe la habilidad del cliente que envía un mensaje para requerir notificaciones negativas, positivas o ambas (el default es como antes, solo casos negativos), como así también requerir el retorno o no del mensaje original. En resumen, el documento define la responsabilidad del servidor receptor para actuar sobre los requerimientos del servidor emisor, por ejemplo: para retornar lo que se pregunta y nada mas, y nunca retornar ambas notificaciones (satisfactoria y falla), en casos donde ambos son requeridos. Suficiente información debe retornarse para permitir al enviador original una única identificación del mensaje y determinar el recipiente, de la misma manera si el mensaje paso por un gateway a un sistema no SMTP.

Técnicamente, las extensiones para un servidor SMTP incluyen un nuevo valor asociado con comando EHLO (DNS) , el cual indica en su respuesta si el servidor entiende DSN extensions.

Dos nuevos parámetros opcionales para el comando RCPT: "NOTIFY" y "ORCPT". Donde NOTIFY contiene el tipo de notificación requerida y ORCPT contiene el tipo dirección original especificada por el usuario

**RCPT TO: <user@domain> NOTIFY=SUCCESS
ORCPT=rfc822;user@domain**

donde:

NOTIFY puede tomar los valores : NEVER , SUCCESS, FAILURE y DELAY.

ORCPT puede tomar cualquier tipo de dirección de mail registrada en la IANA.

Dos nuevos parámetros opcionales para el comando MAIL: "RET" y "ENVID". RET contiene que información será retornada (ya sea el mensaje completo o solo el encabezado) en el caso de no entrega; mientras que el parámetro ENVID, contiene información de identificación que el sender desea retornar como parte de los DSN's.

MAIL FROM: < user@domain > RET=HDRS ENVID=QQ314159

donde:

RET: puede tomar los valores FULL para retornar todo el mensaje o HDR para retornar los headers.

ENVID: permite cualquier texto que será utilizado como identificador de sobre.

Un cliente ESMTP puede chequear al comienzo de la sesión si un servidor SMTP receptor soporta DSN.

Un formato de mensaje extensible para entrega de notificación de estado, conlleva la definición de un nuevo tipo de componente MIME, diseñado para transportar información DSN de retorno a quien la requiere y permitir a los mensajes DSN ser detectados y si se desea, procesarlo automáticamente.

El formato DSN MIME permite el acarreo de formas de direcciones de mail y códigos de error extraños al Mail Internet (tal como X.400 y sistemas de mail propietarios).

Esto significa que la información puede ser transportada a través de gateways o “tunneled” a través del sistema de mail Internet. El formato de mensajes DSN MIME tiene un content-type del tipo Multipart/Report.

Este puede contener hasta tres componentes:

- Un texto explicativo comprensible por humanos (Requerido)
- Un capa más formal, que permite el procesamiento por máquina. Por ejemplo estadísticas de errores o acciones automáticas para la distribución de software de manejo de listas. (Requerido)
- Un header o mensaje completo opcional es retornado, si es requerido en situaciones de no entrega.

La parte extensible del nuevo formato, se refiere a la habilidad para extender el formato a otro tipo de reporte que puede ser desarrollado en el futuro.

Finalmente la Extensión de los códigos de estado del sistema de mail contiene un nuevo conjunto extendido de códigos de error, los cuales pueden ser usados para reportar razones de no entrega. Estos permitirán hacer el seguimiento de errores y la comunicación de errores en el sistema de mail Internet mucho más fácil, una vez implementado en gran escala. La impresión general es que este cubre no solo la misma base correspondiente a códigos X.400, sino también agrega un número de situaciones de error que están asociadas con el transporte de mail sobre diferentes sistemas (gateways).

Sin bien el grupo NOTARY considera que todo lo propuesto previamente fue realizado, existen facilidades que podrían ser aún requeridas. Por ejemplo la funcionalidad de Mensajería Interpersonal de X.400, tal como “receipts and non-receipts” y la mensajería EDI de X.400 tal como, los servicios “Responsability Notification”.

El proyecto NOTARY, deja a los usuarios del Internet Mail con la necesidad de saber si el mensaje ha sido puesto dentro del mailbox del usuario fue leído o no. Debe destacarse que la funcionalidad está limitada al protocolo SMTP y no cubre los protocolos de acceso al mailbox tales como POP e IMAP.

Servicio de Extensión SMTP para transmisión de Mensajes Extensos y Mensajes Binarios.

Introducción

Como se vio en las extensiones mencionadas anteriormente, las limitaciones para el envío de objetos binarios no codificado en el cuerpo de los mensajes siguen existiendo.

Estas dos nuevas extensiones, que aún son experimentales, permitirán eliminar esa limitación.

Independientemente de la necesidad de enviar mensajes extensos, la proliferación de equipos con soporte multimedial, generan la necesidad de manejar mensajes conteniendo información no textual (imágenes, animaciones, sonido) en el correo de la Internet. El hecho de codificar esa información (por ejemplo con quoted-printable o base64) agrega un overhead importante sobre el transporte. Por esas razones se definió sobre el estándar SMTP una especificación que evita este inconveniente.

Definición

La definición de la extensión es:

1. El nombre del servicio de extensión es "**CHUNKING**".
2. La palabra clave asociada al comando EHLO es "**CHUNKING**".
3. Un nuevo comando SMTP, "**BDAT**", alternativo a "DATA" es definido. El comando puede tomar dos parámetros, uno indicando la longitud del packet de datos binarios y el otro opcional indicando que el packet es el último.

Esta extensión puede usarse con cualquier otro tipo de mensajes (7BIT, 8BITMIME o BINARYMIME).

Cuando un cliente desea enviar un mensaje usando esta extensión, primero debe transmitir el comando EHLO. El servidor SMTP debe responderle con un código de respuesta positiva (250) e incluir el valor CHUNKING, para indicar que soporta el comando BDAT.

Después de que fueron procesados todos los MAIL FROM y RCPT TO, el mensaje debe ser enviado utilizando una serie de comandos BDAT. Debe tenerse en cuenta que no se puede utilizar BDAT y DATA en la misma transacción. Además, debe considerarse la importancia de indicar en el parámetro la longitud exacta del mensaje,

dado que el servidor considera la terminación de los datos por medio del conteo de bytes.

La segunda extensión que acompaña a CHUNKING es definida de la siguiente manera:

- El nombre del servicio de extensión es “**BINARYMIME**”.
- La palabra clave asociada al comando EHLO es “**BINARYMIME**”.
- La extensión solo puede ser utilizada con la extensión CHUNKING.
- Se agrega el nuevo valor “**BINARYMIME**” al parámetro BODY del comando MAIL TO e indica que el mensaje a enviar es un mensaje MIME en binario.

De la definición podemos deducir que la respuesta del servidor SMTP al comando EHLO, emitido por el cliente, debe incluir los valores “CHUNKING” y “BINARYMIME”.

Ejemplo de una interacción entre un cliente (C) y un servidor (S), usando las extensiones descritas previamente:

```
R: <Espera la conexión en el TCP port 25>
S: <Abre la conexión al servidor>
R: 220 cnri.reston.va.us SMTP service ready
S: EHLO ymir.claremont.edu
R: 250-cnri.reston.va.us says hello
R: 250-BINARYMIME
R: 250 CHUNKING
S: MAIL FROM:<ned@ymir.claremont.edu> BODY=BINARYMIME
S: RCPT TO:<gvaudre@cnri.reston.va.us>
S: RCPT TO:<jstewart@cnri.reston.va.us>
R: 250 <ned@ymir.claremont.edu>... Sender and BINARYMIME ok
R: 250 <gvaudre@cnri.reston.va.us>... Recipient ok
R: 250 <jstewart@cnri.reston.va.us>... Recipient ok
S: BDAT 100000
S: (First 10000 octets of canonical MIME message data)
S: BDAT 324 LAST
S: (Remaining 324 octets of canonical MIME message data)
R: 250 100000 bytes received
R: 250 Message OK, 100324 octets received
S: QUIT
R: 221 Goodbye
```

UUCP

En la red Unix-to-Unix copy (UUCP), sus hosts se comunican por medio de ejecución remota, básicamente sobre líneas asincrónicas (actualmente también sobre TCP). Las interacciones son straight-forward. Sobre y contenido son transferidos como entrada a un comando el cual se ejecuta sobre un nodo adyacente. El comando examina el sobre y determina si la entrega es final o debe retransmitirlo. Un nombre de un nodo UUCP es identificado por un string alfanumérico de ocho caracteres.

Hoy día la mayoría de la comunidad UUCP utiliza el DNS para identificar los recipientes de mensajes y tablas de ruteo para resolver la conexión inmediata siguiente. Así cuando un MTA en un nodo UUCP desea transmitir un mensaje de mail debe entregar dos clases de información relacionada: cuales son los nodos adyacentes mas cercanos y que comando ejecutar en uno de esos nodos.

Para determinar el próximo MTA a alcanzar, el nodo necesita un método para mapear un nombre de dominio en el nombre del nodo UUCP próximo o bien establecer uno por default. El MTA entonces ejecuta el programa **uux** identificando el comando a ejecutar y el nombre del nodo UUCP donde debe realizar la acción. La entrada para el comando es la información del sobre especificada en una o mas líneas, seguido inmediatamente por el contenido, el cual es un mensaje RFC-822.

Transporte de artículos news

La USENET fue una red de noticias creada por las Universidades de Duke y Carolina del Norte. Esta consta de cientos de grupos (Newsgroup) con una gran variedad de temas. Hay grupos de las más diversas actividades que van desde el interés en un sistema operativo a la compra-venta de cosas [O'REILLY 90].

A diferencia de los mensajes de mail, los cuales son direccionados a un número fijo de recipientes, un artículo News alcanza a toda una comunidad de interés mediante un algoritmo de flooding.

Cada nodo USENET es configurado con una lista de nodos, con los cuales él intercambia artículos News.

Hay varias formas de transmitir los artículos en la comunidad USENET: una forma es transferir News usando la ejecución remota del UUCP. Por convención el comando rnews se utiliza para este propósito. El argumento a este comando es el artículo.

Otra forma, es encapsular artículos News en mensajes de mail y enviarlos a direcciones de mail especiales a un nodo adyacente. Por convención el recipiente rnews es utilizado para este propósito. El esquema de encapsulación usado es muy simple: una letra N es puesta al comienzo de cada línea del artículo antes de la transmisión y removida por el recipiente rnews.

