72.07 Protocolos de Comunicación

Resumen Final

Contenidos

Introduccion

Modelo OSI

1. Aplicación (HTTP, SMTP/POP, FTP)
2. Presentacion
3. Sesion
4. Transporte (TCP, UDP)
5. Red (IP, ARP, RARP, ICMP)
6. Enlace (PPP, Ethernet)
7. Fisica

La capa física establece el estándar de cómo transmitir los bits. Enlace se hace cargo de poder llegar al próximo salto en una misma red. Red se encarga de llegar a la otra punta → sabe por donde tiene que pasar, deben conocer cómo está todo conectado. Transporte es el que habla con el que está en la otra punta. Puede llegar a separar en unidades mas pequeñas y agregar la info (headeR) necesaria). Capa sesion → se encarga de la interacción con la Aplicación. Protocolo de Aplicación brinda soporte a programas sobre el formato que deben usar para intercambiar información.

Protocolos

Orientado a conexion	No orientado a conexión	Confiable	No confiable
1. Establecer conexión 2. Intercambiar información 3. Cerrar la conexión	1. Enviar información al destinatario	1. Confirma si la información fue recibida 2. Reenvía info de ser necesario 3. Informa al nivel superior si no se pudo enviar	1. No puede asegurar si el destinatario recibió o no la información enviada

- ➢ Una capa superior no requiere ofrecer el mismo servicio de conexión que una capa inferior. **Ejemplo:** HTTP es no orientado a conexión y utiliza TCP que es orientado a conexión.
- ➢ Si la capa inferior no ofrece un servicio confiable o con conexión, entonces lo puede ofrecer una capa superior.
  - **Ej**: TCP es confiable y IP no lo es.
- ➢ Ningún protocolo puede asegurar que los datos lleguen correctamente al destino, como mucho pueden confirmar si fueron o no recibidos
- ➢ Protocolo de enlace solo ofrece confiabilidad entre dos host ADJUNTOS que pertenecen al mismo segmento.

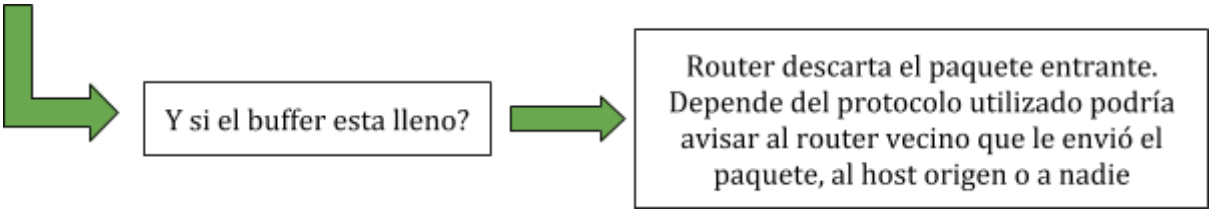
Diferencia entre transmisión broadcast y una comunicación broadcast?

Transmision Broadcast	Comunicacion Broadcast
Toda la información que se coloca en el medio puede ser recibida por todos los conectados a ese medio, pero tal vez estar dirigida a un único destinatario.	Dirigida a todos los posibles destinatarios, pero puede ser hecha por un medio unicast.
Ejemplo: En un supermercado se escucha por los parlantes “El dueño de automóvil con patente...”, es oído por todos pero todos (salvo el dueño del auto) lo ignoran.	1. Volantes dejados en casa 2. Llamadas automáticas por telefono de algun politico.

Interconexion de redes: router

- ➢ Decisiones en base a direcciones de red y no MAC
- ➢ Conectar distintas tecnologias (Ethernet, TR, WiFi)
- ➢ Separa una red en uno o más segmentos
- ➢ Backbone de Internet, ejecutando protocolo de IP
    - Establece rutas entre hosts
    - Regula el trafico

Obs! Una vez que el paquete llega a un router, el mismo tiene que ser analizado (al menos IP destino), tomar la decision de cual camino debe seguir y almacenarlo en un buffer. Una vez que el medio de transmisión está libre debe tomar el siguiente paquete de la cola y transmitirlo. Proceso genera demora que suma al tiempo de propagación en el medio.



Proceso

1. Una aplicación final genera datos, por ejemplo un front de correo electrónico recibe por parte del usuario desde qué cuenta enviar, a quien/és enviará el correo, el asunto, el texto, etc. El programa, en base a un *protocolo de aplicación*, prepara esos datos para que una aplicación servidor pueda recibirlos, y le pide a la capa de transporte que los haga llegar a la aplicación destino.
2. La *capa de transporte* encapsula los datos recibidos, y los divide –de ser necesario- en varios segmentos, agregando en cada uno un encabezado (desplazamiento, identificador del protocolo de aplicación y otros datos que veremos más adelante).
3. Para que esos segmentos lleguen al host destino le pide a un *protocolo de red* que lo haga, y ese protocolo de red agrega a su vez su encabezado (IP origen y destino, qué tipo de protocolo de transporte encapsula, etc.).
4. Pero antes de llegar a destino tiene que pasar por ejemplo un router, por lo que Red le pide a la *capa de enlace* que se lo haga llegar, y enlace agrega –entre otras cosas- los números de MAC origen y destino.
5. Finalmente la *capa física* transforma cada bit a enviar en señal (eléctrica, wifi, etc.)

De qué depende el tiempo que se demora el envío de datos entre dos hosts?

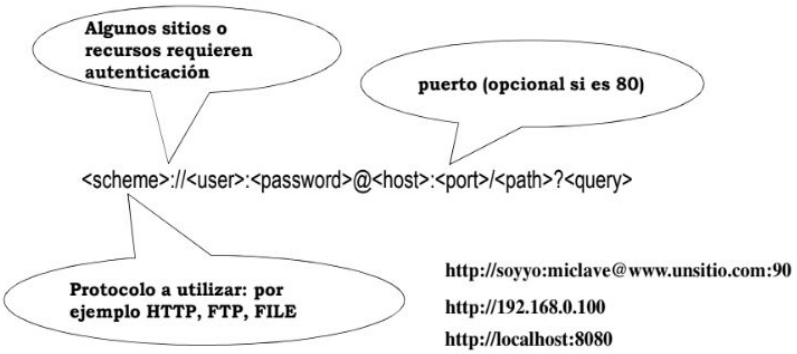
- ➢ **Ancho de banda + longitud del paquete**
- ➢ **Latencia** → suma de retardos temporales dentro de una red producidos por la demora en la propagación y transmisión de paquetes dentro de la red
- ➢ **Tiempo de procesamiento del router** → tiempo que tarda un paquete desde que es atendido hasta que es enviado por la interfaz correcta
- ➢ **Tiempo de encolamiento** → depende del tamaño del buffer dentro del router

HTTP

- Protocolo de aplicación NO ORIENTADO a conexión.
- Hasta HTTP 2, protocolo de texto. HTTP 2 es binario.
- HTTP 1.1
  - Cabeceras en las peticiones
  - GET, POST, HEAD, PUT, DELETE, TRACE, OPTIONS
  - Negociación de contenido
  - Soporte de caché
  - Conexiones persistentes

Recursos

- Bloque de información identificado por su URI
- URL: Uniform Resource Locator
  - Identifica un recurso por su ubicación



- URN: Uniform Resource Name
  - Identifica un recurso por su nombre
  - No implica que le recurso exista o como acceder a el

```
urn:isbn:0132856204
urn:isan:0000-0002-3C36-0000-Y-0000-0000-9
urn:uuid:6e8bc430-9c3a-11d9-9669-0800200c9a66
urn:www.apache.org:
```

