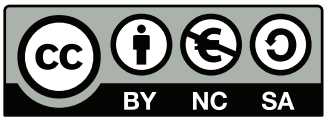
Fundamentos de Redes

LibreIM

Doble Grado de Informática y Matemáticas

Universidad de Granada

libreim.github.io/apuntesDGIIM



Fundamentos de Redes

LibreIM

Doble Grado de Informática y Matemáticas
Universidad de Granada

libreim.github.io/apuntesDGIIM

Índice

I. Teoría 6

1. Introducción 6

1.1. Sistemas de comunicación y redes 6
1.1.1. Estructura y elementos de una red 7
1.2. Diseño y estandarización de redes 7
1.2.1. **Modelo OSI** 8
1.2.2. **Modelo de referencia TCP/IP** 8
1.3. Terminología, conceptos y servicios 9
1.4. Internet: topología y direccionamiento 10

2. Tema 2. Servicios y Protocolos de Aplicación en Internet 12

2.1. Introducción a las aplicaciones de red 12
2.1.1. Protocolos TCP/IP 12
2.1.2. Arquitectura cliente/servidor 12
2.1.3. Procesos cliente y servidor 12
2.1.4. La interfaz Socket 13
2.1.5. Retardo en cola 13
2.1.6. Protocolo de aplicación 13
2.1.7. Características de las aplicaciones de red 14
2.2. Protocolos de transporte 14
2.3. Servicios de Nombres de Dominio (DNS) 15
2.3.1. Resolución distribuida 15
2.4. La navegación Web 16
2.4.1. Características HTTP 16
2.5. Correo Electrónico 17
2.5.1. Proceso de envío de correo electrónico 17
2.6. MIME 17
2.7. Protocolos Seguros 18
2.8. Aplicaciones multimedia 19
2.9. Aplicaciones para la interconectividad de redes locales 19

3. Tema 3. Capa de Transporte en Internet 20

3.1. Protocolo: User Datagram Protocol (UDP) 20
3.2. Protocolo: Transmission Control Protocol (TCP) 21
3.2.1. Control de la conexión: 22

3.2.2. Control de errores y de flujo: 23

4. Práctica 1 **26**

4.1. Información 26

4.1.1. Telnet 26

4.1.2. FTP 27

4.1.3. Apache 27

Parte I.
Teoría

1. Introducción

1.1. Sistemas de comunicación y redes

Un **sistema de comunicación** es una infraestructura (tanto hardware como software) que permite el intercambio de información.

Debemos saber que los **sistemas finales** o **hosts** son aquellos que tienen capacidad de procesar información; que un intercambio se considera eficaz si maximiza el uso de los recursos como el ancho de banda entre otros y que es transparente si actúa con independencia de las diferencias del sistema entre la fuente y el destino.

La **información** es el conjunto de datos con significado.

Una **red** es un un sistema de comunicación con sistemas finales (terminales) autónomos (con capacidad de procesar información) que facilita el intercambio eficaz y transparente de la información.

Razones para usar redes:

- Nos permiten compartir recursos
- Son escalables
- Son fiables y robustas -> Duplicidad (redundancia)
- Permiten ahorrar costes (computación distribuida)

Podemos clasificar las redes:

- Por escala:
  - LAN (Local Area Network) $\simeq$ 1km
  - MAN (Metropolitan Area Network) $\simeq$ 10km
  - WAN (Wide Area Network) $\geq$ 10km
- Por tecnología de transmisión o uso del canal de comunicación
  - Difusión o canal compartido (WiFi, Redes de Datos Móviles, BlueTooth...)
  - Punto a punto (Fibra, ADSL...)

Tenemos varios problemas a resolver respecto a la transparencia y eficacia en la transmisión de información con una red, entre ellos: El método de transmisión de la información y su traducción; compartición del medio y la segmentación de

la información; control del flujo y de errores, en el enlace y también extremo a extremo; control del encaminamiento o enrutamiento de los mensajes; control de congestión; entrega ordenada de los mensajes; gestión del diálogo o turno de palabra; representación o sintaxis de los datos; significado o semántica de los datos.

Aprenderemos a solucionar todo esto utilizando el modelo OSI (Open System Interconnection) de la ISO que pese a ser un modelo estándar tiene un interés puramente académico. Pese a ello, este modelo de 7 capas es la base del modelo TCP/IP que es el que realmente se utiliza a día de hoy.

1.1.1. Estructura y elementos de una red

- Los **hosts** son los sistemas finales (terminales) autónomos.
- La *subred* es la infraestructura para el transporte de información. Las subredes están formadas por líneas de transmisión y nodos que se conectan mediante nodos o elementos de conmutación (routers, switches o estaciones base).

1.2. Diseño y estandarización de redes

Problemas a resolver por la red (transparencia y eficacia):

- £Cómo enviar físicamente la información?
- Compartición del medio. Segmentación de la información
- Control de flujo y de errores, salto a salto y también extremo a extremo
- Control del encaminamiento (enrutamiento)
- Control de congestión
- Entrega ordenada de los mensajes
- Gestión del diálogo o turno de palabra
- Representación (sintaxis) de los datos
- Significado (semántica) de los datos

Conceptos de diseño en redes:

- Solucionar los problemas en capas
- Concepto de “Modelo de Referencia” -> definición de capas + funcionalidades

Principios de diseño para el modelo de referencia: - Funcionalidades distintas deben estar en capas distintas - Minimizar el flujo de información entre capas

Estándares internacionales:

1. **Modelo OSI** (Open System Interconnection) de la ISO
2. **TCP/IP** del Internet Engineering Task Force.

1.2.1. Modelo OSI

Grupo 1 de capas:

- **Capa Física**: Conexión ordenador-punto de acceso. Mueven los bits dentro de la trama de un nodo al siguiente.
- **Capa de Enlace**: Permite la comunicación entre dos nodos directamente conectados. Delimita la trama (nombre que se le da a los paquetes de la capa de enlace) y controla errores y flujo.

Grupo 2:

- **Capa de Red**: Nos ofrece la capacidad de enrutamiento que permite llevar a los paquetes desde un origen a un destino. Controla además la congestión, evitando el envío de paquetes a nodos congestionados. Incluye el protocolo IP, que es único y hace que se haga referencia a esta capa como la capa IP. Abstrae toda la comunicación para que únicamente se preocupe del nodo inicial y nodo destino como si estuvieran conectados directamente.