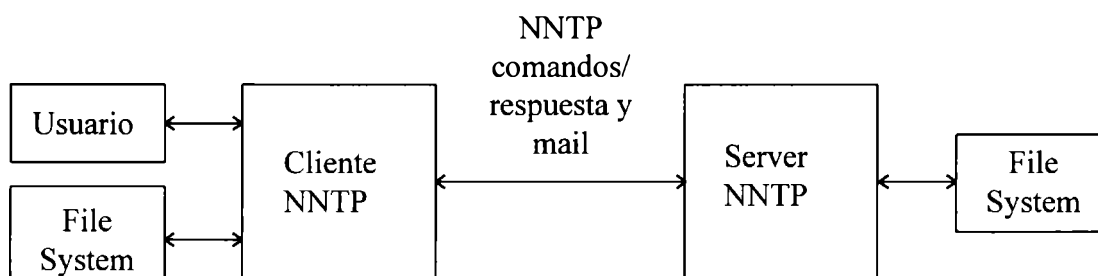
Esta dos formas de transferencias de news son aproximaciones basadas en transacciones.

La diferencia conceptual entre listas de discusión y grupo de interés es que en las listas, la opinión de una persona llega (cuando lo desea y está habilitado) a todos los subscriptores. Entonces un abonado encuentra en su mailbox información que originan los demás en número y volumen indeterminado. Usa como servicio básico el correo electrónico, significando que es simple de usar, pero tiene el inconveniente de mezclarse en el mailbox y no poseer mecanismos de compactación ni forma de desactivarse transitoriamente cuando el lector, por ejemplo, se va de vacaciones.

El grupo de News o de interés requiere que el lector se conecte a un servidor ("Mayorista") y lea las novedades que llegaron:

- el último día
- desde la última vez que se conectó y
- recibiendo tanta información como exista registro histórico mantenido en el servidor

En general, podemos decir que en un sistema de News se da una discusión global en un foro abierto.

NNTP (Network News Transfer Protocol).

NNTP especifica un protocolo para la distribución, consulta, recuperación y envío de artículos News usando un canal de transmisión confiable.

NNTP está diseñado de manera que los artículos son almacenados en una base de datos central, permitiendo a los suscriptores seleccionar aquellos ítems que son de su interés. También provee indexación, referencias cruzadas y expiración de mensajes antiguos.

El servicio de News provee a la comunidad de lectores una rápida diseminación (broadcast) de ítems de interés, tales como bugs de software, revisión de nuevos productos, nuevas tecnologías y discusiones rápidas de problemas.

Hay dos métodos populares de distribuir News:

1. El método de mensajería directa de Internet.
2. El sistema Usenet.

En el primero, la comunidad Internet distribuye artículos usando lista de mail (mailing list). Esta son lista de direcciones de los suscriptores y sublista de reenvío. Dicha listas operan reenviando una copia de la información a ser distribuida a cada suscriptor de la misma. Esto se torna ineficiente cuando las lista comienzan a crecer, dado que enviar copias a cada suscriptor ocupa grandes cantidades de CPU, ancho de banda de la red y espacio en disco en el host. Además se hace significativo el problema de mantenimiento de lista cuando los suscriptores se mueven de una lista a otra, ingresan nuevos u otros se van.

En la Usenet los artículos están almacenados en una base de datos central y se proveen las herramientas de acceso a los mismos. En grupos de host conectados en una LAN de alta velocidad (Por ej. Ethernet), conviene concentrar los artículos en uno de ellos y accederlos utilizando el modelo cliente-servidor.



Típicamente el servidor NNTP corre en background en un host e irá aceptando las conexiones de otros host de la red.

En ambientes donde los host tienen muchos usuarios, tales como universidades o centros industriales, pueden usarse servidores intermedios que actúan de mediadores de requerimientos de lectura de News y hacen "caching" de artículos recientemente recuperados.

Las interacciones del protocolo entre cliente y servidor es similar al SMTP,. El cliente envía comandos y el servidor responde con respuestas encabezadas con un código de tres dígitos que indica el resultado de la ejecución del comando (erróneo, satisfactorio, parcial y errónea transitoria)

La parte principal de USENET utiliza para el volcado de News, el Network News Transfer Protocol (NNTP) , el cual tiene tres servicios:

- Identificación de nuevos newsgroup creados en el servidor NNTP .
- Identificar y opcionalmente transferir artículos News arribados al servidor NNTP.
- Identificar y opcionalmente transfiere artículos News arribados al cliente NNTP.

La sintaxis de los comandos es significativamente más compleja que la utilizada por SMTP. Lo mismo ocurre con el análisis de sus respuestas.

El texto de los artículos sigue las especificaciones de la RFC-850, la cual no es un standard de la comunidad Internet, pero NNTP se ajusta a ella. Cuando un artículo es enviado (headers y body) sigue las pautas del comando DATA de SMTP.

Cada vez que un News llega a un newsgroup el servidor NNTP incrementa un contador y se le asigna ese valor como número de artículo local en ese grupo. Además el servidor mantiene el número mayor y menor de artículo local para cada newsgroup: cuando un News expira el número menor incrementa y cuando un News arriba incrementa el mayor.

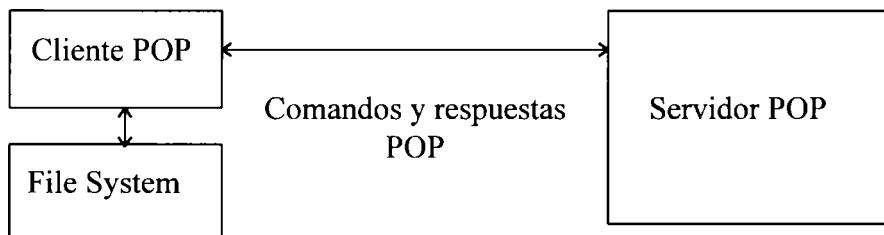
Servicio de mailbox

Volviendo al modelo conceptual, consideremos ahora el *delivery protocol*, utilizado entre el U.A. y el MTA en el momento de la entrega final, la cual dentro de la comunidad Internet, es una cuestión local.

La entrega final es realizada cuando el MTA determina que está lógicamente conectado al recipiente del usuario.

Desde la perspectiva del modelo, la entrega al mailbox es función del U.A. , sin embargo desde la implementación uno puede ver al MTA entregando el mensaje a través de delivery slot al mailbox.

Post Office Protocol (POP)



Para pequeños nodos en Internet es poco práctico o imposible mantener un MTS y permanecer interconectado mucho tiempo a una red estilo IP. A pesar de todo, es útil poder manejar mail sobre estos pequeños nodos y además soportar un U.A. para ayudar con el manejo de mensajes.

Para subsanar estos problemas, un nodo, el cual puede soportar una entidad MTS, deberá ofrecer un servicio de maildrop para aquellos nodos con menos recursos. Esto es. permitir acceso temporal para integrarse al sistema de mensajería.

El POP está pensado para permitir a una Workstation acceder dinámicamente al servicio de maildrop sobre un servidor. Esto significa que POP permitirá a tal workstation recuperar los mails que el servidor retiene para ella.

El POP es usado para proveer acceso remoto a un mailbox. Este puede verse residiendo tanto en el MTA como en el U.A.

Si el servicio POP es parte del MTA, los mensajes son almacenados con el servicio hasta que el U.A. acepta la entrega.

Si el servicio POP es parte del U.A., el mensaje es pasado a través del slot al mailbox del usuario, el cual será aceptado para entrega por el servicio POP. Así una

mitad del U.A. reside en el mismo sistema junto con el MTA y la otra mitad actúa como cliente para recuperar mensajes desde el mailbox del usuario (ver split-UA).

En particular, la versión de POP utilizada es la 3 (POP3), la cual varía muy poco de la anteriores. Los cambios, en general se dan en aspectos relacionados y no funcionales.

Funcionamiento básico del POP

Inicialmente el servidor inicia el servicio de POP3 esperando en el port TCP 110. Cuando un cliente desea usar el servicio establece una conexión TCP con el servidor, una vez establecida dicha conexión, el servidor POP3 envía un “saludo” (greeting). El cliente y el servidor POP intercambian comando y respuestas (respectivamente) hasta que la conexión es cerrada o abortada.

Los comandos en POP3 están compuestos de una palabra reservada, seguido un par CRLF (Carriage Return- Line Feed), mientras que las respuestas están conformadas por un indicador de resultado seguido de una palabra clave y un CRLF. El indicador de resultado toma dos valores posibles : uno positivo (“+OK”) o uno negativo (“-ERR”). Hay respuestas que pueden ser multilineas, donde la última línea está compuesta por un octeto de terminación formado por “.CRLF” (punto seguido de CRLF).

Una sesión POP3 pasa por distintos estados durante su tiempo de vida. Cuando la conexión TCP ha sido abierta y el servidor POP3 envió el saludo, la sesión entra en estado de **Autorización (Authorization)**, en el cual el cliente deberá identificarse ante el servidor. Una vez realizada satisfactoriamente la identificación, el servidor adquiere los recursos asociados con el maildrop del cliente y la sesión entra en un estado de **Transacción (Transaction)**. En este estado, el cliente requiere acciones al servidor.

Finalmente entra en un estado de **Actualización (Update)**, donde el servidor libera cualquier recurso adquirido durante el estado de transacción y envía una señal de despedida, cerrando la conexión TCP/IP.

La sintaxis de los comandos y las respuestas están especificadas **Anexo B**.