- URI: Pueden ser usadas para acceder a recursos, ya sea por medio de URL o URN

```
<a href="/img/logo.png">
<a href="urn:isbn:0453457513">
```

Ejemplo en HTML

Mensajes

- Request message (cliente → servidor)
  - GET: Solicita un recurso al servidor
    - Pueden ser "cacheados", browser los mantiene en el historial, tienen longitud acotada, pueden ser bookmarked, solo para pedir datos.
  - HEAD: Solicita solo los headers del recurso
  - POST: Envía información al servidor para ser procesada

        - Nunca son “cacheados”, no se mantienen en el historial, no pueden ser “bookmarked”, no tienen restricción de longitud de datos.
    - PUT: Envía un recurso al servidor
    - TRACE: Analiza el recorrido de la solicitud
    - OPTIONS: Consulta los métodos disponibles en el servidor
    - DELETE: Elimina un recurso del servidor.
- ➢ Response message (servidor → cliente)

+ Start Line: Versión, código de respuesta y mensaje.

1XX	Información – El proceso continúa
2XX	Éxito - Acción recibida, comprendida y aceptada
3XX	Redirección – Se requiere nueva acción
4XX	Error en cliente – Sintaxis y/o Semántica inválida
5XX	Error en servidor – Falló pedido correcto

Obs! HTTP utiliza MIME (protocolo que pertenece a la capa de presentación del modelo OSI) para describir contenido multimedia.

Headers

- ➢ Tipos → Generales, de solicitudes, de respuestas, de contenido

Generales	De solicitudes	De respuestas	De contenido
Cache-control: Directivas para cache **Connection**: Se definen opciones de conexión **Date**: Fecha de creación del mensaje **Transfer-Encoding**: Indica el encoding de transferencia **Via**: Muestra la lista de intermediarios por los que pasó el mensaje.	**Accept** : Contenido aceptado por el cliente **Accept-Charset** **Accept-Encoding**: gzip, compress **Expect**: Comportamiento esperado del server frente al request. **From**: Email del usuario de la aplicación que generó el request. **Host**: Servidor y puerto destino del request. **If-Modified-Since**: Condiciona al request a la fecha indicada. **Referer**: URL del documento que generó el request **User-Agent**: Aplicación que generó el request **Upgrade**: solicita que use otro protocolo **Range**: solicita un rango (en bytes) del recurso	**Age**: Estimación en segundos del tiempo que fue generada la respuesta en el server. **Connection**: close, keep-alive **Location**: URI a redireccionar **Retry-After**: Tiempo de delay para reintento **Server**: Descripción del software del server. **Authorization**: indica que el recurso necesita autorización	**Allow**: Métodos aplicables al recurso. **Content-Encoding** **Content-Length** **Content-Location** **Content-MD5** **Content-Type** **Expires:** Fecha de expiración del recurso. **Last-Modified**: Fecha de modificación del recurso

Conexión no persistente → un request con su response y se cierra la conexión
Conexión persistente → permite más de un request antes de cerrar
Proxy server → Actúa como intermediario entre una aplicación cliente y un web server. Permite controlar el acceso a determinados sitios o páginas y almacenar en caché recientes consultas.

¿De qué forma puede un Proxy HTTP saber que para el recurso que le solicita un cliente aún es válida la copia que tiene en caché?

El recurso solicitado puede tener una o ambas de los siguientes headers:

- ➢ Max-age: Cuánto tiempo va a vivir el recurso en la caché

- Age me dice hace cuanto tiempo esta → mientras age < max-age este recurso seguirá "vivo" en la caché

➢ Etag: indica la versión del recurso → Cuando solicita el recurso, se le manda un tag If-Modified-Since (cuando se guardó en la caché) y otro tag If-non-match: etag. Si el recurso fue modificado (cambió su etag) desde que se guardó en la caché, pisa la version vieja en el caché.

Cookies

➢ Pequeño texto de info enviada por el servidor y almacenada por el browser
➢ Usadas para mantener un estado entre el cliente y el servidor
➢ Simula una "sesión" abierta
- Cliente manda usuario/contraseña
- Servidor le crea un cookie y se la devuelve al cliente
- Cliente manda este cookie cuando hace un GET
- Servidor valida dicho cookie

Third Party Cookie:

A third-party cookie is one that is placed on a user's hard disk by a Web site from a domain other than the one a user is visiting. As with standard cookies, third-party cookies are placed so that a site can remember something about you at a later time. Both are typically used to store surfing and personalization preferences and tracking information. Third-party cookies, however, are often set by advertising networks that a site may subscribe to in the hopes of driving up sales or page hits. They are often blocked and deleted through browser settings and security settings such as same origin policy.

Sockets

➢ Host se identifica por IP, proceso se identifica por puerto
➢ Socket es identificado por un handle y la info relacionada es almacenada en la tabla de fd
➢ Cliente debe especificar IP y puerto del servidor
- Inet_addr: notación con puntos a decimal
- Gethostbyname: dado un string que contiene el FQDN de un host retorna una estructura que contiene el IP del host en forma decimal
- Getaddrinfo: más completa que la anterior

Algoritmo para servidor iterativo no orientado a conexión	**Algoritmo para servidor iterativo orientado a conexión**
1. Crear un socket y ligarlo (*bind*) a un puerto 2. Leer un *request* de un cliente 3. Enviar una respuesta 4. Volver al punto 2	1. Crear un socket y ligarlo (*bind*) a un puerto 2. Aceptar un pedido de conexión a través del socket 3. Obtener un nuevo socket para la conexión 4. Leer un *request* del cliente 5. Enviar una respuesta 6. Si el cliente no finalizó, volver al punto 4 7. Cerrar el socket creado en punto 3 8. Volver al punto 2

Algoritmo para servidor concurrente orientado a conexión

Master (Proceso Padre)
1. Crear un socket y ligarlo (*bind*) a un puerto
2. Aceptar un pedido de conexión a través del socket
3. Crear un proceso hijo para atender la conexión
4. Volver al punto 2

Slave (Proceso Hijo)
1. Atender la conexión pasada por el master
2. Interactuar con el cliente
3. Cerrar la conexión y finalizar el proceso

Obs! 1 solo socket pasivo, N sockets activos

Resolución de nombres

- ➢ Dominio: colección de compus que pueden ser accedidas usando un nombre en común
- ➢ Nombre de dominio: referencia al nombre de múltiples hosts que son referenciados colectivamente (ej: itba.edu.ar)
- ➢ FQDN = hostname + dominio al cual pertenece

Obtener la IP dado un FQDN?

