UNIVERSIDAD
COMPLUTENSE
MADRID

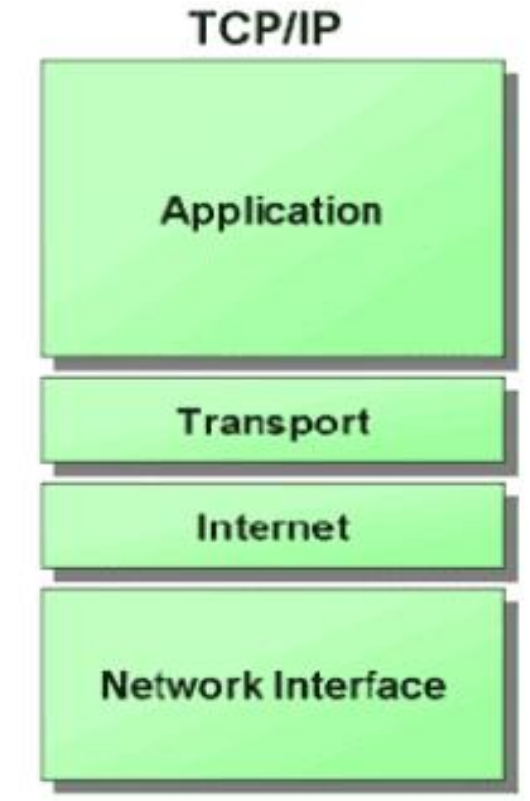
Hypertext Transfer Protocol (HTTP)

Gestión de la Información en la Web

Enrique Martín - emartinm@ucm.es
Grados de la Fac. Informática

Hypertext Transfer Protocol

- HTTP es un protocolo en la capa de aplicación de la pila TCP/IP.
- Es el protocolo que se utiliza al navegar por la Web.



Hypertext Transfer Protocol

- La definición de la versión más actual, HTTP 1.1, se puede encontrar en el RFC 2616 de junio de 1999 (http://tools.ietf.org/html/rfc2616)
- Es un protocolo que se funciona mediante **peticiones** y **respuestas** en un entorno cliente-servidor.

Recursos

- Los recursos se identifican mediante su Uniform Resource Locator (URL)
  - http://www.elpais.com/index.html
- Tiene tres partes:
  - El protocolo → **http**
  - El host → www.elpais.com
  - El identificador: **/index.html**

Recursos

- El identificador del recurso puede incluir parámetros que serán usados por el servidor web. Estos van siempre al final.
- Los parámetros se pasan utilizando el símbolo **?** tras el identificador y añadiendo parejas **nombre=valor**.
  - http://www.test.com/config.php**?color=red&lines=10**

Recursos

- El identificador del recurso también puede incluir un identificador de fragmento dentro del recurso. Para ello se utiliza el carácter **#**
- http://www.test.com/index.html**#menu**

Peticiones HTTP

- El protocolo HTTP define distintas peticiones que se le puede hacer a un servidor, aunque algunas de ellas no se utilizan casi nunca:

- **GET**
- **POST**
- **HEAD**
- DELETE
- PUT
- TRACE
- OPTIONS

Peticiones HTTP

- GET: para solicitar un documento a un servidor. Se debe utilizar únicamente cuando se quiere obtener información del servidor (aunque se usen parámetros en la URL).
- Es la que utilizan los navegadores para abrir una página web.

```
GET /index.html HTTP/1.1
Host: www.elpais.com
User-agent: test-browser/version-1.0
Accept: text/plain, text/html
```

Peticiones HTTP

- GET: se pueden añadir parámetros a la petición incluyéndolos en el identificador del recurso.

```
GET /~user/add.php?name=pepe HTTP/1.1
Host: localhost
User-agent: test-browser/version-1.0
Accept: text/plain, text/html
```

Peticiones HTTP

- POST: para enviar datos al servidor. En lugar de añadir parámetros en la URL, los datos se envían en el cuerpo de la petición HTTP.

```
POST /index.php HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 ...
Content-Type: application/x-www-form-urencoded
Accept: text/html,application/xhtml+xml, ...
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
Content-Length: 17

name=Pepe&edad=28
```

Peticiones HTTP

- Las peticiones POST se deben utilizar cuando enviemos datos al servidor que impliquen **realizar algún cambio en el mismo**, p.ej. añadir un nuevo usuario (modificará su base de datos).
- Si lo único que queremos es enviar datos para obtener una página personalizada, p.ej. visualizar los pedidos, se utilizará una petición GET.

Peticiones HTTP

- HEAD: para pedir únicamente la cabecera del documento en lugar de todo el documento.
- Útil para realizar sin mucha sobrecarga alguna comprobación sobre los metadatos, p.ej. si ha sido actualizado.
  - Usado por *web crawlers*.

```
HEAD / HTTP/1.1
Host: elpais.com
User-agent: test-browser/version-1.0
Accept: text/plain, text/html
```

Peticiones HTTP

- Las siguientes peticiones raramente se utilizan (salvo en servicios **REST**):
  - DELETE: borrar un documento
  - PUT: para añadir un recurso al servidor
  - TRACE: devuelve información de diagnóstico. Contesta con la petición recibida para saber si algún paso intermedio ha introducido algo.
  - OPTIONS: para saber qué peticiones acepta el servidor para un determinado recurso.