Grupo 3:

- **Capa de Transporte**: Transporta los mensajes entre los extremos finales de la comunicación. Existen dos protocolos de transporte, UDP y TCP. Proporciona control de errores y control de flujo. Resuelve problemas de semántica.
- **Capa de Sesión**: Controla el "turno de palabra" y las tareas de sincronización, es decir, decide quién y en qué momento envía información.
- **Capa de Presentación**: Permite una traducción desde el "idioma de la red" al "idioma de la aplicación", que se puedan entender los datos intercambiados. Incluye cifrado y compresión.
- **Capa de Aplicación**: Son las aplicaciones finales, que tienen su propio protocolo (http, ftp, por ejemplo). A los paquetes que transmite se les conoce como "*mensajes*".

1.2.2. Modelo de referencia TCP/IP

Grupo 1:

- **Red subyacente**.

Grupo 2:

- **Capa de Red**.

Grupo 3:

- **Capa de Transporte**.
- **Capa de aplicación**.

1.3. Terminología, conceptos y servicios

Modelo OSI: comunicación real frente a comunicación virtual.

En la siguiente imagen podemos ver que los routers solo operan hasta la 3ł capa.

Terminología:

- **Comunicación real** (vertical): refleja como cada capa se comunica con su capa inmediatamente inferior hasta que llega a la capa física que transmite la información y se comunica con la capa inmediatamente superior hasta llegar a la capa original.
- **Comunicación virtual** (horizontal): es como la capa N-ésima de la primera entidad que se comunica envía un mensaje a la N-ésima capa de la segunda entidad receptora.
- **Entidad del nivel N** (N en OSI del 1 = físico al 7 = aplicación)
- **Entidades pares**: están en la misma capa. Dialogan virtualmente entre ellas.
- **Protocolo**: conjunto de reglas que regulan el intercambio de información virtual entre entidades pares en un modelo de referencia dado
- **Interfaz**: división que hay entre las entidades (interfaz de aplicación y presentación)
- **Servicio**: funcionalidad que ofrece cada capa
- **Capa proveedora/usuaria del servicio**
- **Pila de protocolos**: sobre un modelo de referencia, elegir un protocolo para cada capa
- **Arquitectura de red = Modelo de referencia + Pila de protocolos**. La arquitectura de red es un modelo de referencia al que le asocio una pila de protocolo
- Compartir una arquitectura de red extremo a extremo garantiza el "intercambio de información transparente" entre hosts.
- **SAP** (Service Access Point)
- **SDU** (Service data Unit)
- **PDU** (Protocol Data Unit)

Retardos en la comunicación: es una magnitud que caracteriza la interacción entre los host. Mide el tiempo involucrado.

El **tiempo de transmisión** es lo que tardan los bits del paquete en emitirse. Depende de la velocidad de transmisión y el número de bits a enviar.

$$T_{transmision}(s) = \frac{L(bits)}{V_t(bps)}$$

El **tiempo de propagación** es el tiempo que el primer bit tarda hasta llegar a su destino. La velocidad de propagación depende del medio físico del enlace y se mide en metros por segundo. (Distancia a recorrer entre velocidad de propagación

$$T_{propagacion}(s) = \frac{D(m)}{V_p(m/s)} -> V_{propagacion}(m/s)$$

La fibra óptica aumenta el ancho de banda para poder transmitir más bits a la vez, y por tanto aumenta la velocidad de las transmisiones.

Además de estos tiempos, cuando un nodo recibe un paquete y lo procesa para reenviarlo se producen **retardos de procesamiento**, cuando el nodo recibe dicho paquete y procesa su información, y **retardos de cola**, cuando el mensaje espera en una cola a que todos los mensajes que llegaron antes que él sean enviados.

Servicios:

- **Orientado a conexión (SOC)**. Tales como TCP, su objetivo es evitar las pérdidas. Se abre una conexión y reserva recursos, haciendo que cliente y servidor intercambien información de control antes de que se produzca el intercambio a nivel de aplicación. Hasta que no conteste el otro extremo, no se puede transmitir información. Se usa en aplicaciones donde se quieran evitar las pérdidas de información.
- **No orientado a conexion (SNOC)**. Tales como UDP, proporciona unos servicios básicos, sin establecer una conexión previa ni reservar recursos, por lo que no garantiza la correcta entrega del mensaje. Se usa en aplicaciones de tiempo real.
- **Confirmado (fiable)**. Pueden garantizar la entrega sin errores y en el orden correcto. Necesita realimentación positiva y tienen implícito un mecanismo de confirmación
- **No confirmado (no fiable)**. No garantizan la entrega correcta del mensaje. No hay puntero

1.4. Internet: topología y direccionamiento

Existe una topología jerarquica en el sistema por el cual nos conectamos a internet:

- **Intranets (Ethernet) del usuario**: zona pública + zona privada
- **Redes de acceso (xDSL, RDSI, FTTH...) del ISP (Internet Service Provider)**: Cable telefónico (xDLS,ISDN), cable (coax), HFC(Hybrid Fiber Coax), FTTH(Fiber-To-The-Home)
- **Redes troncales (ATM, SDH, SONET, etc) de grandes operadores de telecomunicaciones**.
- **Acuerdos de Peering y Tránsito**. En Peering no se paga, en tránsito sí.
- **Tier1, Tier2 y Tier3**.
- **Puntos neutros o PoP (Point of Presence) o IXP (Internet eXchange Point)**: garantizan el acceso a muchos Tiers.

Tipos de Tier de ISP:

- **Tier 1**: Los grandes proovedores hacen acuerdos de peering para no cobrarse los servicios mutuamente, sino que solo gestionan el tráfico entre ellos. Sólo tienen acuerdos de peering. Se conocen como redes troncales y tienen cobertura internacional (pueden llegar a cualquier IP).
- **Tier 2**: Tienen tanto acuerdos de peering como pagos a otros ISP más grandes que puedan gestionar su tráfico. Tienen cobertura regional/nacional.
- **Tier 3**: Pagan por los acuerdos de tránsito. No pueden permitirse negociar para hacer peering. Necesitan conectarse a internet a través de un ISP de tier 2.