1. Archivo /etc/hosts
   a. Línea = número de IP y el/los nombre/s asociados a dicho IP
   b. Primero se consulta esto antes del DNS
      i. /etc/host.conf para cambiar el orden
         order hosts, bind → order bind, hosts

2. DNS resolver → se fija en la configuración de /etc/resolv.conf
   a. nameserver *direcciones*
      i. Números IP de los servidores DNS que consultara el resolver. Si no hay ninguno especificado, utilizara al propio host como servidor de nombres.
   b. domain *nombre*
      i. Dominio local por defecto. Resolver agrega este nombre a todo hostname que no contenga un punto antes de resolverlo
   c. search *dominios*
      i. Similar al anterior pero pueden ser varios dominios (no se usan ambas a la vez)
   d. sortlist *red[/mascara]*
      i. En caso de recibir múltiples direcciones IP para un nombre, reordena las direc. en base a las redes listas en esta opción.
   e. **options *opción*** → seteos opcionales

DNS

- PROTOCOLO NO ORIENTADO A CONEXIÓN Y NO CONFIABLE
- Base de datos distribuida implementada en una jerarquía de servidores de nombre.
- Protocolo de aplicación que permite consultar dicha base
- **Root name servers**
  - 13 en total
  - Usados cuando el servidor de nombres local no puede resolver el nombre
  - Contacta al name server autorizado si no conoce la respuesta, almacena, retorna respuesta al servidor de nombre local.
- Preguntas y respuestas utilizan UDP
  - Si la respuesta no entra en un datagrama, la aplicación se debe encargar en partir la información y enviar tantos datagramas como sean necesarios.
- Una vez que el servidor de nombres aprende el mapeo, lo almacena en caché y expiran tras un tiempo (TTL)
- Mismo servidor DNS puede ser autorizado para más de un sitio
- Distintos nombres pueden estar asociados a la misma IP

Niveles de configuración

- Resolver only system
- Caching-only servers
  - Aprende respuestas acerca de nombres de dominio de otro servidor y lo almacena para futuras consultas
  - No es considerado server autorizado → info es de segunda mano
- Master servers
  - Fuente autorizada para toda la información de una zona específica
  - Lee info sobre dominio desde un archivo construido por el administrador
  - Server autorizado → puede responder con autoridad preg sobre ese dominio
- Slave servers
  - Transfiere info completa de una zona desde un master server y la almacena en un archivo en el disco local.
  - Mantiene info completa + actualizada de una zona → puede responder preguntas en forma autorizada sobre la zona

Registros DNS

type	name	value
A	nombre del host	IP
NS	dominio	nombre del host autorizado a resolver nombres del dominio
CNAME	alias para el nombre canónico	nombre canónico
MX	dominio	nombre del servidor de mails para el dominio
AAAA	nombre del host	IPv6
SOA	Start of Authority	Inicio de información autorizada
SRV	Localizador de servicios	Para no crear registros tipo MX
RP	Persona responsable	Normalmente el mail del responsable
...		

Testear si conf es correcta → `host, nslookup, dig`
Obtener info sobre el responsable de un dominio → `whois`

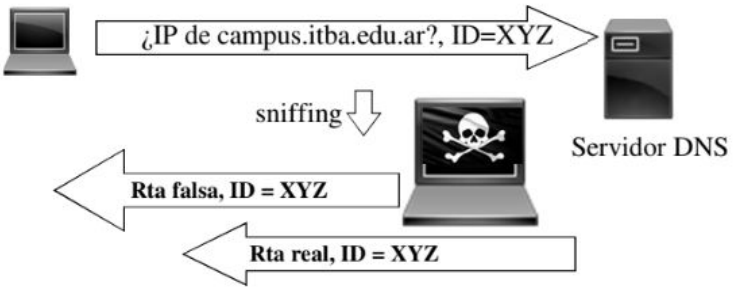
Split-Horizon: un nombre se resuelve con distintas IP dependiendo del origen de la consulta

DDNS

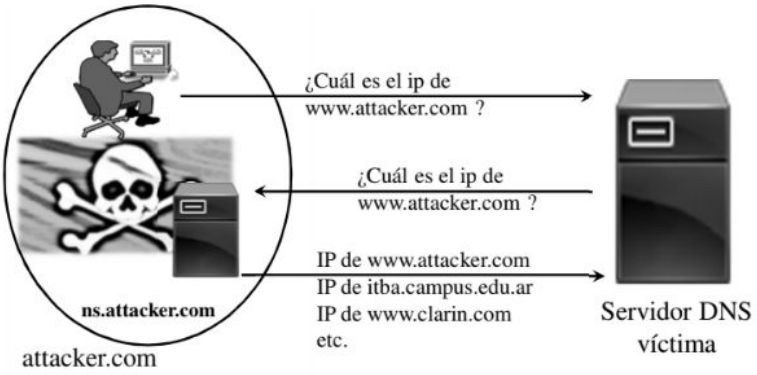
- Permite actualizar en forma dinámica un servidor DNS
- Cuando se negocia una IP con el DHCP, luego del DHCP ACK se actualiza el registro del servidor DNS.

DNS Spoofing

- Lograr que un registro DNS apunte a un IP que no sea el que debería apuntar
- Metodos:
  - Enviar respuesta sin recibir una pregunta
    - Solución: Se verifica que la respuesta se corresponda con alguna pregunta. DNS establece que los paquetes de pregunta y respuesta tengan un mismo ID que los identifique
  - DNS sniffer → Esperar una pregunta al servidor DNS y enviarle al cliente una respuesta falsa antes que le llegue la real.



  - DNS cache poisoning → enviar varias respuestas, no solo la pedida



Telnet y correo

- TELNET → protocolo de comunicación → interconectar dispositivos de terminal y procesos orientados a terminales.
  - No encripta
  - Permite operar en una computadora en forma remota
  - Basado en una Network Virtual Terminal (NVT)
- `telnet` → programa que soporta protocolo TELNET sobre TCP

NVT

- Representación intermedia de una terminal genérica
- Provee un lenguaje estándar para control de terminales
- Puntas negocian un conjunto de opciones
- Dispositivo bidireccional orientado a carácter, con un teclado y una impresora

SMTP y protocolos de entrega final a usuario

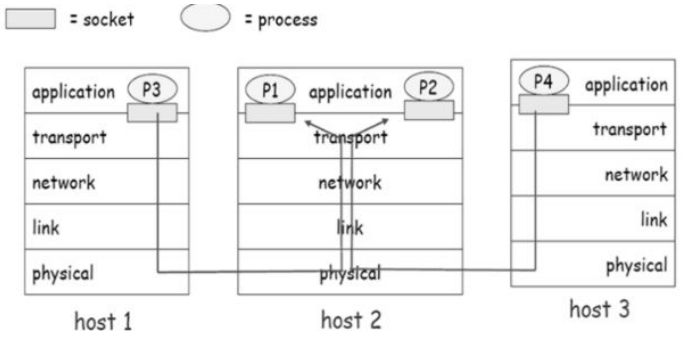
- SMTP
  - ORIENTADO A CONEXIÓN Y CONFIABLE
  - Protocolo por el cual se transmiten los mails por Internet
  - Puertos estándares de SMTP → 25, 265, 587
  - TCP para transferir mensajes del cliente al servidor
  - Fases: handshaking, transferencia de mensaje, cierre
  - Mensajes son ASCII-7 bits
    - No puede transmitir objetos binarios ni texto con caracteres nacionales → Para esto se utiliza MIME
- POP3
  - Obtener mensajes del servidor y almacenarlos en el host local (puerto 110)
  - ORIENTADO A CONEXIÓN Y CONFIABLE
- IMAP (puerto 143)
  - Usuario consulte correo desde varios hosts
    - Almacenamiento central accesible desde cualquier host
  - Cliente IMAP no copia corre en el host local y distingue si los mensajes fueron ya leídos o no.
  - Permite crear carpetas en el servidor
  - Realiza backup de los mails