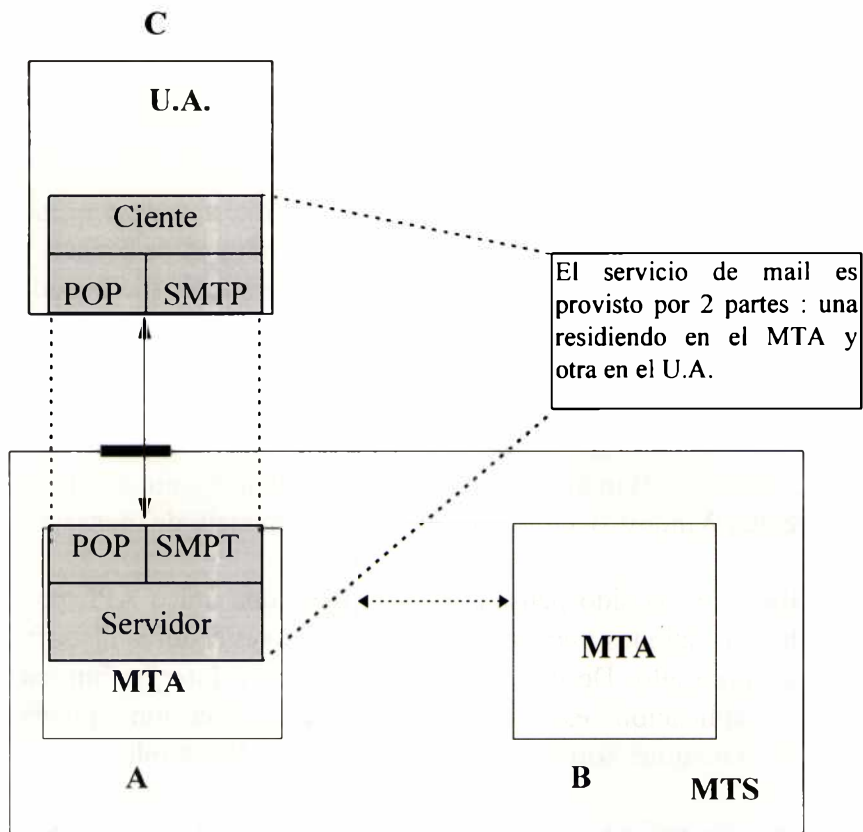
El modelo Split-U.A.

El paradigma sobre el cual funciona POP es llamado Split-U.A.

¿Qué es un Split-U.A. ? El host POP que actúa como un cliente (llamado **C** en la figura) para el MTS, no provee servicio de entrega/autenticación. Por lo tanto, actúa solo como un U.A.; utilizando SMTP para introducir el mail dentro del MTS. Por lo tanto hay dos U.A. funcionando como interfaz del MTS, uno realizando la entrega o posting (SMTP) y el otro realizando la recuperación o retrieval (POP). La entidad que

soporta este tipo de ambiente es llamado *Split-U.A.*, ya que el User Agent está dividido en dos host, los cuales deberán interoperar para proveer la funcionalidad deseada.

También se la conoce como *remote U.A.*



Criterio de diseño.

Antes, conceptualmente el MTA debía llegar hasta la computadora del usuario (que debía estar en línea), con POP3 la funcionalidad se separa llegando los mensajes a un servidor departamental siempre en línea y permitiendo que la distribución final de los mensajes se haga bajo demanda, lo cual asegura una conexión exitosa.

Los objetivos de diseño del POP3 son dos :

- Minimizar la complejidad del MTA por medio de varios métodos de acceso simple a los mensajes de mail , que han sido almacenados en los mailbox.
- Reducir al mínimo la inteligencia requerida por parte del cliente.

INTERFAZ WINDOWS SOCKETS.

Introducción.

La especificación Windows Sockets define una interfaz de programación para redes sobre Microsoft Windows, la cual se basa en el paradigma de sockets desarrollado en la Universidad de California en Berkeley . La interfaz socket fue introducida con el Unix 4.2 BSD en el año 1983 y fue actualizada por la versión 4.3 BSD abril de 1986. [COMER 93]

Esta comprende, tanto las rutinas estilo Berkeley Sockets, como un conjunto de extensiones específicas de Windows, que le permiten al programador tomar ventajas propias del ambiente (Windows), en lo que se refiere al manejo de mensajes.

La especificación ha sido pensada para proveer una única API, por lo cual los desarrolladores de aplicaciones pueden programar y los vendedores de software de red pueden concordar sobre ello. De esta manera se define un interfaz binaria (ABI) que permite que una aplicación escrita sobre esta especificación, pueda funcionar correctamente sobre cualquier software de red que también la cumpla.

El software de red que cumpla la especificación Windows Sockets se lo denomina "Windows Sockets Compliant" y para serlo deberá cumplir el 100% de la especificación.

Este proyecto se ha desarrollado sobre Trumpet Winsock versión 1.1, que considera el Internet protocol suite (IPS, conocido generalmente como TCP/IP), soportando sockets del tipo STREAM (TCP) y DATAGRAM (UDP).

Además de Trumpet Winsock se testeó la compatibilidad de las unidades de protocolos implementadas con Pathway Winsock de Wollongong Group.

Existe una versión posterior, la 2.0, que al mes de Julio de 1995 se encontraba en draft (provisional) , la cual extendió el acceso a otros protocolos como los son IPX/SPX, DECNet y OSI, y además incluye un protocolo adicional a algunos de estos (los que ofrecen servicios confiables orientados a conexión) para seguridad, el Secure Socket Layer (SSL) protocol. La especificación Windows Sockets 2.0 es compatible con la versión anterior, pero se deben cambiar las DLL y los archivos de headers en las implementaciones.

Conceptos básicos.

La unidad básica para la comunicación es el socket. Un socket es un punto de comunicación al cual se le puede dar un nombre. Cada uno tiene un tipo y un proceso asociado y existen dentro de un dominio de comunicación.

Un dominio de comunicación es una abstracción introducida para agrupar propiedades comunes de threads comunicándose por medio de sockets.

Los sockets normalmente intercambian datos solo con sockets en el mismo dominio (Es posible atravesar los límites pero solo si se realiza una translación).

Las facilidades de Windows Sockets soportan un único dominio de comunicación: el dominio Internet, el cual es usado por los procesos para comunicarse usando el Internet protocol suite (versión 1.1).

Hay dos tipo de sockets :

1. **Stream Sockets** : Provee un flujo de datos bidireccional, confiable, secuenciado, no duplicado y sin tamaño fijo de registro.
2. **Datagram Sockets** : Soporta flujo bidireccional de datos, el cual no es necesariamente secuenciado, confiable y no duplicado. Una característica de los datagramas es que preserva el límite de registro.

Modelo Cliente-Servidor.

El paradigma comúnmente utilizado en la construcción de aplicaciones distribuidas es el modelo Cliente-Servidor. En este esquema, aplicaciones clientes requieren servicios desde una aplicación servidora.

El cliente y el servidor requiere un conjunto de convenciones antes que el servicio pueda realizarse. Este conjunto de convenciones comprenden un protocolo el cual debe ser implementado a ambos extremos de la conexión.

El protocolo puede ser simétrico o asimétrico, en el primer caso ambos lados pueden cumplir los dos roles (Ej. TELNET); en el segundo caso un lado es el cliente y el otro es el servidor (Ej. POP).

Una aplicación servidor normalmente “escucha” sobre una dirección todos los servicios requeridos. Esto significa que el servidor se mantiene inactivo hasta que una conexión es requerida por el cliente . En ese momento el servidor se “despierta” y atiende el requerimiento.

Se debe tener en cuenta que el modelo cliente/servidor es un modelo de computación, donde el cliente y el servidor pueden estar en el mismo o en distintos hosts.

Para el alcance de este proyecto generalmente el cliente está en un equipo menos dotado de recursos y el servidor es un hosts remoto que actúa como mail server. Ambos equipos están conectados por un canal confiable de comunicación provisto por TCP/IP.

PARTE II

IMPLEMENTACION DE LOS PROTOCOLOS

Implementación.

Esta sección describe las funciones que definen los protocolos de transporte en un sistema de mensajería. Los protocolos funcionalmente descriptos son : POP3, SMTP y NNTP.

Cada función indica que archivos de headers debe incluir. Debe destacarse que la implementación considera los parámetros necesarios desde el punto de vista de un cliente. Además, debe considerarse que hay funciones coincidentes en los tres protocolos, las cuales fueron puestas para dar completitud . Los nombres de las funciones fueron seleccionados de acuerdo al nombre especificado por la definición de los respectivos protocolos.

POP3

DoUSER()

Descripción : Se utiliza para identificar la cuenta un usuario.

```
#include <winsock.h>
#include <pop3.h>
```

BOOLEAN DoUSER(SOCKET *skt*, LPSTR *UserName*)

Parámetros

<i>skt</i>	Descriptor del socket conectado al servicio POP3
<i>UserName</i>	Un puntero a un string que contiene el nombre de la cuenta de usuario en el host que actúa como mail server.

Comentarios: Este comando inicia el estado de autorización, el cual finalizará después que el comando DoPASS() haya sido satisfactorio.

Valor Retornado Un valor lógico indicando si el usuario es reconocido o no.

DoPASS()

Descripción: Envía la password del usuario que desea acceder al mailbox .

```
#include <winsock.h>
```

```
#include <pop3.h>
```

BOOLEAN DoPASS(SOCKET *skt*, LPSTR *Password*)**Parámetros**

<i>skt</i>	Descriptor del socket conectado al servicio POP3
<i>Password</i>	Un puntero a un string que contiene la password del usuario escrita en texto plano.

Comentario Este comando finaliza la etapa de autorización para pasar al estado de transacción. Si la password es correcta , el usuario puede operar con el mailbox.

Valor Retornado Un valor lógico indicando si la password se corresponde con el nombre de cuenta previamente indicado.

DoSTAT()

Descripción : Retorna el estado del mailbox.

```
#include <winsock.h>
#include <pop3.h>
```

LPSTR DoSTAT(SOCKET *skt*)**Parámetros**

skt Descriptor del socket conectado al servicio POP3

Comentario Este comando retorna información de la cantidad de mensajes en el mailbox y el tamaño de los mismos en bytes.

Valor Retornado Un puntero string conteniendo la cantidad de mensajes y la total de bytes ocupados por los mismos, separados por un espacio en blanco. Si el comando falla, retornará un valor nulo.

DoLIST()

Descripción: Permite recuperar información referente a cada mensaje en el mailbox.

```
#include <winsock.h>
#include <pop3.h>
```

LPSTR DoLIST(SOCKET *skt*, *CHAR [*MsgNum*])

Parámetros

skt Descriptor del socket conectado al servicio POP3

MsgNum Un puntero a un string que contiene un número de mensaje

Comentarios Si se lo invoca con un parámetro retornará la información referente al mensaje indicado. Si el parámetro no está presente retornará una lista, con información de todos los mensajes. Por convención la información del mensaje consta del número de mensaje, su tamaño en bytes y opcionalmente un texto.

Valor Retornado Un puntero a un string conteniendo información de los mensajes. En caso de error retornará un valor nulo.