Para conocer cómo se dirigen las conexiones y cómo se identifican los destinatarios se usan diversos elementos. La dirección IP, en la capa de red, identifica a los hosts. Para los humanos es más sencillo usar los nombres de dominio, los cuales traducen las direcciones del lenguaje humano a su correspondiente dirección IP.

Niveles de direccionamiento: hay uno por cada capa.

- **URL**: capa de aplicación.
- **Puertos**: identifica el proceso origen y destino -> capa de transporte.
- **Dirección IP (identifica los hosts)**: capa de red (dirección IP origen y destino).

2. Tema 2. Servicios y Protocolos de Aplicación en Internet

2.1. Introducción a las aplicaciones de red

2.1.1. Protocolos TCP/IP

Todas las aplicaciones finales que usamos se basan en otros protocolos. Los protocolos que use la aplicación estarán basados en otros de transporte como UDP y TCP y estos a su vez en protocolos de internet como IP, que son los que finalmente tienen acceso a la red.

Hay un modelo de referencia y una pila de protocolos. A cada capa (Internet, transporte y aplicación) se le asocia protocolos:

2.1.2. Arquitectura cliente/servidor

Generalmente se comportan de acuerdo a una estructura cliente-servidor. En este existe un host que siempre está activo, el servidor, con IP permanente y pública. Y este presta un servicio a las solicitudes de muchos otros hosts, que son sus clientes, con IP que puede ser dinámica y privada y no se comunican entre sí. Estos, por su parte, pueden estar activos de manera permanente o intermitente.

Un **servidor** siempre está en funcionamiento, tiene IP permanente y pública. Se agrupan en granjas o clústeres, también denominados centros de datos, que permiten dar soporte a todas las peticiones de sus clientes sin ser desbordados.

Los **clientes** funcionan intermitentemente, pueden tener IP dinámica y privada, se comunican con el servidor pero no se comunican entre sí.

2.1.3. Procesos cliente y servidor

El **proceso cliente** inicia la comunicación y el **proceso servidor** espera a ser contactado (IP permanente y pública).

El proceso cliente es el que inicia la comunicación, mientras que el proceso servidor es el que espera a ser contactado. Es por esto por lo que necesita tener una IP permanente y pública. Además, para recibir mensajes, un proceso debe tener un identificador, compuesto por una IP y un puerto.

> Nota: una IP pública es única asignada de forma permanente y solo se puede usar bajo solicitud a la autoridad.

Un proceso envía/recibe mensajes a/desde su **socket**. Para recibir mensajes un proceso debe tener un **identificador** (IP +puerto). Por ejemplo: servidor web gaia.cs.umass.edu con IP 128.119.245.12 y número de puerto 80.

2.1.4. La interfaz Socket

La comunicación entre diversos hosts se realiza mediante el uso de sockets por parte de los procesos que se encuentran tanto en servidor como en el cliente.

Un **socket** es un descriptor de una transmisión a través del cual la aplicación puede enviar y/o recibir información hacia y/o desde otro proceso de aplicación. Es decir, es la interfaz ("puerta" de acceso) entre la aplicación y los servicios de transporte. La aplicación del lado emisor empuja los mensajes a través del socket. En el otro lado del socket, el protocolo de la capa de transporte tiene la responsabilidad de llevar los mensajes hasta el socket de recepción. En la práctica, un socket es una variable tipo puntero que apunta a una estructura:

2.1.5. Retardo en cola

Para estimar los retardos (tiempos) en cola se usa teoría de colas: el uso de un servidor se modela con un sistema M/M/1

El retardo en cola es:

$$R = \frac{\lambda (T_s)^2}{1 - \lambda T_s}$$

Donde T_s (distribución exponencial) es el tiempo de servicio y λ (Poisson) es la ratio de llegada de solicitudes.

Esta expresión se puede usar para calcular el retardo en cola en un router.

2.1.6. Protocolo de aplicación

Los protocolos están definidos por el **tipo de servicio** (orientado o no a conexión, o realimentado o no), el **tipo de mensaje** que emita (request, response), su **sintaxis** (definición y estructura de campos en el mensaje. En aplicación generalmente son orientados a texto (HTTP) aunque hay excepciones (DNS) y tienden a usar formato Type-Length-Value), **semántica** (significado de los campos) y sus **reglas** (cuándo los procesos envían mensajes/responden a mensajes).

Los **tipos** de protocolos son:

- **Protocolos de dominio público** (definidos en RFCs (Request For Comments), por ejemplo: HTTP, SMTP, FTP, IP...) **versus propietarios** -> (por ejemplo Skype, IGRP).
- **Protocolos in-band vs out-of-band**: cuando la gestión se realiza por la misma vía que la comunicación frente a cuando se usa una vía paralela. Envía datos e información de control. In-band solo utilizan una transmisión

para enviar datos como control, usa el mismo socket para ambos. Out-of-band indica que la señalización de control va por un lado y la del dato por otro.
- **Protocolos state-full vs stateless**: cuando la aplicación guarda información sobre todo lo que ocurrió desde el inicio de la misma frente a cuando no la guarda. Por ejemplo, cuando descargamos páginas web con HTTP es stateless porque no se fija en mi historial.
- **Protocolos persistentes vs no-persistentes** (sobre servicios SOC – Servicios Orientados a conexión): cuando el protocolo hace uso de conexiones persistentes frente a cuando no (hace una para cada objeto).

2.1.7. Características de las aplicaciones de red

Una aplicación debe de tener unos requisitos o características:

- **Tolerancia a pérdidas de datos (errores):** algunas aplicaciones pueden tolerar las pérdidas de datos, tales como streamings de audio/vídeo, pero otras deben de asegurar la fiabilidad de la transferencia (transferencia de archivos, FTP, telnet, HTTP).
- **Exigencia de requisitos temporales:** también pueden necesitar el mínimo retraso (*delay*) para ser efectivos. Algunas apps denominadas *inelásticas* (telefonía Internet, juegos interactivos) requieren retardo (*delay*) acotado para ser efectivas, otras aplicaciones no. Un streaming también tiene el requisito de que este retraso no sea excesivo, o en los videojuegos es necesaria esa sincronización para evitar el lag.
- **Demanda de ancho de banda (tasa de transmisión o throughput):** algunas apps también necesitan un ritmo determinado de envío de datos (*codec* de vídeo) y otras no.
- **Nivel de seguridad:** la encriptación, autenticación y no repudio (no puedes negar ser el remitente de un envío de datos) son factores importantes en las aplicaciones.