Protocolos de Transporte

- UDP, TCP, SCTP
  - NO ofrecen mínimo de demora o latencia ni mínimo de ancho de banda
- Provee la comunicación lógica entre dos procesos que corren en distintos hosts
- Ejecutan en las puntas finales
  - El que envía → separa mensajes en segmentos y los entrega al nivel red
  - El que recibe → reensambla los segmentos en mensajes y los pasa a nivel app

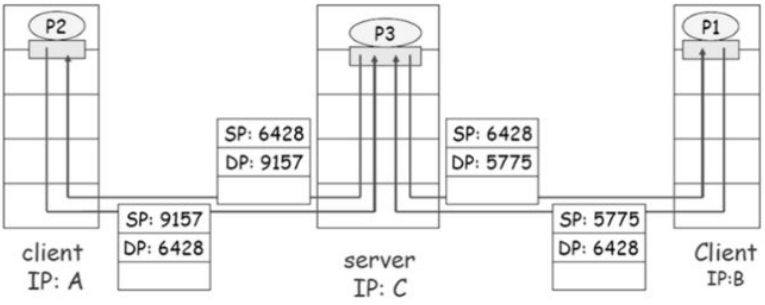
Multiplexacion / Demultiplexacion

Demultiplexación



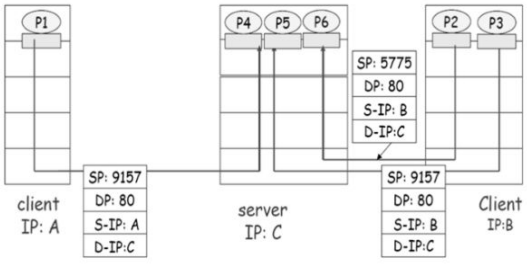
- Host recibe datagramas IP. Cada datagrama contiene:
  - IP origen e IP destino, puerto de origen y puerto destino
  - Segmento del protocolo de transporte
- Host utiliza IP + puerto para dirigir el segmento al socket apropiado

Demultiplexación sin conexión



- Crear un socket con un número de puerto
- Socket identifica por el par <IP destino, puerto destino>
- Host recibe segmento UDP → dirige segmento al socket que atiende <IP, puerto>
- Datagramas con distinto IP origen son atendidos por el mismo socket

Demultiplexación orientado a conexión



- Cada socke identificado por: IP origen, puerto origen, IP destino, puerto destino
- Host utiliza estos 4 valores para dirigir el segmento al proceso que corresponde
- Conexiones simultáneas, cada una con su socket

Transferencia de datos confiables

➢ Si el protocolo de red es confiable
- No se alteran bits
- No se pierden paquetes y los mismos llegan en orden

Sender	Receiver
1. Esperar llamada de aplicación 2. Leer datos a enviar 3. Armar paquete 4. Pedir a capa de red que lo envíe	1. Esperar llamada de nivel de red 2. Leer datos recibidos 3. Extraer datos del paquete 4. Enviar datos a aplicación

➢ Recuperación ante errores?
- ACK o NAK → Sender debe retransmitir ante NAK o falta de ACK
- Número de secuencia en cada paquete para evitar paquetes duplicados

Pipelining → process of accumulating instruction from processor through a pipeline. Allows storing and executing instructions in an orderly process (instructions overlapped during execution). ACK con hasta donde se recibió.

UDP	TCP
- No orientado a conexión - No confiable - No reensambla los mensajes entrantes - No proporciona control de flujo - Sin acuse de recibo	Ofrece circuito virtual entre aplicaciones de usuario final. - Orientado a conexión y confiable - Divide los mensajes salientes en segmentos y luego los reensambla - Vuelve a enviar lo que no se recibió - Control de flujo - Control de congestion

Obs! En TCP, no se envía ACK por segmento, sino por el próximo byte a recibir.
Todo los pedidos de conexión TCP son enviados al mismo socket pasivo, una vez aceptada la conexión (pasado el three handshake) se crea un socket activo para atender esa conexión, la clave del socket activo es <IP origen, puerto origen, IP destino, puerto destino>

Control de congestion

➢ Hace referencia al tráfico IP
➢ Muchas fuentes enviando muchos datos y a una velocidad que la capa *de red* no puede manejar. IP no se hace cargo.
➢ DISTINTO a control de flujo (receptor controla al emisor de modo que el emisor no sobrecargue el buffer del receptor).

- ➢ Manifiesta en pérdida de paquetes (descartados por el router) y demoras
- ➢ **IDEAL** → TCP no emita segmentos que serán descartados para evitar la retransmisión
  - AIMD: additive increase multiplicative decrease

Problema: Mandando mucho datos que no estoy leyendo → se llena el buffer
Solución: WINDOW → Cuantos bytes tengo libre para almacenar en el buffer. Si w = 0 → no me mandes nada más!

Emisor incrementa tamaño de ventana hasta que detecta pérdida de paquete (timeout)

- ➢ *additive increase*: incrementa tamaño de ventana en relación al MSS (maximum segment size) por cada RTT (round trip time) hasta que detecta pérdida
- ➢ *multiplicative decrease*: al detectar pérdida reduce tamaño de ventana a la mitad

NAPT

- ➢ En el router, se agrega un firewall que traduce los números de IP y puertos
  - Se puede tener muchas computadoras con 1 solo IP publico
- ➢ Router tiene una tabla del IP privado origen y el destino para saber donde mandar los paquetes.
- ➢ Protege red ocultando los hosts internos al resto de las redes

SCTP

- ➢ Orientado a mensaje → por cada write hago un read
- ➢ Resuelve head of line blocking (no puedo usar el recurso hasta que me lleguen todos los paquetes → si se pierde el primero no puedo utilizar los otros)
- ➢ Multihoming: Si hay mas de una interfaz de red SCTP, puede usar ambas. Mensaje heartbeat para verificar que la que estoy usando esta activa.
- ➢ Inicio protegido
  - Pedido de conexión → servidor crea un cookie y se la envía (sin generar recursos todavía) → cuando llegan los primeros datos, la cookie llega con ellos y ahí se asigna el espacio.
  - TCP genera el espacio de una y no envía datos en el segundo ACK :(

Característica	SCTP	TCP	UDP
Estado almacenado en los hosts	SI	SI	NO
Transferencia de datos confiable	SI	SI	NO
Control de congestión	SI	SI	NO
Delimitación de límites de mensajes	SI	NO	SI
Fragmentación e integración de datos	SI	SI	NO
Multiplexación	SI	SI	NO
Soporte de multi-homing	SI	NO	NO
Soporte de multi-streaming	SI	NO	NO
Envío de datos desordenado	SI	NO	SI
Cookie de seguridad para evitar ataques de inundación de SYN	SI	NO	NO
Mensaje heartbeat	SI	NO	NO

Establecer conexión	SI	SI	NO
Garantia de mínimo delay	NO	NO	NO
Checksum de segmentos	SI	SI	SI
Control de flujo	SI	SI	NO
Envio de NAK	NO	NO	NO
Almacenamiento de segmentos hasta recibir ACK	SI	SI	NO

SSL

- ➢ Capa intermedia entre protocolos de aplicación y de transporte (TCP)
- ➢ Permite a un cliente autenticarse con un servidor y encriptar la conversación entre ellos
- ➢ CLAVES
  - Simetrica: misma clave para encriptar y desencriptar
  - Asimetricas
    - Encriptar = llave pública → Desencriptar = clave privada
    - Encriptar = llave privada → Desencriptar = clave pública