Contestaciones HTTP

- Cuando un servidor web recibe una petición HTTP contestará con un mensaje conteniendo:
  - **Estado**: 200 OK, 404, 301, etc.
  - **Cabecera**: fecha, servidor, longitud...
  - **Cuerpo**: el código HTML

Contestaciones HTTP

```
HTTP/1.0 200 OK
Date: Fri, 31 Dec 1999 23:59:59
GMT
Content-Type: text/html
Content-Length: 1354

<html>
<body>
<h1>Happy New Millennium!</h1>
….
</body>
</html>
```

Contestaciones HTTP

- El estado de una contestación HTTP puede estar en 5 familias:
  - **1xx → Información**, p.ej. informar de que se ha recibido la petición y se está procesando.
  - **2xx →** La petición ha tenido **éxito**.
  - **3xx → Redirección**, para informar que el recurso ha cambiado de dirección.
  - **4xx → Error del cliente**: mala sintaxis, fichero no encontrado, acceso prohibido.
  - **5xx → Error del servidor**. La petición era correcta pero el servidor ha fallado.

HTTP Seguro (HTTPS)

Criptografía

- La **criptografía** sirve para dotar de **seguridad a las comunicaciones**:
  - Confidencialidad
  - Integridad
  - No repudio
- Se distinguen dos **tipos** principales de criptografía:
  - De clave **secreta** (o simétrica)
  - De clave **pública** (o asimétrica)

Criptografía de clave secreta

- La **misma clave** se utiliza para cifrar y descifrar el mensaje.
- Por tanto, la clave debe mantenerse **secreta** y compartirse por canales seguros → *problema de compartición*.
- Este tipo de cifrado es bastante **rápido** incluso para cifrar grandes cantidades de datos.
- Se distinguen dos **tipos** principales:
  - De bloque
  - De flujo

Criptografía de clave pública

- En lugar de una sola clave secreta, cada participante tiene un **par de claves**: una **privada** y una **pública**.
- La clave **pública** se puede **distribuir libremente** a cualquier participante.
- La clave **privada** solo debe conocerla el propio dueño → **secreta**.
- El par de claves son **duales**: el mensaje que se cifra con una clave únicamente se puede descifrar con la otra.

Criptografía de clave pública

- Además, es **impracticable** obtener la clave privada a partir de la pública.
- Tiene varios **usos**:
  - Si A cifra el mensaje M con su clave privada, cualquiera que lo reciba podrá descifrarlo → **integridad**, **no repudio**
  - Si A cifra el mensaje M con la clave pública de B, solo B podrá leerlo → **confidencialidad**
- La criptografía de clave pública es más **lenta** que la de clave secreta
- No presenta *problema de compartición*.

SSL y TLS

- *Secure Sockets Layer* (**SSL**) y su sucesor *Transport Layer Security* (**TLS**) son protocolos criptográficos para comunicación en **redes inseguras**.
- Usan el cifrado de **clave pública** y de **clave secreta** (simétrica).
- Son los protocolos que utiliza el HTTP seguro (**HTTPS**).

Creación de una conexión TLS

- 1) El usuario se conecta a un servidor, que le envía su **certificado**. Este certificado:
  - Contiene la clave pública del servidor.
  - Está firmado por una autoridad de certificación.
- 2) El navegador **verifica el certificado** del servidor. Para ello utiliza la clave pública de la autoridad de certificación.
  - Este proceso puede requerir de varios pasos de verificación usando autoridades de certificación intermedias hasta llegar a una de confianza (FNMT, Verisign, Thawte...)

Creación de una conexión TLS

- 3) Usando la clave pública del servidor, el usuario envía de manera **segura** un **valor aleatorio** al servidor.
- 4) A partir del valor aleatorio conocido únicamente por ambas partes, se genera una **clave secreta**. Esta clave se usará para cifrar de manera simétrica la comunicación → **rapidez**.

Referencias y recursos

Referencias

- Secure Socket Layer (SSL):
  - http://www.criptored.upm.es/intypedia/video.php?id=introduccion-ssl&lang=es
  - http://www.criptored.upm.es/intypedia/video.php?id=ataques-ssl&lang=es

Recursos sobre criptografía

- Un buen lugar para buscar información básica sobre criptografía y su uso es **Intypedia** (www.criptored.upm.es/intypedia)
  - Encontraréis vídeos, transparencias y ejercicios.
- Libro electrónico de seguridad informática y criptografía. Jorge Ramió Aguirre. http://www.criptored.upm.es/guiateoria/gt_m001a.htm

Recursos sobre criptografía

- Handbook of applied cryptography. Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone. (http://cacr.uwaterloo.ca/hac/)