La conclusión es que las distintas aplicaciones tienen requisitos **heterogéneos**.

2.2. Protocolos de transporte

En la capa de transporte existen diversos protocolos:

- **Servicio TCP:** está orientado a conexión (establecer una conexión entre los dos involucrados previo al envío), este transporte es fiable ante pérdidas (control de errores), con control de flujo y de congestión.
- **Servicio UDP:** no está orientado a conexión, es decir, no se comprueba que ambos estén preparados para realizar la comunicación. El transporte no es fiable, no tiene control de flujo ni de congestión.

Estos al ser usuarios del protocolo IP (capa de red) **no garantizan calidad de servicio** (QoS): retardo no acotado, las fluctuaciones no acotadas, no hay una velocidad de transmisión mínima garantizada, no hay una probabilidad de pérdidas acotada ni hay garantías de seguridad.

2.3. Servicios de Nombres de Dominio (DNS)

Es un servicio (implementado en un servidor) que se encarga de traducir los nombres a direcciones IP.

Tiene una estructura jerárquica en dominios:

ParteLocal.dominioNivelN.(...).dominioNivel1

Donde Nivel1 es el dominio genérico.

La ICANN se encarga de delegar los nombres y números asignados.

Cuando un host quiere solicitar una determinada página a un servidor web introduce el nombre de dicho servidor web. El navegador extrae el nombre del host a partir del URL y lo pasa a la aplicación DNS. El cliente DNS envía una consulta con dicho nombre al servidor DNS, tras procesarlo, el servidor le responde con una dirección IP correspondiente, mediante el cual ya puede iniciar la conexión.

2.3.1. Resolución distribuida

El ordenador original no resuelve todo el nombre del dominio. Este conecta con un servidor que será el encargado de conectar iterativamente con el resto de servidores.

De esta manera nos conectaríamos con los servidores ".", los de dominio (Top-Level Domain, TDL), servidores Locales y servidores Autorizados y Zona.

Un host solicitaría la dirección de una URL (www.una.direccion.com) a su servidor local. Este envía la petición a un servidor raíz, el cual toma el sufijo (.com) y le responde al DNS local una lista de direcciones responsables de dicho dominio (los responsables del sufijo .com). Estos responsables son servidores TLD (top level domain) y a continuación se les envía una petición a estos. El servidor TLD examina el sufijo (direccion.com) y responde con la dirección del servidor DNS autorizado que puede dar la dirección del URL inicial. Después el servidor local consulta a dicho servidor DNS autorizado y este le responde con la dirección IP de la URL inicial (www.una.direccion.com).

Gestión de la base de datos DNS:

Como hemos comprobado la base de datos está distribuida. Esto implica que cada zona debe tener al menos un servidor de autoridad, y en cada zona habrá servidores primarios y secundarios, que bien tendrán la copia master de la base de datos o bien la obtendrán a partir de los primarios. Existen sólo 13 servidores raíz. Existe un servicio de caché que permite agilizar las consultas.

Los servidores pueden dar una **respuesta con autoridad**, si tiene autoridad

sobre la zona en la que se encuentra el nombre solicitado y devuelve la dirección IP. Una **respuesta sin autoridad**, cuando no tienen autoridad pero tienen la respuesta en caché. O bien puede no conocer la respuesta, lo que implicaría una consulta a un servidor superior.

2.4. La navegación Web

El protocolo de transferencia de hipertexto, **HTTP**, es el protocolo de la capa de aplicación de la web y se encuentra en el corazón de la Web. Este se implementa en dos programas, un cliente y un servidor.

Una página web es un fichero (HTML) formado por objetos, que pueden ser ficheros HTML, imágenes, applets y demás tipos de archivos. Cada objeto se direcciona con una URL. La mayoría de las páginas web tienen un archivo base HTML donde se referencian los objetos que están contenidos en esa web. Tiene su **puerto bien definido**, el 80.

El protocolo HTTP sigue un modelo cliente-servidor. El cliente es el que pide, recibe y muestra objetos web mediante el browser. El servidor por su parte es el que envía los objetos web en respuesta a peticiones.

2.4.1. Características HTTP

TCP al puerto 80: Inicio de conexión TCP, envío HTTP, cierre de conexión TCP.

HTTP es "stateless" → Cookies: El servidor no mantiene la información sobre las peticiones de los clientes. Esto puede implicar, por ejemplo, que cuando recibe dos peticiones idénticas del mismo cliente devuelve el objeto solicitado en lugar de devolver ningún tipo de error o mensaje informativo. Para identificar a los usuarios y su actividad se utilizan cookies, archivos que se almacenan en el sistema terminal del usuario y son gestionados por su navegador.

La conexión puede ser persistente o no persistente. En el primer caso se pueden enviar múltiples objetos sobre una única conexión TCP entre cliente y servidor, mientras que la no persistente crea nueva conexión para cada objeto a enviar.

El persistente tiene un tiempo de transmisión total menor que el no persistente. Pero el no persistente permite gestionar mejor los recursos del servidor, pues no tiene que mantener el socket abierto durante toda la conexión, a cambio, al tener que establecer una conexión por objeto reduce su velocidad.

Hay dos tipos de mensajes HTTP: request y reponse. La petición de un elemento y su concesión. Cada uno de ellos tiene un formato específico, donde se indica la información concreta que se desea solicitar, o, en caso de ser desarrolladores de la página, mensajes de gestión. (GET,POST,HEAD,PUT,DELETE) Las respuestas se asemejan a las peticiones en cuanto a la indicación de un estado y cabecera, pero adicionalmente poseen el cuerpo de la entidad, donde se encuentra el objeto solicitado. La línea de estado indica un código de respuesta (200 OK, 301 moved permanently, 400 bad request, 505 HTTP version not supported).

Caché: Cuando se solicitan numerosas veces algo a un servidor es conveniente configurar un proxy intermedio que almacene dicha solicitud. De ese modo quien solicite ese mismo servicio no involucrará al servidor original sino al proxy, este sólo le envía una petición al servidor original para saber si es necesario actualizar la caché o no. Si no dispone de esta solicitud tendrá que inevitablemente solicitar al servidor original.