TLS

- ➢ Conexión segura para HTTPS (443), IMAPs (993), POPs (995). Usa SSL
- ➢ Conexión segura por protocolo: comienza con un "hello" inseguro y luego cambia a una conexión segura.
- ➢ TLS y SSL = mismo nivel de encriptación, difieren en cómo se inicia la conexión segura

Programación con sockets

- ➢ Signals
  - Mecanismo para notificar un proceso que ocurre un evento
  - Se puede ignorar la señal, terminar abruptamente, interrumpir su ejecución normal y ejecutar una “rutina de manejo de señal”, bloquear la señal
  - sa_mask → bloquea las señales entrantes mientras se esté ejecutando un handler
  - Puede interrumpir un socket call (recv, connect, etc).
    - El socket call return -1
    - errno = EINTR
- ➢ Blocking
  - Llamadas a E/S pueden bloquearse por:
    - recv(), recvfrom(): no hay datos listos
    - send(), sendto(): no hay espacio en el buffer de TX
    - connect(), accept(): hasta conectar

Sockets no bloqueantes

- ➢ Si función se necesita bloquear, asigna EWOULDBLOCK a errno
- ➢ UDP send y sendto no bloquean porque no hay buffer de salida

- ➢ TCP asigna EINPROGRESS a errno en connect()
- ➢ Programa realiza otras tareas hasta tener que manejar la señal SIGIO
- ➢ select() recibe una lista de descriptores y se bloquea hasta que alguno esté listo para I/O
    - Para escuchar en varios puertos

Capa de Red

- ➢ Direcciones → direccionamiento jerárquico → ruta en forma eficiente
- ➢ Traslada segmentos de un host a otro
- ➢ En cada host se ejecutan los protocolos de nivel de red
- ➢ Protocolo IP ofrece a su capa superior un servicio sin conexión con reconocimiento
    - Fragmenta paquetes → cada uno múltiplo de 8
- ➢ Cada host en el camino → router
    - Examina los encabezados IP y decide el camino a seguir
- ➢ Funciones en capa de red
    - Forwarding: mover paquetes del input al output del router. Decisiones en base a la tabla de ruteo.
    - Routing: determinar la ruta que deben tomar los paquetes para llegar a su destino. Como se crea la tabla de ruteo.
    - Establece conexión
        - Antes de intercambiar datagramas, se establece un circuito virtual entre dos hosts
- ➢ Servicios adicionales
    - Conmutación de paquetes (datagramas)
        - Delivery garantizado
        - Delay mínimo garantizado
        - "Datagram networks"
            - No hay llamada previa
            - Routers: no hay estado sobre las puntas
            - Paquetes se forwardear en base a dirección de destino
    - Flujo de datagramas (circuito virtual)
        - Entrega en orden
        - Bandwidth mínimo
    - Ambos → integridad de datos, encriptacion

Circuitos virtuales

- ➢ Camino fijo por el que viajan paquetes → fuerza a los paquetes a seguir siempre el mismo camino.
- ➢ Establecer la conexión antes de enviar datos
- ➢ Cada paquete → identificador de CV
    - Si recibe de distintas interfaces, un mismo número CV no pasa nada. El problema es cuando tenes el mismo número para la misma interfaz porque no podes distinguir por donde mandar los paquetes.
- ➢ Cada router en el camino mantiene el estado de cada CV
- ➢ **Fases**

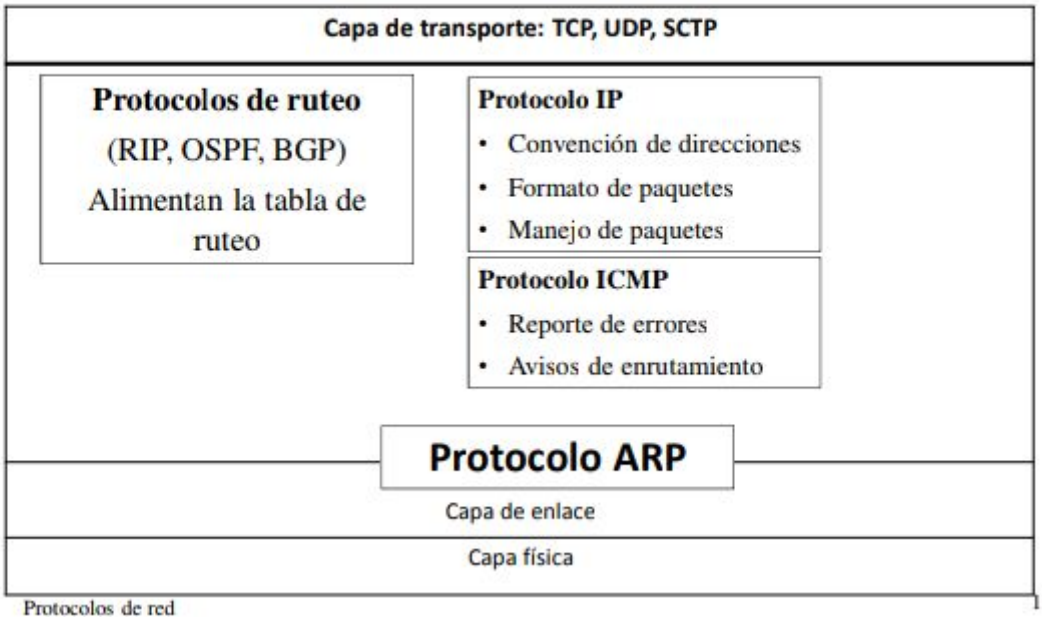
- Establecer la conexión → añadir una entrada a la tabla de ruteo, reservar recursos
- Transferencia de paquetes
- Cierre de la conexión

CV vs. Conmutación de paquetes

	Datagrama	Circuito virtual
Establecer conexión	---	Requerido
Direccionamiento	Id. De host origen y destino	Número de CV
Info de estado	No	Tabla de subdred con números de CV en cada router
Enrutamiento	Cada paquete una ruta independiente	Todos los paquetes del CV una misma ruta
¿Si falla un nodo?	Se pierden algunos paquetes	Todos los CV que pasan por el nodo finalizan
Control de congestión	Complejo	Simple
Complejidad	En la capa de transporte	En la capa de red

Protocolos de red

Capa de red en Internet



Protocolos de red 13

Direccionamiento IPv4

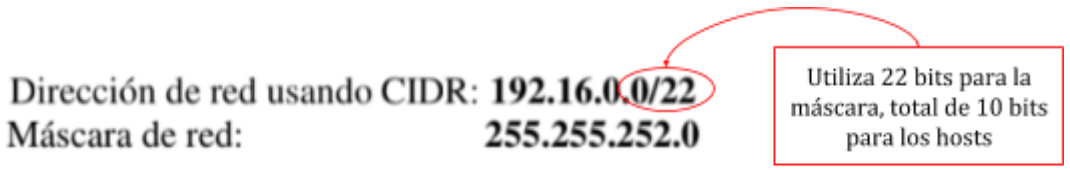
- 32 bits
- Cada nro IP consta de:
  - Network ID → Compartido por los host en una misma red
  - Host ID

- Reservadas
  - *Loopback* → comienzan con 127
  - *Identificar la red* → host id con todos 0
  - *Broadcast* → host id con todos 1
  - Redes privadas
    - 10.x.x.x
    - 172.16.0.0 a 172.31.255.255
    - 192.168.x.x

Clase IP	Rango	Máscara	Redes	Hosts en cada red
A	1.0.0.0 a 126.0.0.0	255.0.0.0	126	16,777,214
B	128.0.0.0 a 191.255.0.0	255.255.0.0	16,384	65,534
C	192.0.0.0 a 223.255.255.0	255.255.255.0	2,097,151	254

Dada una dirección IP se puede determinar la red a la cual pertenece y así "*rutear*" el paquete.