DoRETR()

Descripción: Recupera un mensaje del mailbox del usuario.

```
#include <winsock.h>
```

```
#include <pop3.h>
```

LPSTR DoRETR(SOCKET *skt*, CHAR *MsgNum*)

Parámetros

skt Descriptor del socket conectado al servicio POP3

MsgNum Un puntero a un string que contiene un número de mensaje a recuperar

Comentario El comando emite un requerimiento al server, y este devuelve el mensaje donde cada línea esta determinada por CR-LF.

Valor retornado Un puntero string conteniendo el mensaje. En caso de falla, retornará un valor nulo.

DoLAST()

Descripción Recupera el mayor número de mensaje accedido.

```
#include <winsock.h>
```

```
#include <pop3.h>
```

LPSTR DoLAST(SOCKET *skt*)**Parámetros**

skt

Descriptor del socket conectado al servicio POP3

Comentario Este comando permite recuperar el mayor número de mensaje accedido por otro comando.

Valor retornado Un puntero string conteniendo un número de mensaje.

DoDELETE()

Descripción Marca para borrado un mensaje en el mailbox del usuario.

```
#include <winsock.h>
#include <pop3.h>
```

BOOLEAN DoDELETE(SOCKET *skt*, *CHAR *MsgNum*)**Parámetros**

skt Descriptor del socket conectado al servicio POP3

MsgNum Un puntero a un string que contiene un número de mensaje a borrar

Comentario Este comando marca el mensaje para borrado. La eliminación se realizará cuando la sesión sea finalizada.

Valor Retornado Un valor lógico indicando si la operación fue efectuada con éxito.

DoNOOP()

Descripción Envía un requerimiento al server POP para saber si aún está conectado.

```
#include <winsock.h>
```

```
#include <pop3.h>
```

BOOLEAN DoNOOP(SOCKET *skt*)**Parámetros**

skt

Descriptor del socket conectado al servicio POP3

Valor Retornado Un valor lógico que indica si el server esta disponible.

DoRESET()

Descripción Permite desmarcar los mensajes previamente marcados para borrado.

```
#include <winsock.h>
#include <pop3.h>
```

BOOLEAN DoRESET(SOCKET *skt*)**Parámetros**

skt Descriptor del socket conectado al servicio POP3

Comentarios Este comando desmarca todos los mensajes señalados para borrado en la sesión actual.
Cuando la sesión es finalizada con DoQUIT() , los mensajes son borrados.

Valor retornado Un valor lógico indicando si la operación fue o no satisfactoria.

DoTOP()

Descripción Permite recuperar un número de líneas determinado de un mensaje.

```
#include <winsock.h>
#include <pop3.h>
```

LPSTR DoTOP(SOCKET *skt* , *CHAR *MsgNum* , *CHAR *Lines*)

Parámetros

<i>skt</i>	Descriptor del socket conectado al servicio POP3.
<i>MsgNum</i>	Un puntero a un string que contiene el número de mensaje.
<i>Lines</i>	Un puntero a un string que contiene número de líneas a recuperar.

Comentario Recupera un número de líneas a partir de la finalización de la sección de headers. Si el número de líneas indicadas es mayor que el número de líneas del mensaje, retornará el mensaje completo.

Valor retornado Un puntero a un string conteniendo las líneas del mensaje, en caso de falla retornará un valor nulo.

DoQUIT()**Descripción** Termina la sesión entre el server y el cliente

#include <winsock.h>

#include <pop3.h>

BOOLEAN DoQUIT(SOCKET *skt*)**Parámetros***skt* Descriptor del socket conectado al servicio POP3**Comentario** Permite cerrar el canal de comunicación, de forma “amigable”.**Valor retornado** Un valor lógico indicando si la operación fue o no satisfactoria.

DoGREET()

Descripción Lee el saludo inicial desde la conexión.

```
#include <winsock.h>
#include <pop3.h>
```

VOID DoGREET(SOCKET *skt*)

Parámetros

skt Descriptor del socket conectado al servicio POP3

Comentario Este comando no forma parte de la especificación del protocolo, pero se implementa con el fin de iniciar la sesión entre el cliente y el server POP.

Valor retornado Ninguno.

SMTP/ESMTP

DoHELO()

Descripción Este comando es usado por el emisor para identificarse ante el receptor.

BOOLEAN DoHELO(SOCKET *skt*, LPSTR *DomainName*)

```
#include <winsock.h>
#include <smtp.h>
```

Parámetros

skt Descriptor del socket conectado al servicio SMTP

DomainName Nombre del host emisor.

Comentario Este comando permite al sender SMTP presentarse al receiver. El receiver acepta con una respuesta positiva. Internamente el server y receiver, después de la ejecución de este comando, están en estado inicial, sin transacciones pendientes y además, los buffers y tablas de estados son reseteadas.

Valor retornado Un valor lógico que indica que el server y el cliente están en estado inicial.

DoEHLO()

Descripción Es utilizado para consultar al server SMTP que extensiones soporta.

```
#include <winsock.h>
```

```
#include <smtp.h>
```

LPSTR DoEHLO(SOCKET *skt* , LPSTR *DomainName*)

Parámetros

skt Descriptor del socket conectado al servicio SMTP.

DomainName Un puntero a string conteniendo el nombre del dominio.

Valor Retornado Un puntero a un string de caracteres conteniendo una lista de valores que indican las extensiones que el server soporta.

**DoMAIL()**

Descripción Identifica al emisor de un mensaje.

BOOLEAN DoMAIL(SOCKET *skt* , LPSTR *Sender*, [LPSTR *Param_ext*])

```
#include <winsock.h>
```

```
#include <smtp.h>
```

Parámetros

<i>skt</i>	Descriptor del socket conectado al servicio SMTP.
<i>Sender</i>	Un puntero a un string que contiene la dirección de mail del emisor.
<i>Param_ext</i>	Parámetros adicionales , para soportar extensiones de SMTP.

Comentario Este comando establece la transacción inicial para el envío de un mensaje de mail, identificando al sender (emisor) e informando al receiver (receptor) que comienza la transacción de un nuevo mail. En caso de utilizar ESMTP, las extensiones adicionan parámetros de la forma “parametro= valor”, los cuales están definidos por la extensión. La extensiones soportadas son consultadas por el comando **DoEHLO()**.

Valor Retornado Valor lógico que indica si el sender está establecido.

DoRCPT()

Descripción Identifica un mailbox de usuario al que será enviado un mensaje de mail.

```
#include <winsock.h>
#include <smtp.h>
```

BOOLEN DoRCPT(SOCKET *skt* ,LPSTR *Receiver* , [LPSTR *Param_ext*])

Parámetros

<i>skt</i>	Descriptor del socket conectado al servicio SMTP
<i>Receiver</i>	Puntero a un string que contiene la dirección de mail del receptor del mensaje
<i>Param_ext</i>	Parámetros adicionales , para soportar extensiones de SMTP.

Comentario Múltiples mailbox pueden ser especificados haciendo uso múltiple de este comando. Este comando debe ejecutarse después de un **DoMAIL()**. En caso de utilizar extensiones, normalmente, se requieren parámetros adicionales . Los parámetros serán de la forma “parametro=valor”. El detalle de los parámetros está indicado en la especificación de la extensión y el comando **DoEHLO()** debe ser utilizado para iniciar la sesión. La respuesta de dicho comando, además, es una lista de todas las extensiones soportadas por el server.

Valor retornado Valor lógico indicando si el receptor puede ser identificado.

DoNOOP()

Descripción Envía un requerimiento al server y espera una respuesta.

```
#include <winsock.h>
```

```
#include <smtp.h>
```

BOOLEAN DoNOOP(SOCKET *skt*)**Parámetros**

skt Descriptor del socket conectado al servicio SMTP

Comentario Este comando no afecta a ningún comando ni parámetro previamente enviado. No especifica ninguna otra acción más que recibir una respuesta del server.

Valor Retornado Valor lógico indicando la operatividad del server o no.

DoRESET()

Descripción Aborta todas las transacciones, inicializando las tablas de estados y los buffers.

```
#include <winsock.h>
#include <smtp.h>
```

BOOLEAN DoRESET(SOCKET *skt*)**Parámetros**

skt Descriptor del socket conectado al servicio SMTP

Comentario Cualquier sender o receivers almacenados por DoMAIL() o DoRCPT() y datos de mails serán descartados. Se retorna al estado inicial.

Valor retornado Valor lógico indicando si la operación fue o no satisfactoria

DoVRFY()

Descripción Verifica la existencia de una dirección de mail.

LPSTR DoVRFY(SOCKET *skt* , LPSTR *MailAddr*)

```
#include <winsock.h>
```

```
#include <smtp.h>
```

Parámetros

skt Descriptor del socket conectado al servicio SMTP

MailAddr Puntero a un string conteniendo una dirección de mail

Valor Retornado Si es satisfactorio retorna el full name del usuario. Caso contrario retornará un valor nulo.

DoDATA()

Descripción Envía los datos del cuerpo del mensaje.

```
#include <winsock.h>
```

```
#include <smtp.h>
```

BOOLEAN DoDATA(SOCKET *skt*, LPSTR *MessageText*)

Parámetros

skt Descriptor del socket conectado al servicio SMTP

MessageText Puntero a un string conteniendo el texto del mensaje.

Valor Retornado Valor lógico que indica si el mensaje a sido aceptado por server SMTP.

DoQUIT()

Descripción Requiere el cierre del canal de transmisión.

```
#include <winsock.h>
```

```
#include <smtp.h>
```

BOOLEAN DoQUIT(SOCKET *skt*)**Parámetros**

skt Descriptor del socket conectado al servicio SMTP

Comentario El server no cerrará el canal de transmisión hasta no recibir y responder a este comando.

Valor Retornado Valor lógico indicando que el canal se cerró o no satisfactoriamente.

DoEXPN()**Descripción** Expande una lista de mail (mailing list)

#include <winsock.h>

#include <smtp.h>

LPSTR DoEXPN(SOCKET *skt* , LPSTR *ListName*)**Parámetros***skt* Descriptor del socket conectado al servicio SMTP.*ListName* Un puntero a un string conteniendo el nombre de una lista de Mail.**Valor Retornado** Un puntero a un string conteniendo los nombres completos de los miembros de la lista y el mailbox .