2.5. Correo Electrónico

2.5.1. Proceso de envío de correo electrónico

El correo electrónico es un servicio de red que perminte a los usuarios enviar y recibir mensajes y archivos rápidamente mediante sistemas de comunicación electrónicos. En el correo electrónico intervienen dos clientes que enviarán y recibirán el correo cuando ellos decidan. La arquitectura del sistema de correo se basa en subsistemas consistentes de agente de transferencia (mueve mensajes del origen al destino, por lo general son demonios) y agente de usuario (programas locales que permiten al usuario leer y enviar correo). El procedimiento es el siguiente:

El usuario de origen utiliza su user agent para mandar el correo a su servidor de correo, se envía mediante SMTP o HTTP. El protocolo SMTP consiste en una conexión TCP con el servidor de correo destinatario mediante el puerto 25. El servidor de destino almacena el mensaje en la bandeja de entrada del usuario destino. El destinatario usará su agente de usuario arbitrariamente para leer el mensaje utilizando los protocolos POP3 (Post Office Protocol), IMAP (Internet Mail Access Protocol) o HTTP.

La diferencia entre POP3 e IMAP es que POP3 no puede ser usado por usuarios "nómadas" que quieran acceder desde distintos hosts a su correo, pues una vez recibidos los archivos, estos sólo permanecen en el receptor. IMAP permite mantener una carpeta bandeja de entrada en un servidor IMAP, donde puede gestionar ahí su correo. A causa de esto IMAP requiere mayor tiempo de conexión y utiliza más recursos del servidor.

El principal problema de SMTP es que no requiere autenticación, lo que permite que cualquiera pueda enviar correo a cualquier persona o grupo de personas. Esta característica hace posible la existencia de correo basura o spam. Los servidores SMTP modernos intentan minimizar este comportamiento permitiendo que solo hosts conocidos accedan al servidos SMTP. Los servidores que no ponen tales restricciones se llaman open relay.

2.6. MIME

MIME (Multipurpose Internet Email Extension) es un estándar de intercambio a través de Internet de todo tipo de archivos. Fue diseñado para e-mails pero

finalmente es tambien importante en protocolos de comunicación externos al email.

Cuando se envía un mensaje con este protocolo, este mensaje tiene una serie de campos para facilitar la comprensión del mensaje, tales como:

- Content-Transfer-Encoding: El método de codificación, como ASCII7, ASCII 8, binaria o base64.
- Content-type: El tipo de datos que se van a transferir, que puede ser Text, Image, Audio, Application, Message...
- Content-ID: identificador único

2.7. Protocolos Seguros

Hay unos aspectos destacables de la seguridad en internet:

- **Confidencialidad:** Sólo quien esté autorizado a acceder a la información puede hacerlo.
- **Responsabilidad:** Autenticación, confirmar que los agentes de comunicación son quien dicen ser. No repudio: No se pueda negar la autoría de una acción y que la acción se ha hecho. Control de accesos: Garantía de identidad para el acceso.
- **Integridad:** La información no debe ser manipulada.
- **Disponibilidad:** El acceso a los servicios debe estar regulado para evitar, por ejemplo, ataques Denial Of Service.

También hay un conjunto de mecanismos de seguridad:

- **Cifrado Simétrico:** Utilizamos la misma clave para cifrar/descifrar (DES,3DES,AES,RC4).
- **Cifrado Asimétrico:** Una clave distinta para encriptar y para desencriptar (Diffie & Hellman, RSA).
  Se pueden combinar simétrico y asimétrico, le envío al receptor la clave del cifrado simétrico cifrada con su clave pública, él la desencriptará con su privada y así compartiremos la clave simétrica para aumentar la velocidad de las comunicaciones.
- **Message Authentication Code:** Hay que impedir que se modifiquen los archivos de envío durante el mismo. El mensaje se pasa por un hash con una clave que no se puede modificar sin tener la misma clave. (MD5, SHA-1...)
- **Firma Digital:** Se pasa por un hash el mensaje, se cifra con la clave privada y cualquier receptor puede descifrar la firma y comparar el resultado con el hash del mensaje recibido, si coinciden se garantizaría la autoría.
- **Certificado:** Para asegurar que las claves pertenecen al remitente existen terceros que lo garantizan. Estos terceros son confiados y aceptados "públicamente".

Diversos **ejemplos** de seguridad criptográfica en distintas capas son:

- **Capa de aplicación:** PGP (Pretty Good Privacy), SSH (Secure Shell).
- **Capa de sesión** (entre aplicación y transporte): SSL (Secure Socket Layer) ,TSL (Transport Secure Layer).
- **Capa de Red:** IPSec(VPN)

También existen elementos de seguridad Perimetral y Gestión de riesgos:

- **Firewall:** Sistema que limita los accesos a los elementos de nuestro ordenador. También UTMs se están implementando últimamente, son una extensión de firewalls.
- **Sistemas de detección de intrusiones** IDS. En red NIDS, en host HIDS. Analizan el tráfico y detectan posibles ataques y anomalías.
- **Antivirus, evaluación de vulnerabilidades, seguridad en Aplicaciones**, filtrado web/anti-spam.
- **Advanced Thread Detection:** Listas donde se comparte información de seguridad. Así se permite enterarse cuanto antes de problemas de seguridad.
- **SEMs, SIEMs:** Realiza el conjunto de las funcionalidades anteriores.

2.8. Aplicaciones multimedia

Son de audio y vídeo. Se procura mejorar la calidad del servicio.

Las aplicaciones de flujo almacenado no dan problemas de delay, sin embargo el throughput es más exigente. Sin embargo el delay en emisores en directo no pueden tener delay, su velocidad de transmisión sigue siendo igual de importante.

Todas estas requieren un alto ancho de banda. Son tolerantes a pérdidas de datos. Tienen un Delay y un Jitter(variabilidad del delay) acotado. Hacen uso además de multicast (Ejemplo YouTube en las diapositivas).