- Paquete IPv4 (datagrama)
  - Vers, Hlen, Longitud total, flags, identification, ToS
  - Desplazamiento de fragmento, header checksum, protocolo
  - TLL (ver apuntes cuaderno)
  - Dirección origen
  - Dirección destino
  - Opciones, relleno, datos
- Asignación de direcciones IP
  - Direccionamiento dinámico
    - RARP → conoce su MAC y la IP y MAC destino pero no su IP. Envía paquete RARP a broadcast a nivel red (cual es mi IP?)
    - BOOTP → cuando bootea pregunta por su IP mediante un mensaje broadcast que contiene su MAC
    - DHCP



DHCP

- Protocolo de transporte → UDP
  - TCP no sirve porque está orientado a conexión y necesita una IP
- Discover → offer → request → ACK
  - Offer pues hay más de un servidor DHCP y otro ya le pudo haber respondido
  - Recien tengo mi IP a partir del ACK
- Se debe ir renovando la IP → trata de volver al DHCP que ya pidió alguna vez

NAT (SNAT)

- Medida para aliviar la necesidad de IPs públicas
- DNAT, SNAT → funciones de un firewall, no un router
- Modifica el IP/puerto de origen
  - También tiene que cambiar el checksum!
- Port forwarding

DNAT

- Cambia la dirección destino
- Redirigir conexiones de afuera

ICMP

- IP no maneja errores ni retransmisiones. Para avisar de un error → mensaje ICMP
- Ping, traceroute (alcanza host destino informando de cada router por el que pasó)

IPv6

- 128 bits
- Número de MAC como parte de la IP
- Seguridad y multicasting
- Encabezado de tamaño fijo → eficiente su procesamiento por routers
- No permite fragmentación
- Formato
  - Priority: prioridad del paquete dentro del flujo
  - Flow label: identifica paquetes dentro de un mismo "flujo"

Tunneling → Consiste en encapsular datos de un protocolo en los datos de otro protocolo de igual o mayor nivel

MTU → size of the largest protocol data unit that can be communicated in a single network

Protocolos de Routing

- Ambas puntas deben utilizar el mismo protocolo de red
- Routing permite que tráfico de la red pueda ser enviada hacia cualquier otra red.
- Tablas de ruteo se pueden actualizar en base a paquetes **ICMP Redirect**
  - An ICMP redirect is an error message sent by a router to the sender of an IP packet . Redirects are used when a router believes a packet is being routed sub optimally and it would like to inform the sending host that it should forward subsequent packets to that same destination through a different gateway.

Formato de un paquete ICMP Redirect

Type=5	Code	Checksum
Gateway Internet address (*192.168.10.1*)		
Internet header + 64 bits Data		

CODE
- 0 = Redirect datagrams for the Network.
- 1 = Redirect datagrams for the Host.
- 2 = Redirect datagrams for the ToS and Network.
- 3 = Redirect datagrams for the ToS and Host.

Tipos de Routing

- **Routing mínimo:** una red aislada de otras redes TCP/IP
- **Routing estatico:** la tabla de ruteo es construída por un administrador en forma estática.
- **Routing dinamico**: la tabla de ruteo es construida en base a información intercambiada por protocolos de routing
  - Categorias:
    - *Routing interno* (IGP): diseñados para una red controlada por una sola organización. RIP y OSPF más usados.
    - *Routing externo* (EGP): diseñados para intercambiar información entre redes controladas por distintas organizaciones (sistemas autónomos). Se usa BGP.
  - Algoritmos adaptivos
    - Dijkstra → camino mas corto
    - Flooding
    - Papa caliente
    - Distance Vector
    - Link state

Routing dinamico → Distance Vector

- Cada router recibe de sus vecinos una tabla de enrutamiento. En base a ella actualiza su tabla y la envía a sus vecinos.
- Cada nodo tiene una tabla donde sus columnas son:
  - Destination → a donde quiere ir
  - Costo → cuánto le "cuesta" la ruta elegida
  - Próximo salto → a qué nodo tengo que ir primero (va a ser un vecino)
- **Problemas** = contar hasta infinito. **Posibles soluciones:**
  - Split horizon: no enviarle a mi vecino lo que mi vecino me está enviando.
  - Ruta envenenada: lo que mi vecino me envía se lo envío con salto ∞, para que no actualice.
  - Enviar rápido las malas noticias: si un router detecta un problema en el vínculo, enviar en forma inmediata esa entrada "envenenada".

Action at a router

- On receiving a message from a neighbor v:
  - Update cost (estimate) to destinations, change nest hop accordingly
  - For each y (destination in the routing table of the received message)
    - $D_x(y) = min\{current\ estimate,\ c(x,y) + D_v(y)\}$
      - Dv(y) is the cost of the neighbor to get to y
      - c(x,y) is the cost of x (where I am) to y (my neighbor)
  - Estimated costs finally converges to optimal cost after series of message exchanges.
- *When to send a routing message to neighbors?*
  - Triggered update: Sent whenever the DV changes

        - Link/Node failure or ost increase
            - Know this happened because a periodic update wasn't received or can actively probe
    - Periodic update: sent even when no change in routing table
        - To tell others that "I am still alive"
        - To update others' DV in case some route becomes invalid
        - Order: few sec to few min

Routing dinamico → RIP

- Pertenece a los algoritmos distance vector
- Selecciona una ruta en base a la cantidad de saltos a usar para llegar a destino.
- Adecuado para redes pequeñas , simple de configurar
- El demonio en Linux es routed.
    - Cuando se inicia envía un pedido de actualizaciones de ruteo.
    - Cada router envía en forma periódica paquetes con información basada en su tabla de ruteo.

RIP v1

- Paquetes son broadcast
- Notación classful → Direcciones IP de red con máscara natural
- Paquetes no tienen autenticación
- **Ventajas**
    - Simple de configurar
    - No tiene complejidad
    - Usa menos CPU
- **Desventajas**
    - Solo se basa en la cantidad de saltos → si hay una ruta con más ancho de banda pero mas saltos, no la va a elegir.
    - Al mandar los paquetes en broadcast a toda la red, crea mucho trafico
        - Esto también utiliza mucho ancho de banda
    - Suporta como mucho 15 saltos
    - Toma mucho tiempo en encontrar una ruta alternativa cuando falla un enlace
    - No es muy reliable

RIP v2

- Informa direcciones CIDR
- Updates en forma multicast y con autenticación (opcional)

Link state

- Cada router recibe del resto de los routers información sobre sus vecinos. En base a ella arma un grafo de la red y calcula el o los caminos más cortos.