NNTP

DoARTICLE()

Descripción Recupera un artículo indicado por el parámetro o el actual en caso contrario.

```
#include <winsock.h>
#include <nntp.h>
```

LPSTR DoARTICLE(SOCKET *skt* , [LPSTR *ArticleID*])

Parámetros

<i>skt</i>	Descriptor del socket conectado al servicio NNTP.
<i>ArticleID</i>	Un puntero a un número o identificador de artículo, en este último caso el valor debe ir entre "<>" .

Comentario Este comando tiene dos formas de uso, en la primera el parámetro es un identificador de artículo. En la segunda es un número de artículo dentro del rango permitido por el Newsgroup o bien sin parámetro, lo cual señala que el artículo referenciado es el actual. Si el parámetro es un identificador del artículo, el puntero al artículo actual no es alterado. Si es un número de mensaje, el puntero al artículo actual es actualizado. En todos los casos el comando retornará el header, una línea en blanco y el cuerpo del artículo especificado.

Valor Retornado Un puntero a un string conteniendo el artículo seleccionado. Si no existe, devuelve un valor nulo.

DoGROUP()

Descripcion: Permite seleccionar el nombre de un newsgroup.

```
#include <winsock.h>
#include <nntp.h>
```

LPSTR DoGROUP(SOCKET *skt*,LPSTR *GroupName*)**Parámetros:**

<i>skt</i>	Descriptor del socket conectado al servicio NNTP.
<i>GroupName</i>	Un puntero a un nombre del newsgroup a ser seleccionado.

Comentarios Una lista de newsgroup válidos puede ser obtenida con el comando **DoLIST()**

Valor Retornado Un puntero a un string conteniendo el número estimado de artículos en el grupo, el número del primer y último artículo y el nombre del grupo. En caso de que el grupo no sea válido retornará el valor nulo.

DoHELP()

Descripción: Provee una breve descripción de los comandos implementados en el server.

```
#include <winsock.h>
```

```
#include <nntp.h>
```

LPSTR DoHELP(SOCKET *skt*)**Parámetros**

skt Descriptor del socket conectado al servicio NNTP.

Valor Retornado Un puntero a un string conteniendo la lista de comandos soportados por el server NNTP.

DOIHave()

Descripción Permite informar al server que el “host cliente” tiene un artículo. Si el server desea una copia retornará una instrucción para que el cliente se lo envíe.

BOOLEAN DOIHAVE(SOCKET *skt* , LPSTR *ArticleID*)

```
#include <winsock.h>
```

```
#include <nntp.h>
```

Parámetros

skt Descriptor del socket conectado al servicio NNTP.

ArticleID Un puntero a un identificador de artículo.

Comentario Si el server acepta el requerimiento del artículo, el host cliente debe enviarlo (headers y body). Este comando normalmente se utiliza para intercambio de artículos entre servers y no cuando el cliente es un programa lector de news.

Valor retornado Valor lógico indicando si el artículo fue transferido o no con éxito.

DoLAST() / DoNEXT()

Descripción Mueve el puntero que referencia al artículo actual, al artículo previo o posterior dentro del newsgroup.

LPSTR DoLAST(SOCKET *skt*)

LPSTR DoNEXT(SOCKET *skt*)

```
#include <winsock.h>
```

```
#include <nntp.h>
```

Parámetros

skt Descriptor del socket conectado al servicio NNTP.

Comentarios En el caso de **DoLAST()**, si el artículo actual es el primero ocurre un error, de la misma manera en **DoNEXT()**, si es el último.

Valor Retornado Un puntero a un string conteniendo el número de artículo y el identificador de artículo separado por blanco. En caso de error retornará el valor nulo.

DoLIST()

Descripción Permite obtener una lista de newsgroups válidos e información asociada a los mismos.

LPSTR DoLIST(SOCKET *skt*)

```
#include <winsock.h>
```

```
#include <nntp.h>
```

Parámetros

skt Descriptor del socket conectado al servicio NNTP.

Comentarios Este comando permite extraer información para ser utilizada por el resto de los comandos. Por ejemplo: permite averiguar si un newsgroup tiene habilitado el posting de artículos.

Valor Retornado Un puntero a un string multilínea, donde cada línea es de la forma: nombre newsgroup, número del último y primer artículo y un valor indicando si el posting esta permitido o no.

DoNEWGROUPS()

Descripción Permite obtener una lista de newsgroup creados a partir de una fecha y hora dada en el mismo formato que el comando **DoLIST()**.

```
#include <winsock.h>
#include <nntp.h>
```

LPSTR DoNEWGROUPS(SOCKET *skt*, *CHAR *date*, *CHAR *time*, [const *CHAR *gmt*], [LPSTR<DistList>])

Parámetros

<i>skt</i>	Descriptor del socket conectado al servicio NNTP.
<i>date</i>	Puntero a un string conteniendo la fecha en formato YYMMDD
<i>time</i>	Puntero a un string conteniendo la hora en formato HHMMSS
<i>gmt</i>	Describe la zona horaria.
<i>DistListt</i>	Es una lista de grupos de distribución delimitada con "<>" donde los elementos están separados por comas.

Comentarios El Parámetro *gmt* puede contener un valor nulo o "GMT". En el primer caso la zona horaria es igual a la del server. En el segundo la hora y fecha son evaluadas en el meridiano cero.

Valor Retornado Un puntero a un string multilínea, donde cada línea es de la forma: nombre newsgroup, número del último y primer artículo y un valor indicando si el posting está permitido o no.

DoNEWNEWS()

Descripción Retorna una lista de identificadores de artículos puestos o recibidos en el newsgroup especificado desde la fecha indicada.

```
#include <winsock.h>
#include <nntp.h>
```

LPSTR DoNEWGROUPS(SOCKET *skt*, LPSTR *GroupName*, *CHAR *date*, *CHAR *time*, [const *CHAR *gmt*], [LPSTR<*DistList*>])

Parámetros

<i>skt</i>	Descriptor del socket conectado al servicio NNTP.
<i>GroupName</i>	Puntero a un string conteniendo el nombre del Newsgroup.
<i>date</i>	Puntero a un string conteniendo la fecha en formato YYMMDD
<i>time</i>	Puntero a un string conteniendo la hora en formato HHMMSS
<i>gmt</i>	Describe la zona horaria.
<i>DistList</i>	Es una lista de grupos de distribución delimitada con “<>” donde los elementos están separados por comas.

Comentario El Parámetro *GroupName* permite la utilización de los metacaracteres “*” y “!” para expansión y negación de nombre de newsgroup.

Valor Retornado Un puntero a un string multilínea conteniendo los identificadores de artículos del grupo agregado a partir de la fecha y hora indicada.

DoSLAVE()

Descripción Informa al server que el cliente es un server esclavo.

BOOLEAN DoSLAVE(SOCKET *skt*)**Parámetros**

skt Descriptor del socket conectado al servicio NNTP.

Comentario Este comando se utiliza para que un server subsidiario requiera mayor prioridad, dado que atiende a varios clientes.

Valor retornado Valor lógico que indica si el server reconoce el estado de server esclavo.

DoPOST()

Descripción Permite enviar un artículo a un server.

```
#include <winsock.h>
#include <nntp.h>
```

BOOLEAN DoPOST(SOCKET *skt*, LPSTR *MessageText*)**Parámetros**

skt Descriptor del socket conectado al servicio NNTP.

MessageText Puntero a un string conteniendo el texto del mensaje a enviar.

Comentario Si el server permite el “posting”, el cliente enviará el artículo con header y body al newsgroup seleccionado.

Valor retornado Valor lógico indicando si el artículo fue aceptado o no por el server.

DoQUIT()

Descripción Requiere el cierre del canal de transmisión.

```
#include <winsock.h>
```

```
#include <nntp.h>
```

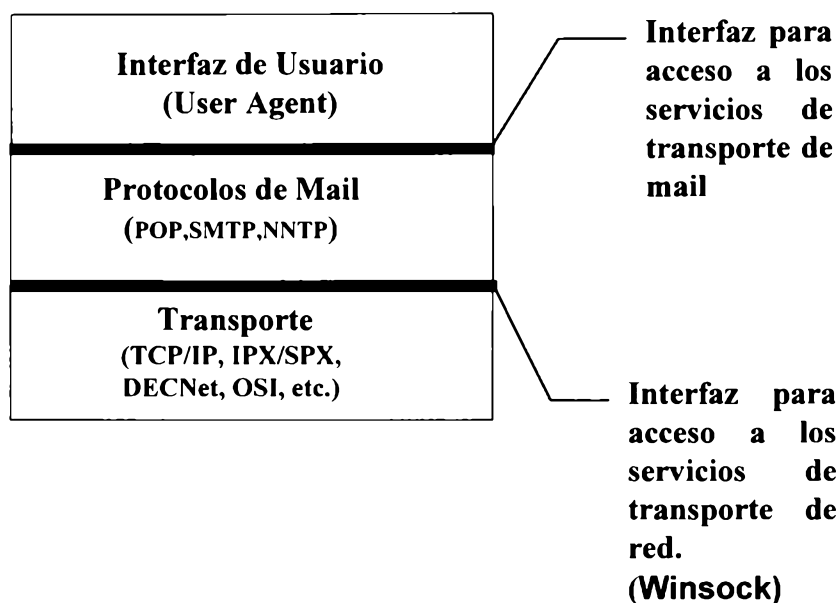
BOOLEAN DoQUIT(SOCKET *skt*)**Parámetros**

skt Descriptor del socket conectado al servicio NNTP.

Valor retornado Valor lógico indicado el resultado de la operación

Interfaz con los protocolos de transporte.

La implementación de interfaces genéricas es un objetivo perseguido por la mayoría de los modelos informáticos. El desarrollo de software cliente para manejo de mensajería en la comunidad Internet, debe considerar el siguiente modelo :



Primero analicemos la interfaz con el transporte de red, provista por los Windows Socket que dispone una forma transparente de acceso a los servicios de red, los cuales hasta la versión 1.1 soportaba TCP/IP solamente y en la nueva versión 2.0 [WINSOCK2] agregan diferentes protocolos como IPX/SPX, DECNet y OSI. Esta nueva definición permite desarrollar aplicaciones independiente del protocolo de transporte.

El manejo de esta interfaz requiere un conocimiento del protocolo de transporte y de la filosofía de los BSD sockets. Para el alcance de nuestro proyecto se consideró el TCP/IP y en particular, los socket basados en Streams [COMER 93].