2.9. Aplicaciones para la interconectividad de redes locales

- **DHCP:** Nos permite configurar dinámicamente direcciones IP
- **DynDNS, No-IP:** Servicios en la red privda con IP pública variable, configuración de acceso necesaria
- **UPnP:** “Pervasive adhoc com”. Comunicación dispositivo <-> NAT

3. Tema 3. Capa de Transporte en Internet

Como sabemos un protocolo de la capa de transporte proporciona una abstracción de la comunicación de manera que podemos operar con dos hosts distantes como si estuvieran conectados directamente.

Estos protocolos están implementados en los hosts terminales, pero no en los routers de la red. La capa de transporte **transforma** el mensaje en paquetes de capa de transporte conocidos como **segmentos**. Esto por lo general se hace dividiendo el mensaje original en fragmentos más pequeños y añadiendo una cabecera a cada uno de ellos. Tras esto la capa de transporte pasa a la capa de red, que gestiona con paquetes de capa de red el envío (en un datagrama).

En internet hay dos protocolos que ofrecen distintos tipos de servicios para la capa de transporte, **TCP y UDP**.

Multiplexación y demultiplexación:

Un concepto importante para entender estos protocolos es saber cómo se comunican las aplicaciones, los procesos, con la capa de transporte, pues estos no se comunican directamente, sino que hacen uso de **sockets** intermediarios. A la hora de enviar un mensaje un proceso pasa a los sockets la información, la capa de transporte coge la información de estos sockets para crear los segmentos y pasarlos a la capa de red. Esta es la **multiplexación**. El procedimiento de recepción, donde la capa de transporte del receptor obtiene la información de la capa de red y la entrega a los sockets correspondientes es la **demultiplexación**.

Cada uno de estos protocolos usa distintos tipos de sockets, con distinto funcionamiento. A continuación se explicarán estos protocolos.

3.1. Protocolo: User Datagram Protocol (UDP)

Es un servicio de entrega de mejor esfuerzo (best effort), pues no garantiza la correcta entrega de todos los segmentos, pero hará todo lo que pueda para ello.

No es un servicio orientado a conexión, lo que implica que no es fiable y que ni hay garantías de entrega ordenada ni control de congestión, pues su principal desempeño es entregar la información tan rápida como se pueda.

Entre sus ventajas destacan el control a nivel de aplicación sobre los datos que se envían y cuándo se envían, pues en cuanto lo indica la aplicación, UDP los empaqueta y los envía inmediatamente, sin mecanismos de control de flujo. Además, al no establecer ningún tipo de conexión evita retardos por ese motivo. Todo esto tiene adicionalmente una menor sobrecarga en la cabecera de los paquetes.

Estos motivos hacen de UDP un protocolo ideal para aplicaciones tolerantes a fallos y sensibles a retardos, como las aplicaciones multimedia o, incluso, DNS.

Las tareas de multiplexación y demultiplexación se realizan asignando un puerto disponible al socket del emisor, indicando el puerto de destino (y la dirección IP), entonces se realiza el mejor esfuerzo para entregar el datagrama. Cuando llega el paquete este es examinado en el destino y se entrega al socket apropiado según el número de puerto asociado. El socket UDP consta únicamente de puerto y dirección IP de destino.

3.2. Protocolo: Transmission Control Protocol (TCP)

Es un servicio orientado a conexión, con una entrega ordenada garantizada. Es además full-duplex. TCP garantiza una entrega fiable sobre una capa de red no fiable (IP).

La multiplexación y demultiplexación varían frente a UDP, aquí los sockets están identificados por cuatro elementos: Puerto e IP de destino y puerto e IP de origen.

Tiene un sistema de control de flujo de detección y recuperación de errores (ARQ, Automatic Repeat reQuest). Esto se implementa mediante temporizadores que esperan la confirmación de la recepción (ACK, que es acumulativo, es decir, si se recibe un ack posterior es porque se ha recibido todo lo anterior correctamente). Si se agota el tiempo se reenvía el paquete. Trabaja sobre ventanas adaptables que veremos más adelante.

Utiliza "piggybacking", es decir, en un paquete que va en un sentido de la comunicación se añade información sobre el otro sentido de la comunicación. Así aprovechan los paquetes de datos para incluir información de control.

Todas estas propiedades nos aseguran que es fiable en el control de congestión y flujo.

La información a enviar por TCP se divide en segmentos TCP. Cada uno de esos segmentos contiene **información del puerto origen y destino**. Además contiene información relativa a sí misma y a la posición que ocupa esta información respecto al total, **número de secuencia y número de acuse (ACK)**, el primero es el número del primer byte del segmento dentro del flujo de bytes que se inicializa a un valor aleatorio elegido por los hosts, mientras que el segundo es un valor que indica el número de secuencia que el receptor está esperando recibir, es decir, el siguiente byte a leer. Junto a esto se incluye la **longitud de la cabecera del paquete**, junto a un espacio reservado para usos concretos, flags, bytes acerca del control de flujo, comprobación (**checksum**) y un puntero a datos que hay que incluir de manera urgente, por ejemplo datos de control para subir a la capa de aplicación.

(Un ejemplo de esto es un servicio streaming al que se le solicita cambiar la compresión del vídeo debe de procesarse antes que el resto de los datos sobre el vídeo que se deben de procesar "más tranquilamente").

La cantidad de información de la capa de aplicación que se transmite en cada uno de estos paquetes está limitada por el tamaño máximo del segmento (**MSS**, Maximum Segment Size), que generalmente está definido por la longitud de la trama más larga de la capa de enlace que el host emisor puede enviar (también conocida como unidad máxima de transmisión, MTU, Maximum Transmission Unit).

3.2.1. Control de la conexión:

El intercambio de información tiene tres fases.

1. **Establecimiento de la conexión** (sincronizar el número de secuencia y reservar recursos). Este es el denominado acuerdo en tres fases o *three-way handshake*. Consiste en el envío de un segmento al receptor que no contiene información de la capa de aplicación (un número aleatorio, paquete SYN), una respuesta de este al emisor, indicando que ya está disponible para la recepción (que devuelve un valor una unidad mayor, como reconocimiento, SYN-ACK), y un tercer segmento que puede llevar carga útil (al receptor se le envía otro valor una unidad mayor que la última recibida, a modo de reconocimiento, para que ambos hayan recibido un reconocimiento por parte del otro, ACK), este último no necesita ser respondido.
2. Intercambio de datos (full-duplex). Full-duplex indica que este intercambio de datos puede ser bidireccional, un host puede estar enviando datos a otro mientras está recibiendo datos del mismo.
3. Cierre de la conexión (liberar recursos). Proceso que se asemeja al three-way handshake.