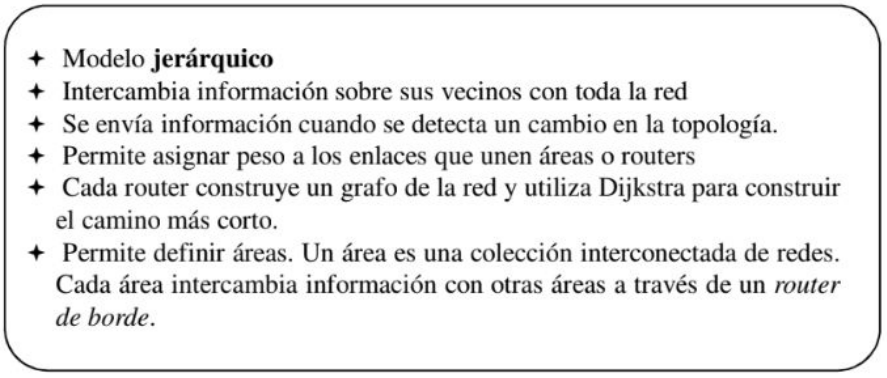
Cada router debe hacer lo siguiente:

1. Descubrir sus vecinos
2. Medir la distancia (demora o costo) a cada vecino
3. Construir un paquete con lo que aprendió
4. Enviar el paquete a todos los routers de la red
5. Calcular el camino más corto a cada router

Distance Vector	Link state
Visualiza la topología de red en base a la visión de sus vecinos	Visualiza topología completa de la red
Suma distance de router a router	Calcula la ruta más corta hacia otros routers
Actualizaciones frecuentes y en forma periódica	Envía estado de enlace a todos los routers
Envía copia de la tabla de ruteo a los vecinos	Envía estado de enlace a todos los routers
Limita el diámetro de la red	No tiene límite para el diámetro de la red

OSPF

- Open shortest path first



- Modelo **jerárquico**
- Intercambia información sobre sus vecinos con toda la red
- Se envía información cuando se detecta un cambio en la topología.
- Permite asignar peso a los enlaces que unen áreas o routers
- Cada router construye un grafo de la red y utiliza Dijkstra para construir el camino más corto.
- Permite definir áreas. Un área es una colección interconectada de redes. Cada área intercambia información con otras áreas a través de un *router de borde*.

- Definición de áreas
  - Cada área está conectada con el área cero a través de routers de borde (ABR)
  - Routers de borde sumarizan las redes de una área

Capa de Enlace

- TRAMA: Paquete a nivel enlace → encapsula datagramas
- Transferencia confiable de datagramas entre dos nodos adyacentes
- Servicios
  - Control de flujo
  - Detección y corrección de errores

Redes broadcast

Se necesita coordinar cuando mandar las cosas para que no sucedan colisiones

- Particionamiento del canal
  - Nodo → tamaño fijo de slot por turno
  - Slot no usados se desperdician
- Protocolo por turnos

- Polling → nodo master invita a un nodo a transmitir, cuando termina le dice a otro. Contras: latencia, polling overhead
- Token Ring
  - Nodos conectados en forma de anillo
  - Datos circulan en un único sentido
  - Cada nodo recibe token que lo habilita a transmitir

➢ Protocolo de acceso aleatorio

➢ CSMA

- Sensar el canal antes de transmitir, si está ocupado → esperar
  - Si está libre → transmitir trama completa
- Sensar medio para verificar si hubo colisión

OBS! Mensajes broadcast no pasan de un segmento a otro

Ethernet

➢ Topologia de bus

- Todos los host en la misma línea, conectados a un hub
- Cada host se puede conectar/desconectar en cualquier momento
- Gran posibilidad de colisiones
- Red crece → divide en segmentos

➢ Topologia de estrella

- Cada compu conectada a un switch
- Dirige los datos por el puerto al que está conectado el host, pudiendo encolar distintas tramas
- Se basa en el MAC address

ARP

➢ Para poder enviar tramas de un host a otro, se debe conocer el número MAC del host destino. ¿Cómo sabe el host, dado un número IP, a qué MAC address corresponde?

- arp → muestra tabla que mantiene cada host con la relación entre números IP y números MAC.

➢ SOLO se consulta para las IPs de la red LOCAL. PAra IPs de otras redes se necesita el MAC del Gateway

➢ Utiliza en 4 formas:

- ARP: un host con dirección IP necesita conocer la dirección MAC de otro host del cual conoce la dirección IP.
- RARP: un host sin almacenamiento (diskless workstation) conoce su dirección MAC pero "olvidó" su dirección IP.
- InARP: un host conoce la dirección MAC de otro host pero no su IP
- Proxy ARP: host responde a un request para otro host

➢ ARP spoofing

- ARP → sin autenticación → cualquiera puede responder un ARP request
  - "Main in the middle"
  - "Denial of service"
  - Session hijacking

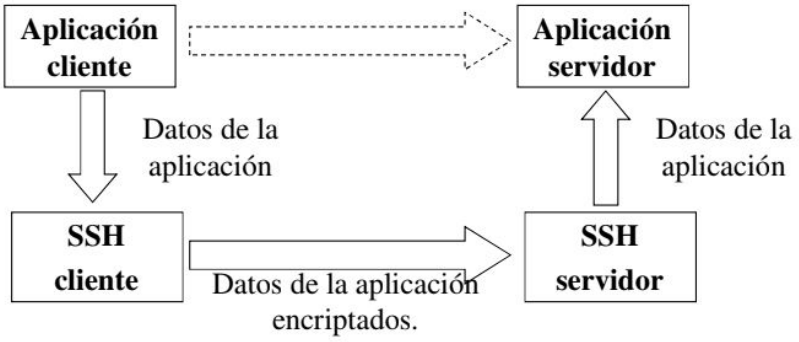
Nota de transmisión: Si el tiempo de propagación es menor al tiempo en que tarda en llegar el último bit al extremo del cable, entonces puede haber una colisión pero no ser detectada por los hosts que están transmitiendo. Para asegurarse que se detectan las colisiones, Ethernet establece tanto una longitud máxima de cable como un tamaño mínimo de trama, de esta forma se asegura que si dos hosts A y B en los extremos del cable envían información, el primer bit enviado por A llegará a B antes de que B haya transmitido su último bit, y viceversa.

SSH

- SSH es un protocolo, una herramienta para encriptar, una aplicación cliente/servidor o una interfaz de comandos.
- Ofrece: autenticacion, encriptación e integridad
- Puede encriptar en forma transparente el flujo de datos de otra aplicación (tunneling)

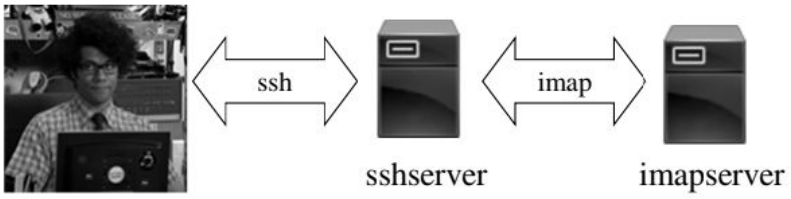
Port forwarding

- SSH port forwarding protege conexiones TCP redirigiendolas a través de una conexión SSH (solo funciona con TCP)
- Local forwarding → aplicación cliente está localizada en el cliente SSH




```
ssh -Lport:host:hostport SSH server
```

- Crea socket pasivo en localhost que escucha en port y todo lo que se manda acá se replica dentro del túnel
- -g para que otros host de la red tambien usen el tunel