Como se mencionó anteriormente, se puede establecer una división funcional del U.A., la relacionada con la interfaz de usuario y la relacionada con el transporte de mail. Esta última debería proveer un conjunto de funciones genéricas, de manera tal, que al modificar o agregar protocolos de mail, no deba modificarse la interfaz de usuario propiamente dicha.

En el modelo Split-U.A., mencionado en este informe, las necesidades genéricas para transporte son:

1. Requerir el servicio

2. Enviar o Recuperar mensajes
3. Liberar el servicio.

Recuperación de mensajes.

Un usuario se conecta a su Mail Server para leer sus mensajes, los que serán transportado a su inbox local y posteriormente manejados localmente por medio de las herramientas brindada por la interfaz de usuario.

La conexión se solicita explícitamente al MTA, entonces los mensajes son recuperados y finalmente borrados del sitio remoto. Este esquema es el seguido por la mayoría de los U.A. utilizados en la comunidad Internet.

Funcionalmente la recuperación incluye :

1. Identificación
2. Testeo de existencia de mensajes
3. Recuperación de los mensajes
4. Borrado de mensajes en el MTA

El paso 4 debe ser complementado con el almacenamiento de los mensajes en el file system local dentro de las estructuras definidas por el U.A.

Envío de mail

El mensaje de mail será compuesto locamente, esto es, se generará un mensaje que siga el formato de la RFC-822 y luego será enviado al destino.

Genéricamente para enviar un mensaje se deben establecer un emisor, un receptor y un mensaje a enviar.

Funcionalmente el envío incluye :

1. Establecer emisor
2. Establecer uno o más receptores
3. Enviar los datos del mensaje

Funciones propuestas

RequestService(*ServiceName,UserName,Password*)

Descripción: Esta primitiva permite solicitar un servicio e identificarse si es necesario.

Parámetros:

<i>ServiceName</i>	Nombre del servicio a utilizar.
<i>UserName</i>	Identificación del usuario que hará uso del servicio.
<i>Password</i>	Password asociada al usuario.

Comentario: Tratándose de un servicio de mensajería, el requerimiento de este, está asociado con un protocolo de transporte de mail (POP3, SMTP, NNTP) y que esta previamente definido por la configuración.

Valor retornado: Un entero positivo que actuará como identificador de servicio y solo es válido para la sesión. Si el servicio no está disponible el valor retornado es menor o igual a cero.

ReleaseService(*ServiceID*)

Descripción: Con esta primitiva se libera un servicio activo.

Parámetros

ServiceID Identificador de servicio retornado por **RequestServices()**.

Valor Retornado: Un valor lógico indicando si el servicio pudo ser liberado.

TotalMessage(*ServiceId*)

Descripción: Recupera la cantidad de mensajes en mailbox del usuario.

Parámetros

ServiceID Identificador de servicio retornado por **RequestServices()**.

Comentario: Generalmente los mensajes son ordinalmente numerados, por lo tanto el valor retornado por esta función permite establecer el rango máximo de valores permitidos para identificador de mensajes.

Valor Retornado: Un valor entero indicando la cantidad de mensajes.

RetrieveMessage(*MessageID*)

Descripción: Recupera el mensaje identificado por el parámetro.

Parámetro:

MessageID Identificador del mensaje a recuperar.

Valor Retornado: Un string conteniendo el mensaje requerido como una sola unidad (header y body).



MessageSender(*MailAddress*)

Descripción: Establece el emisor del mensaje.

Parámetros

MailAddress Una dirección de mail en el formato establecido por la RFC-822.

Comentario: La dirección del emisor debe ser indicada para conocer a donde se envían los mensajes de error o notificación.

Valor Retornado: Un valor lógico indicando si el comando fue satisfactorio o no.

MessageReceiver(ListMailAddr)

Descripción: Establece una o mas direcciones receptoras de un mensaje.

Parámetros

ListMailAddr Una lista de direcciones de mail en formato especificado en la RFC-822.

Valor Retornado : Uno de los siguientes valores

- **RCV_OK:** Indica que todos los receptores fueron establecido con éxito.
- **RCV_PARTIAL:** Alguno de los receptores pudieron ser establecidos y otros no.
- **RCV_ERROR:** Ninguno de los receptores pudieron ser establecidos.

SendMessage(*MessageText*)

Descripción: Envía el mensaje a las direcciones previamente establecidas.

Parámetros

MessageText Mensaje a enviar compuesto según el formato establecido por la RFC-822.

Valor Retornado: Valor lógico indicando si el comando fue o no satisfactorio.

Conclusión

Ahora que tenemos definidos todos los aspectos conceptuales que componen un sistema de mensajería, podemos sacar las siguientes conclusiones:

Todo los productos que hemos analizado, tales como Eudora, PCMail, Dmail, etc., tienen aspectos en común, los cuales nos permite dividirlos en dos partes bien diferenciadas una de la otra.

- Por un lado está la interfaz de usuario que los distingue unos de otros, dándole mas o menos funcionalidad al U.A., o mayor o menor facilidad de uso.
- La otra parte, es la que se encarga de entregar y recibir los mensajes a los MTA, utilizando los protocolos adecuados.

Obviamente esta división es funcional, ya que todos los productos que vimos están dentro de un único módulo que realiza las dos tareas.

Pues bien, lo que nosotros intentamos hacer es que esa división exista explícitamente. ¿ Por qué ? La respuesta hay que buscarla por el lado de las motivaciones que encontramos, que a continuación detallamos:

Por un lado, tenemos la experiencia de un U.A. iconizado desarrollado en el laboratorio, con prestaciones adaptadas a nuestras necesidades, utilizando como protocolo de transporte la versión UUCP de dominio público para sistemas operativos DOS y Windows (UUPC) [HARARI 93]. Un problema que tiene es la sincronización que hay que hacer entre las dos partes, ya que funcionan en forma separada.

Otra motivación es la que se comentó al comienzo, que son las nuevas definiciones de extensiones que agregan funcionalidad al sistema de mail, lo cual obliga a redefinir la interfaz para poder soportarlas.

Otra, fue la necesidad de aprovechar la potencialidad de los Windows Sockets (como interfaz de networking para el desarrollo de los clientes) sobre todo la nueva versión 2 la cual expande su uso no solo a redes de TCP/IP.

Y por último, la cual nosotros consideramos mas importante es la posibilidad de ir integrando nuevos trabajos, e ir aprovechando tanto el desarrollo como la experiencia, para poder brindar nuevos servicios, ya que tanto los sockets, por un lado y las extensiones de protocolos por el otro, lo permiten hacer naturalmente.

Saber hasta donde se puede implementar cada uno de los protocolos, no sería un problema, ya que naturalmente cada uno tiene su funcionalidad bien definida. El problema está que hoy día ninguno de los protocolos brinda todos los servicios

requeridos, o no lo hace de la manera mas eficiente. Siempre haciendo referencia al entorno de un usuario que cuenta con un equipo de prestaciones limitadas.

En la actualidad, lo más natural en este escenario es tener un protocolo que se encargue de recuperar los mensajes del MTA del sistema, como lo es POP3, aunque bien lo podría hacer SMTP, pero obliga a que el nodo se convierta en un servidor SMTP, ya que solo cuenta con capacidad de enviar mensajes. Lo cual es viable en tanto y en cuanto el equipo en cuestión tenga el porte suficiente para poder soportarlo. Cosa que con POP3 no es necesario.

La otra cuestión que estuvo presente en el proyecto fue la posibilidad que los U.A. en desarrollo puedan utilizar UUCP, para transportar los mensajes, ampliando aún más su uso.

Si bien en una primera instancia se trató de definir una interfaz general que abarcara todos los protocolos por nosotros implementados (POP3, SMTP, NNTP) y UUCP, llegamos a la conclusión de que la integración solo la podemos obtener a nivel del U.A., utilizando primitivas que permitan manejar los mensajes con UUCP de la misma manera que se hizo con los demás protocolos a través de la configuración del U.A., donde se establecerá el ambiente local necesario para uno u otro transporte elegido.

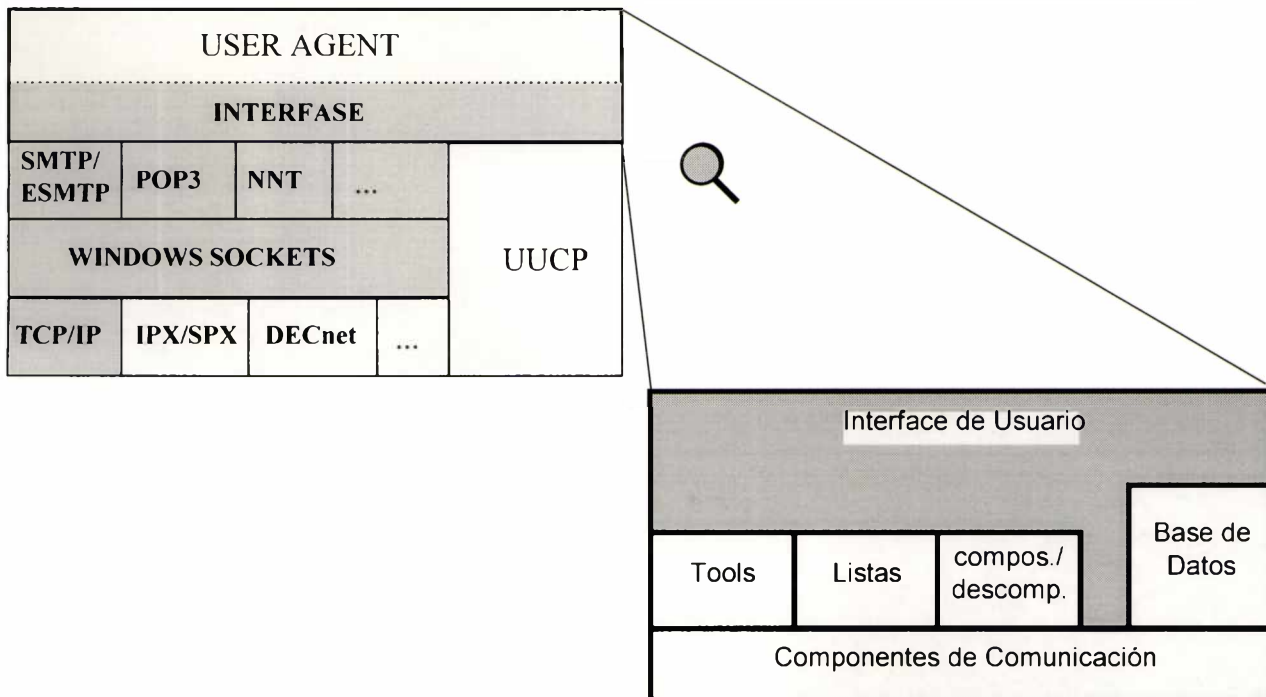
Otra conclusión que aún más fundamenta lo anterior, es el hecho de que tampoco es posible crear una interfaz general por encima de los protocolos TCP/IP (por llamarlo de alguna manera) ya que son complementarios y no alternativos (POP3, SMTP por ejemplo), sobre todo si tenemos en cuenta que nuestro universo de trabajo son la implementación de protocolos de entrega y recepción de mensajes en equipos con prestaciones limitadas (PC) como clientes.