Se abre activamente el cliente y pasivamente el servidor. En esto se ven involucrados diversos bits y campos de control.

El proceso que se realiza para finalizar la conexión consiste en lo siguiente:

Para liberar todos los buffers y variables del host el cliente decide cerrar la conexión. Este ejecuta un comando de cierre que hace que el cliente TCP envíe un segmento especial al proceso servidor. Este segmento contiene el bit FIN activo y el servidor le responde con el reconocimiento. Posteriormente el servidor envía su propio segmento de desconexión, con el bit FIN activo. El cliente lo recibe y envía el reconocimiento para que ambos queden con sus recursos liberados.

Otros detalles: Los campos del control de conexión tienen 32 bits, osea 2^{32} valores.

La inicialización se inicia por el ISN, que es elegido por el sistema. Los campos del control de conexión tienen 32 bits, osea 2^{32} valores. El sistema lo elige, y el estándar sugiere utilizar un contador entero incrementado en uno por cada 4 microsegundos. Esto protege de coincidencias, pero no de sabotajes.

El incremento se realiza según los bytes de carga útil (payload). El nuevo número de secuencia se genera a partir de la suma del número de secuencia

anterior más el número de bytes de carga útil. Los flags SYN y FIN incrementan en 1 el número de secuencia.

Ejercicio: Se desea transferir con protocolo TCP un archivo de L bytes usando un MSS(Maximum Segment Size) de 536.

a) £Cuál es el valor máximo de L tal que los números de secuencia de TCP no se agoten?
b) Considerando una velocidad de transmisión de 155 Mbps y un total de 66 bytes para las cabeceras de las capas de transporte, red y enlace de datos, e ignorando las limitaciones debidas al control de flujo y congestión, calcule el tiempo que se tarda en transmitir el archivo en

A.

a) Un segmento tiene 2^{32} valores distintos, pero luego tiene dos bits del SYN y del FIN. En total el valor máximo de L para que no desborde es $2^{32} - 2$.
b) Dados los bytes de las cabeceras buscamos obtener el total y luego dividimos por la velocidad de transmisión.

Dividimos L entre el tamaño del segmento para saber cuántos segmentos tenemos. Cogemos el entero inmediatamente superior de $\frac{2^{32}-2}{536}$ para calcular el número de segmentos. Luego calculamos el número de bytes así: N_segmentos * 66 + L

Conocidos el número de bytes a transmitir simplemente tenemos que dividir por la velocidad $\frac{(N_{segmentos} \cdot 66 + L) \cdot 8}{155 \cdot 10^6 bps}$

3.2.2. Control de errores y de flujo:

El control de flujo regula ámbitos como la velocidad de transmisión, mientras que el de errores comprueba si se transmiten correctamente los paquetes.

Después del three-way handshake comienza la conexión. Un paquete (con su retardo de transmisión al ser enviado) se transmite al receptor (que lo recibe en un tiempo de propagación. El receptor le responde con un paquete minúsculo que sólo tendría un breve retardo de propagación (paquete con las cabeceras).

Por tanto el tiempo total sería igual a dos veces el retardo de propagación, más el tiempo de tranmisión del paquete inicial, más el tiempo de transmisión del paquete de respuesta (tiempo despreciable), más el tiempo de recepción de este último (también despreciable).

En la práctica se realizan envíos en ventanas de varios paquetes como este. Es una **ventana deslizante** pues cada vez que recibe la confirmación de la recepción de un paquete permite enviar uno nuevo, manejando así un número fijo de paquetes.

Control de errores:

Estudia el checksum de los paquetes, si el receptor desecha un paquete, el emisor, tras acabar el temporizador del paquete sin haber recibido una confirmación,

reenvía el paquete. Si se envían dos paquetes y se recibe el ACK del último que se envió entonces queda confirmada la correcta entrega del paquete anterior.

Existen una serie de reglas de generación del ACK para cuando ocurren determinados eventos para evitar cualquier tipo de error:

- Cuando llega el segmento de manera ordenada con todo lo anterior ya confirmado: Esperar un tiempo determinado (hasta 500 ms) por si se recibe un nuevo segmento enviar el ACK de este último y así ahorrar un envío.
- Cuando llega un segmento de manera ordenada pero hay pendiente un ACK retrasado: Se envía el ACK único acumulativo, de este modo nos aseguramos de que no se reciban más de dos paquetes sin confirmar el ACK.
- Cuando llega desordenado el segmento con un número de secuencia mayor que el esperado, implicando una discontinuidad: Se envía un ACK repetido, con el valor del siguiente byte de la secuencia esperado.
- Cuando llega un segmento que completa una discontinuidad parcial o totalmente: Confirma el ACK inmediatamente si el segmento comienza en el extremo inferior de la discontinuidad.

Los timeouts del control de errores varían dinámicamente. Esto se realiza mediante una estimación de la situación en la red.

En primer lugar el timeout tiene que ser mayor que el tiempo de ida y vuelta (RTT). Al menos dos veces el tiempo de transmisión. Luego hay que controlar si es demasiado corto, pues produciría timeouts prematuros, o demasiado largo, pues generaría grandes esperas innecesarias.

Para calcular el RTT que usaremos para estimar el timeout usamos la siguiente fórmula:

$$x_t = x_{t-1}\alpha + y_t(1-\alpha); \alpha \in [0,1)$$

Donde x_t es el RTT estimado en un instante determinado , x_{t-1} es el cálculo previo del RTT estimado y y_t el RTT que se acaba de medir. α será el peso de la media ponderada que le otorgará más peso a las medidas antiguas o a la reciente.

Control de flujo: Los datos que se reciben correctamente pasan de los segmentos a un buffer de recepción, pero estos sólo pasaran a la capa de aplicación cuando los recursos correspondientes estén libres. De este modo TCP proporciona este procedimiento para evitar que el emisor sature al receptor.

Es un esquema crediticio el receptor avisa al emisor de lo que puede aceptar.