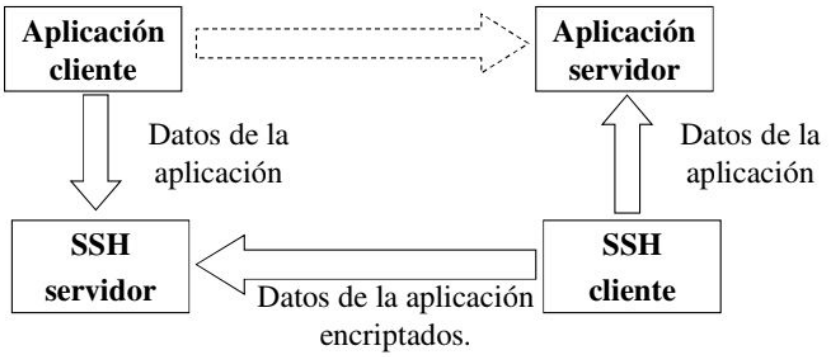


Paso 1: establecer la conexión

```
moss@lhost:~$ ssh -L2001:imapserver:143 -l user sshserver
```

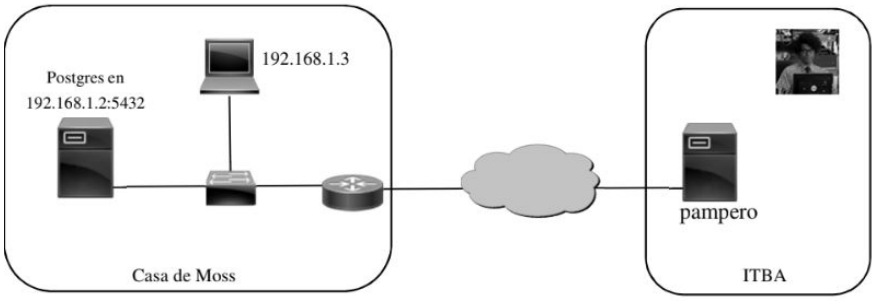
1. El lector de mails envía los datos a (localhost,2001)
2. El cliente local SSH lee el puerto 2001, encripta los datos, y los envía a través de la conexión SSH al servidor *sshserver*.
3. El servidor SSH desencripta los datos y los envía al servidor IMAP en el puerto 143 de *imapserver*.
4. Las respuestas son enviadas por el mismo túnel.

➢ Remote port forwarding → aplicación cliente está localizada en el servidor SSH



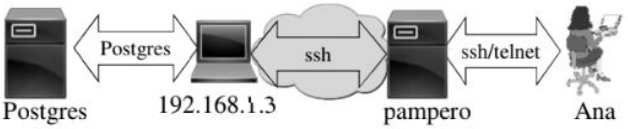

```
ssh -Rport:host:hostport SSH server
```

- **Ejemplo**: Moss tiene un servidor Postgres y quiere conectarse desde el ITBA




```
192.168.1.3:~$ $ ssh -R2001:192.168.1.2:5432 moss@pampero
```

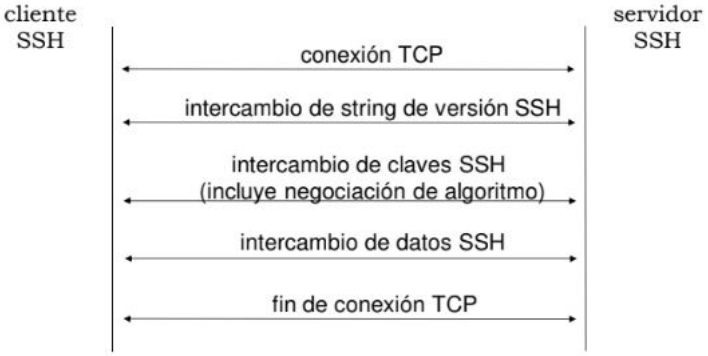
El puerto remoto será el 2001, el puerto local el 5432.
Una vez establecida la sesión, se ha creado un túnel desde el puerto 2001 del host remoto al puerto 5432 en el servidor Postgres.




```
ana@pampero:~$ psql -p 2001
```

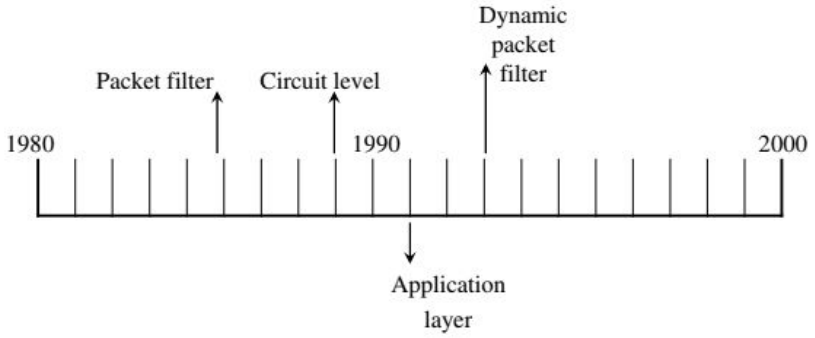
Por defecto todos los usuarios en pampero podrán usar el túnel

Conexión SSH



Firewalls

- Opera en el nivel más bajo de red
  - Evita accesos no autorizados a la Intranet
  - Permite definir qué tráfico entra y sale de nuestra red
  - Oculta host internos de redes externas



Packet Filter
- Analiza trafico a nivel de transporte
- Cada paquete IP evaluado para ver si cumple con una serie de reglas
- La acción a seguir: deny o allow

Application Layer
- Generar logs, filtrar URL
- Caché de objetos HTTP
- Modificar datos

Circuit level
Validar si un paquete pertenece a una sesión válida. Establecida la sesión, se almacena:
- Id y estado de la conexión
- Número de secuencia
- IP y puertos de origen y destino
- Interface de la red por la que arriban y salen los paquetes

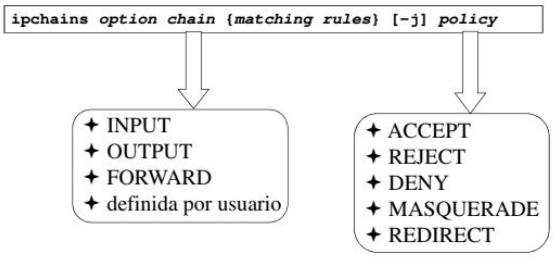
Dynamic packet filter: permiten la modificación de reglas "on the fly"

IPchains

- Cada paquete evaluado por una o más cadenas
  - Cadena → lista de reglas de cómo tratar lo paquetes
  - Si mapea se aplica la regla asociada
- Usado para controlar NAT (masquerade) y port forwarding
  - Firewall contiene una tabla para realizar NAPT

IP origen	Pto origen	IP destino	Pto destino	IP salida	Pto salida
192.168.0.1	5000	24.232.1.5	100	200.168.1.17	5000
192.168.0.2	5000	24.232.1.5	100	200.168.1.17	5010

- Acción por defecto es rechazar el paquete



IPTables

- Filtrado de paquetes y NAT
- Paquete va pasando por distintas tablas, las cuales contienen una o más listas
- Tablas independientes que utilizan ipTables
  - **Raw** → PREROUTING, OUTPUT
  - **Filter** → INPUT, PUTPUT, FORWARD
  - **Nat** → PREROUTING (DNAT), POSTROUTING (SNAT), OUTPUT
    - Traducir el IP origen/destino de una secuencia de paquetes
  - **Mangle** → PREROUTING, OUTPUT, FORWARD, INPUT, POSTROUTING
  - **Security**
- Criterios de selección: para indicar si a un paquete se le debe aplicar una regla
- Destino local → no pasa por ninguna cadena, destino no local → tabla de FORWARD

  Obs! DENY = silencioso, REJECT = aviso de error

- Targets: indica a la regla que hacer con el paquete que se ajusta al criterio de selección
  - ACCEPT, DROP, REJECT, MIRROR, LOG, DNAT, SNAT