Para lograr dicha “interfaz” implementamos toda la funcionalidad que cada uno de los protocolos proveen, apegándonos fielmente a la definición de cada uno de ellos. Si bien esto nos permite entregarle a los desarrolladores de U.A. una abstracción bastante homogénea de los protocolos subyacentes, no evita que los implementadores tengan que conocer aspectos funcionales de los protocolos. Cosas tales como códigos de respuestas, códigos de error y valores retornados, son bastantes complejos de interpretar.

Por tal motivo, nosotros consideramos apropiado “ parsear “ las respuestas de tal manera que se manejen tipos bien definidos de parámetros y argumentos que evitan a las capas superiores tener que procesar la información antes de consumirla o entregarlas a los distintos protocolos.

Si bien esta propuesta viola de alguna manera el modelo conceptual para el manejo de mail que dimos anteriormente, ya que el parsing seria parte del U.A. y no de los delivery/posting protocols. Es la única forma de lograr separar lo que habitualmente está en un único módulo.

La representación gráfica de lo anteriormente expresado sería :



Esta representación muestra de alguna manera que si bien el correo electrónico es un servicio conceptualmente simple, no lo es tanto si se lo analiza desde cada una de las partes que lo componen.

Posibilidad de continuación del trabajo.

Además de la integración con una interfaz icónica de alto nivel que está en progreso por parte de otros tesisas, existen varias extensiones de interés como:

- a) Proveer mecanismos para notificación, extendiendo y complementando los estándares propuestos en un entorno de uso bien definido.
- b) Implementación de Secure Socket Layer (SSL), un nuevo servicio de seguridad que, a pesar de sus limitaciones, se está adoptando como un estándar para transacciones privadas (comerciales).
- c) Extensión de los servicios de mensajería sobre otros protocolos. Por ej. IPX.

ANEXO A

Gramática Backus Naur Form (BNF)

Usualmente, una sintaxis simple es utilizada para describir interacciones, escritas usando un lenguaje en el cual las reglas de producción son definidas de tal manera que se resuelven en un conjunto de símbolos terminales. Una regla de producción es definida como:

lhs ::= rhs

donde rhs consiste de uno o mas elementos.

Los elementos posibles son:

alternativos: indica una o mas opciones

opción1 / opción 2

agrupamiento: se utiliza para formar una expresión de dos o mas elementos

(expresión1 expresión2 ...)

repetición: se utiliza para especificar la repetición de un elemento dentro de algún rango.

<inferior>*<superior>elemento

Si <inferior> no existe , el limite inferior del rango es 0. Similarmente si <superior> no existe, entonces el limite superior del rango es infinito. Una notación reducida es

<n>elemento y es equivalente a <n>*<n>elemento

opcional: indica que un elemento puede, opcionalmente, estar presente.

[elemento] lo cual es equivalente a *1 elemento

lista-de-repetición:

<inferior>#<superior> y es equivalente a

elemento <inferior-1>*<superior>("," elemento)

Finalmente, un ";" indica un comentario.

Definición Formal de un mensaje (Según RFC 822).

La representación en BNF (manteniendo la sintaxis original) de las entidades involucradas en la definición de un mensaje sería:

		;(Octal, Decimal.)
CHAR	= <any ASCII character>	;(0-177, 0.-127.)
ALPHA	= <any ASCII alphabetic character>	
		;(101-132, 65.- 90.)
		;(141-172, 97.-122.)
DIGIT	= <any ASCII decimal digit>	;(60- 71, 48.- 57.)
CTL	= <any ASCII control character and DEL>	;(0- 37, 0.- 31.)
		;(177, 127.)
CR	= <ASCII CR, carriage return>	;(15, 13.)
LF	= <ASCII LF, linefeed>	;(12, 10.)
SPACE	= <ASCII SP, space>	;(40, 32.)
HTAB	= <ASCII HT, horizontal-tab>	;(11, 9.)
<'>	= <ASCII quote mark>	;(42, 34.)
CRLF	= CR LF	
LWSP-char	= SPACE / HTAB	; semantics = SPACE
línear-white-space	= 1*([CRLF] LWSP-char)	; semantics = SPACE ; CRLF => folding
specials	= “(“ / “)” / “<” / “>” / “@” / “,” / “;” / “:” / “\” / “<”> / “.” / “[” / “]”	; Must be in quoted- string, to use ; within a word.
delimiters	= specials / línear-white-space / comment	
text	= <any CHAR, including bare CR & bare LF, but NOT including CRLF>	; => atoms, specials, ; comments and ; quoted-strings are ; NOT recognized.
atom	= 1*<any CHAR except specials, SPACE and CTLs>	
quoted-string	= <'> *(qtext/quoted-pair) <'>	; Regular qtext or ; quoted chars.
qtext	= <any CHAR excepting <'>, “” & CR, and including línear-white-space>	; => may be folded
domain-literal	= “[” *(dtext / quoted-pair) “]”	
dtext	= <any CHAR excluding “[”, ; => may be folded	

	“]”, “\” & CR, & including línear-white-space>
comment	= “(“ *(ctext / quoted-pair / comment) “)”
ctext	= <any CHAR excluding “(“ ; => may be folded “)”, “\” & CR, & including línear-white-space>
quoted-pair	= “\” CHAR ; may quote any char
phrase	= 1*word ; Sequence of words
word	= atom / quoted-string

MIME (RFC 1521)

Esta sección contiene la gramática BNF para la definición de MIME, manteniendo la sintaxis original. La gramática es incompleta ya que algunas entidades están definidas en la sección anterior

application-subtype := ("octet-stream" *stream-param) / "postscript" / extension-token
 application-type := "application" "/" application-subtype

attribute := token ; case-insensitive

atype := "ftp" / "anon-ftp" / "tftp" / "local-file" / "afs" / "mail-server"
 /extension-token
 ; Case-insensitive

audio-type := "audio" "/" ("basic" / extension-token)

body-part := <"message" como está definido en la RFC 822>

boundary := 0*69<bchars> bcharsnospace

bchars := bcharsnospace / " "

bcharsnospace := DIGIT / ALPHA / "'" / "(" / ")" / "+" / "_"
 / "," / "-" / "." / "/" / ":" / "=" / "?"

charset := "us-ascii" / "iso-8859-1" / "iso-8859-2" / "iso-8859-3"
 / "iso-8859-4" / "iso-8859-5" / "iso-8859-6" / "iso-8859-7"
 / "iso-8859-8" / "iso-8859-9" / extension-token
 ; case insensitive

close-delimiter := "—" boundary "—" CRLF; Again, no space by "—",

content := "Content-Type" ":" type "/" subtype *(";" parameter)
 ; case-insensitive matching of type and subtype

delimiter := "—" boundary CRLF ; taken from Content-Type field.
 ; There must be no space
 ; between "—" and boundary.

description := "Content-Description" ":" *text

discard-text := *(*text CRLF)

encapsulation := delimiter body-part CRLF

encoding	:= "Content-Transfer-Encoding" ":" mechanism
epilogue	:= discard-text ; to be ignored upon receipt.
extension-token	:= x-token / iana-token
external-param	:= (";" "access-type" "=" atype) / (";" "expiration" "=" date-time) ; Note that date-time is quoted / (";" "size" "=" 1*DIGIT) / (";" "permission" "=" ("read" / "read-write")) ; Permission is case-insensitive / (";" "name" "=" value) / (";" "site" "=" value) / (";" "dir" "=" value) / (";" "mode" "=" value) / (";" "server" "=" value) / (";" "subject" "=" value) ;access-type required; others required based on access-type
iana-token	:= <a publicly-defined extension token,registered with IANA>
id	:= "Content-ID" ":" msg-id
image-type	:= "image" "/" ("gif" / "jpeg" / extension-token)
mechanism	:= "7bit" ; case-insensitive / "quoted-printable" / "base64" / "8bit" / "binary" / x-token
message-subtype	:= "rfc822" / "partial" 2#3partial-param / "external-body" 1*external-param / extension-token
message-type	:= "message" "/" message-subtype
multipart-body	:=preamble 1*encapsulation close-delimiter epilogue

multipart-subtype	:= "mixed" / "parallel" / "digest" / "alternative" / extension-token
multipart-type	:= "multipart" "/" multipart-subtype ";" "boundary" "=" boundary
octet	:= "=" 2(DIGIT / "A" / "B" / "C" / "D" / "E" / "F") ; octet must be used for characters > 127, =, SPACE, or TAB, ; and is recommended for any characters not listed in
padding	:= "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7"
parameter	:= attribute "=" value
partial-param	:= (";" "id" "=" value) / (";" "number" "=" 1*DIGIT) / (";" "total" "=" 1*DIGIT) ; id & number required; total required for last part
preamble	:= discard-text ; to be ignored upon receipt.
ptext	:= octet / <any ASCII character except "=", SPACE, or TAB> ; characters not listed as "mail-safe" in Appendix B ; are also not recommended.
quoted-printable	:= ([*(ptext / SPACE / TAB) ptext] ["="] CRLF) ; Maximum line length of 76 characters excluding CRLF
stream-param	:= (";" "type" "=" value) / (";" "padding" "=" padding)
subtype	:= token ; case-insensitive
text-subtype	:= "plain" / extension-token
text-type	:= "text" "/" text-subtype [";" "charset" "=" charset]
token	:= 1*<any (ASCII) CHAR except SPACE, CTLs, or tspecials>
tspecials	:= "(" / ")" / "<" / ">" / "@" / "," / ";" / "." / "\" / <> / "/" / "[" / "]" / "?" / "=" ; Must be in quoted-string, ; to use within parameter values
type	:= "application" / "audio" ; case-insensitive

	/ "image" / "message" / "multipart" / "text" / "video" / extension-token ; All values case-insensitive
value	:= token / quoted-string
version	:= "MIME-Version" ":" 1*DIGIT "." 1*DIGIT
video-type	:= "video" "/" ("mpeg" / extension-token)
x-token	:= <The two characters "X-" or "x-" followed, with no intervening white space, by any token>

ANEXO B

SMTP. Comandos y códigos de respuestas

A continuación se dará una descripción de los comandos SMTP.