Se proporciona un servicio de control de flujo manteniendo en el emisor una variable conocida como **ventana de recepción** (al ser full-duplex hay una ventana a cada lado). Se utiliza el campo ventana (WINDOW) en el segmento TCP para establecer la ventana ofertada.

El receptor responde al emisor con el número de bytes que tiene libre en su ventana, si este le responde con 0 es que no puede recibir más paquetes. El emisor

tiene que esperar a recibir un nuevo paquete con el mismo ack pero con un valor de window mayor que cero.

Control de congestión:

Actuando de manera parecida al control de flujo da respuesta a los problemas que pueda causar la congestión de la red IP. La finalidad de este es evitar que el emisor llegue a saturar la red. (Tanto el ancho de banda de las líneas como los buffers en los dispositivos de interconexión).

La principal solución que se propone es limitar el tráfico generado. Y esto se puede estudiar de distintas maneras:

- Control de congestión terminal a terminal: Cuando la capa de red no proporciona un soporte explícito a la capa de transporte esta se guía por comportamientos observados a través de la red, tales como la pérdida de paquetes o los retrasos.
- Control de congestión asistido por la red: Los routers proporcionan una realimentación explícita al emisor informando de la congestión de la red, y puede ser tan simple como informar de la existencia de congestión en algún enlace de la red mediante el uso de un bit.

TCP ha de usar el primer tipo de control, pues IP no proporciona una realimentación explícita a los sistemas terminales en cuanto a congestión de la red.

En el emisor se usa una **ventana de congestión** y un umbral, inicialmente VCongestion = MaximumSegmentSize, y Umbral es un valor arbitrario inicializado por el emisor y ambos son regulados cuando se produce algún timeout, es decir, TCP es auto-temporizado. Esta ventana limita la velocidad de transmisión para evitar la congestión. Si se pierde un segmento se deberá reducir la ventana, si se recibe el ACK correctamente puede aumentarse.

Esto se define por el **algoritmo de control de congestión de TCP**, que tiene los siguientes estados:

1. **Arranque lento:** Cuando se inicia una conexión TCP el valor inicial de la ventana es igual al MSS, como hemos dicho antes. Puesto que lo ideal sería poder aprovechar al máximo el ancho de banda, para esto se aumenta el tamaño de la ventana, agrandándola en tantos MSS como reconocidos en ese momento, "duplicándola" de este modo cada período de tiempo RTT. Por esto decimos que tiene un arranque lento, pero un crecimiento exponencial. Este crecimiento se detiene cuando se produce una pérdida de paquete (señalada por un timeout), que reduce el tamaño de la ventana a la mitad y otorga este valor al umbral. También puede estar limitada por un umbral tomado con anterioridad. Cuando se alcanza o sobrepasa el valor del umbral se finaliza el arranque lento y se pasa a la prevención de la congestión.
2. **Prevención de la congestión:** Al entrar en este estado en lugar de duplicar el valor de la ventana, simplemente se aumenta el tamaño de esta en

un MSS. Generalmente cuando llega un paquete de reconocimiento. Este crecimiento lineal debe detenerse igualmente para evitar la congestión, comportándose igual que cuando se produce un timeout. Si en lugar de un timeout se produce una pérdida de paquete detectada por el recibimiento de tres ACK duplicados el comportamiento es menos drástico, por tanto reduce el tamaño de la ventana a la mitad y, si está implementado entra en recuperación rápida. **TCP Tahoe** no tiene recuperación rápida. Tras una pérdida limita a 1 MSS la ventana de congestión y entra en estado de arranque lento.

3. **Recuperación rápida:** Sólo implementada en **TCP Reno**, la ventana se incrementa en 1 MSS por cada ACK duplicado recibido correspondiente al segmento que falta y que ha causado la pérdida que le hizo entrar en este estado. Si llega un ACK para el segmento que falta, se entra de nuevo en el estado de prevención de la congestión. Si en este modo se produce otra pérdida se vuelve al estado de arranque lento.

4. Práctica 1

ąReiniciar el servicio correspondiente tras los cambios en sus archivos de configuración!

4.1. Información

4.1.1. Telnet

Para habilitar el servicio telnet hay que editar el archivo `/etc/xinetd.d/telnet` y cambiar la variable correspondiente. Para que el equipo solo sea accesible desde una IP específica añadimos la línea `only_from = <IP>` al archivo telnet.

Para registrar los intentos de log con telnet añadimos en el archivo telnet las líneas:

- Escoger dónde y cómo se guardarán los logs: `log_type = FILE <file>`.
- Quá información guardamos de los logs fallidos: `log_on_failure += HOST`.
- Qué información guardamos de los logs válidos: `log_on_success += HOST`.

4.1.2. FTP

Para que ftp no funcione en *standalone* sino que utilice el servidio xinetd, hay que editar en `/etc/vsftpd.conf` y cambiar `listen = no`. Para impedir el acceso anónimo `anonymous_enable = no`. Para permitir acceder a cuentas locales descomentamos `local_enable = yes`.
Para crear una lista de usuarios que se puedan conectar añadimos al fichero `/etc/vsftpd.conf`, `userlist_enable=YES` `userlist_file=/etc/vsftpd.user_list` `userlist_deny=NO` Esto último significa que la lista no es para denegar acceso. Los usuarios en cuestión deben estar escritos en el fichero `/etc/vsftpd.user_list`.
Para aceptar la subida de ficheros descomentamos en el archivo `/etc/vsftpd.conf` la línea `write_enable = YES`.

4.1.3. Apache

Para alterar la página de inicio cambiamos el archivo `/var/www/index.html`.
El puerto se cambia en los archivos `/etc/apache2/ports.conf` y `/etc/apache2/sites-enabled/000-default`.
Para crear unos usuarios restringidos: `htpasswd -c <ruta/fichero usuarios restringidos> <usuario>`

En la ruta `/var/www/<pagina restringida>.htaccess` añadimos las líneas:

```
1 AuthType Basic
2 AuthName "Restricted Content"
3 AuthUserFile "<fichero de usuarios restringidos>"
4 Require valid-user
```

Es necesario editar el archivo `/etc/apache2/apache2.config` y añadir las siguientes líneas:

```
1 <Directory "/var/www/<pagina restringida>">
2 Options Indexes FollowSymLinks
3 AllowOverride All
4 Require all granted
5 </Directory>
```