HELLO <dominio>

Este comando es utilizado por el emisor SMTP para identificarse ante el receptor. El argumento contiene el nombre de host del emisor.

MAIL FROM: <dirección mail>

Este comando le indica al receptor que una transacción para un nuevo mail va a comenzar y que debe resetear todas las tablas de estados y buffers.

RCPT TO: <dirección de mail>

Este comando se utiliza para identificar a un recipiente individual de datos de mail. Por medio de múltiples usos de este comando pueden ser especificados, múltiples recipientes a los cuales enviar este mensaje.

DATA

Este comando es emitido para indicar que los datos que vienen a continuación corresponden al mensaje. Para indicar la finalización de los datos del mensaje se transmite la secuencia de caracteres "<CRLF>.<CRLF>".

RSET

Este comando permite abortar la transacción actual de mail. Los datos del emisor, receptor y del mensaje serán descartados.

NOOP

Este comando no especifica otra acción mas que el receptor envíe un código de respuesta OK.

EXPN <lista de mail>

Este comando permite expandir una lista de mail.

QUIT

Este comando es utilizado para cerrar el canal de comunicación entre el receptor y el emisor SMTP.

Respuestas a los comandos SMTP

Como mencionamos en la descripción de SMTP todas las respuestas a los comandos están formadas por un código de respuestas de tres dígitos, que puede estar seguido por un texto descriptivo.

El primer dígito indica si el comando ha sido completado y si fue satisfactorio o no. Los valores que puede tomar son:

Positive preliminary (1YZ): el server está lista para ejecutar el comando, pero espera la confirmación del cliente. No hay comandos en SMTP que retornen este código, es agregado por definición.

Positive completion (2YZ): el server a ejecutado el comando satisfactoriamente.

Positive intermediate (3YZ): el server está listo para realizar el comando, pero requiere información adicional por parte del cliente.

Transient negative (4YZ): el server no pudo ejecutar el comando debido a un problema transitorio. El cliente puede retransmitir el comando posteriormente.

Permanent negative(5YZ): el server no puede realizar el comando debido a un problema permanente.

El segundo dígito indica porqué el comando fue ó no satisfactorio.

Syntax (X0Z): tiene relación con errores sintácticos.

Informational (X1Z): la respuesta contiene información requerida por el cliente.

Connection (X2Z): la respuesta esta relacionada con el servicio de transporte.

Aplication-Specific (X5Z): la respuesta está relacionada con el sistema de transferencias de mail.

El tercer dígito es utilizado para distinguir códigos de respuestas cuyos dos primeros dígitos son iguales.

Las respuestas a los comandos pueden ser de una o mas líneas. Si la respuesta es de una sola línea estará formada por un código seguido de un espacio y opcionalmente algún texto. Si la respuesta es multilínea, cada línea de la respuesta comienza con el mismo código, seguido por el caracter "-", excepto la última línea en la cual el código es seguido por un espacio.

POP. Comandos y códigos de respuestas.

A continuación se detalla los comando POP.

USER *nombre*

Es utilizado para identificar al mailbox.

PASS *password*

Sirve para transmitir la password asociada con el mailbox.

STAT

Es utilizado para determinar el numero de mensajes en el mailbox y el tamaño total del mailbox.

LIST [*mensaje*]

Si el comando es transmitido con el argumento opcional la respuesta contiene información del tamaño del mensaje solicitado. Si no es especificado devuelve una lista conteniendo información de todos los mensajes.

LAST

Es utilizado para tomar el número mas alto de mensaje accedido.

RETR *mensaje*

Es utilizado para recuperar un mensaje de mail.

TOP *mensaje cantidad_de_lineas*

Es utilizado para recuperar los headers del mensaje y cero o más líneas del mismo según se especifique en el parámetro *cantidad_de_lineas*..

NOOP

Este comando solo requiere una respuesta positiva del server.

DELE *mensaje*

Este comando permite marcar un mensaje como borrado. Los mensajes serán borrados físicamente cuando se cierre la sesión.

RSET

Este comando es utilizado para desmarcar los mensajes previamente marcados para borrado.

QUIT

Es utilizado para liberar la conexión con el server.

Respuestas a comando POP

Las respuestas en POP consisten de un indicador, una palabra clave y posible información adicional.

Actualmente existen dos indicadores: el positivo "+ OK" y el negativo "-ERR" . La respuesta a ciertos comandos son multilineas. En este caso después de enviar la primer línea (en la cual está el indicador) todas las líneas adicionales son transmitidas, cada una terminada con un CRLF. Cuando todas las líneas fueron enviadas una línea final es transmitida la cual contiene el caracter "." que indica la finalización de la respuesta.

Glosario

ESMTP: Extended Simple Mail Transfer Protocol - Describe las extensiones realizadas al estándar SMTP.

IANA: Internet Assigned Numbers Authority. Entidad responsable de asignar números en el Internet Suite.

IETF: Internet Engineering Task Force . Grupo de trabajo que intenta brindar soluciones a las necesidades de Internet en un corto plazo.

Internet Draft : Documentos en los cuales el IETF está trabajando (Aún no son definitivos).

Internet Suite: Conjunto de protocolos desarrollados por DARPA.

MTA : Message Transfer Agent . Una aplicación que ofrece transferencia de mensajes.

MTS : Un conjunto de MTAs conectados.

NNTP: Network News Transfer Protocol. Protocolo que ofrece servicio de transferencia de artículos

NVT ASCII : Network Virtual Terminal. Es la variante americana del ASCII e indica que en cada byte de información el bit de mayor orden será ignorado.

POP: Post Office Protocol. Protocolo utilizado para recuperar mail del mailbox del usuario.

RFC : Request For Comments. Serie de documentos que describen el Internet Suite y experimentos relacionados

SMTP: Simple Mail Transfer Protocol . Protocolo de Aplicación que permite transferencia de mensajes y servicio de entrega al mailbox del usuario.

U.A.: User Agent . Interface que usuario utiliza para interactuar con el sistema de mail.

Bibliografía:

I. The Internet Messaging

MARSHALL T. ROSE - Prentice Hall [ROSE 93].

II. Internetworking with TCP/IP Vol. III

(Client-Server programming and Applications)

DOUGLAS COMER - DAVID STEVENS - Prentice Hall [COMER 93].

III. TCP/IP Network Administrations

CARIG HUNTS - O'Reilly & Asoc [HUNTS 92].

IV. Redes de Ordenadores

ANDREW TANENENBAUM - Prentice Hall [TANEN 91].

V. Managing UUCP and Usenet

TIN O'REILLY - GRACE TODINA - O'Reilly & Asoc [O'REILLY 90].

VI. The Simple Book

MARSHALL T. ROSE - Prentice Hall [ROSE 91].

VII. Internetworking

MILLER - Prentice Hall [MILLER 90]

Proyectos Anteriores relacionados

- Agentes de Transmisión de Correo - Lic. Adrian Russo [RUSSO 94].
- S.I.M.P.L.E. - Lic. Ivana Harari - Lic. Claudia Banchoff [HARARI 93].

Request for Comments (RFC)

- RFC 821 : Simple Mail Transfer Protocol (SMTP)
- RFC 822 : Standard for the Format of ARPA Internet Text Messages
- RFC 976 : UUCP Mail Interchange
- RFC 977 : Network News Transfer Protocol (NNTP)
- RFC 1426 : SMTP Service Extension for 8bit-MIME Transport
- RFC 1521 : Multipurpose Internet Mail Extensions - Part 1 (MIME)
- RFC 1522 : Multipurpose Internet Mail Extensions - Part 2 (MIME)
- RFC 1651 : SMTP Service Extension
- RFC 1653 : SMTP Service Extension for Message Size Declaration
- RFC 1725 : Post Office Protocol (POP)

- RFC 1830 : SMTP Service Extension for Transmission of Large and Binary MIME Message

Internet Draft

- draft-ietf-notary-mime-delivery-04.txt: Enhanced Mail System Status Codes.
- draft-ietf-notary-mime-delivery-05.txt: SMTP Service Extension for DSN .
- draft-ietf-notary-mime-delivery-06.txt: An Extensible Message Format for DSN.

Publicaciones

- Messaging Magazine.
- Red Iris - Fundesco.

Especificaciones

- Windows Sockets Specifications Version 1.1 [**WINSOCK1**]
- Windows Sockets Specifications Version 2.0 [**WINSOCK2**]



Agradecimientos

Quiero agradecer fundamentalmente a mi mujer, Alicia, y a mi hijo, Juan Manuel por el tiempo que no les he dedicado, no solo para este proyecto sino en toda mi carrera.

A mi compañero y amigo de tanto tiempo, Miguel Luengo, con el cual compartí muchos años de mi vida.

A mi director, Javier Diaz, por el tiempo y esfuerzo dedicado para que este proyecto salga adelante.

Juan C. Marra

Deseo agradecer especialmente a Karina , por acompañarme y apoyarme en todo momento durante ocho años. A mi amigo y compañero Juan Marra, con quien compartimos los últimos años de la carrera y el esfuerzo de hacer este trabajo y quien, además, puso el impulso necesario en los momentos difíciles.

Finalmente, a nuestro director, Javier Diaz, por el tiempo que nos dedicó y por la oportunidad de integrarme a su grupo de trabajo.

Miguel A. Luengo

DOMICILIO.....	1900
\$.....	1900
Fecha..... 16/8/05	
Inv. E..... Inv. B..... 1906	

TES
95/1
DIF-01906
SALA



UNIVERSIDAD NACIONAL DE LA PLATA
Biblioteca
50 y 120 La Plata
catalogo.info.unlp.edu.ar
biblioteca@info.unlp.edu.ar



DIF-01906