1 O nivel de aplicación

1.1 Sumario

2 Introdución

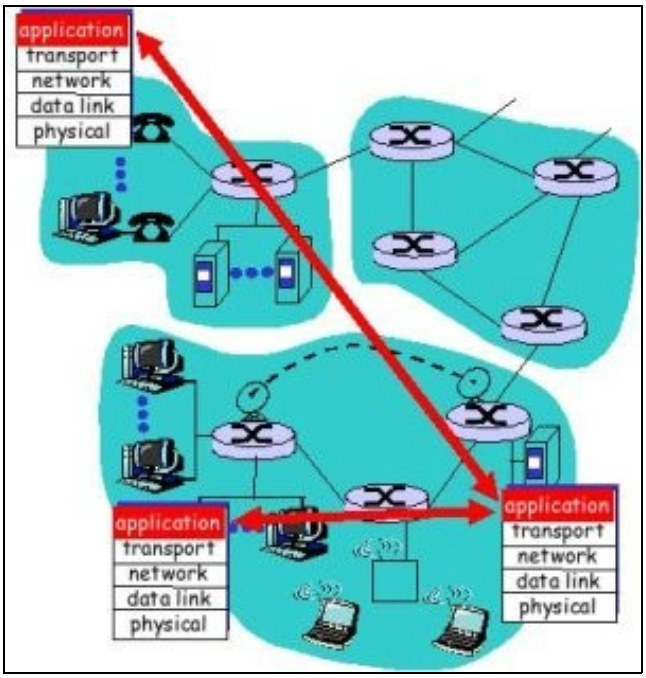
Neste tema descríbese toda unha serie de aplicacións ou programas que utilizan a rede como medio de comunicación, así como os protocolos de comunicacións que teñen asociados. Devanditas aplicacións coñécense como **aplicacións distribuídas**, posto que están formadas por diferentes partes e cada unha se atopa en máquinas diferentes. Por norma xeral, hai unha parte chamada **servidor** que se executa nun computador, á que se conectan os diferentes **clientes** (que se atopan noutros computadores remotos) para requirir os seus servizos (polo xeral, solicitan a execución dalgún tipo de operación).

Existe unha gran cantidade de aplicacións que seguen este modelo e que, nalgúns casos, usamos diariamente, entre elas: o correo electrónico, a navegación Web, a mensaxería instantánea, a telefonía IP, o acceso remoto a outros computadores, a vídeoconferencia, a computación paralela masiva, a compartición de ficheiros, os xogos en rede multiusuario, a descarga de vídeo e audio e un longo etc.

Estas aplicacións empregan protocolos concretos para poder entenderse. Este protocolos atópanse no nivel de aplicación.

3 Arquitecturas de aplicacións en rede

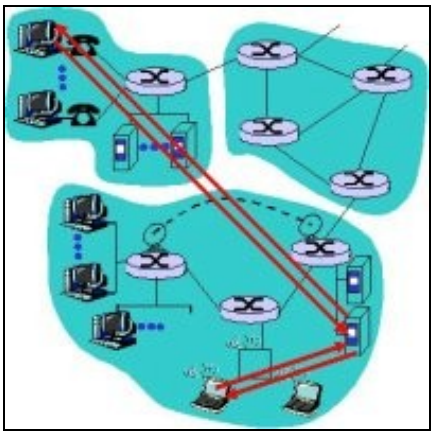
As aplicacións execútanse en diferentes computadores no que se coñece como **sistemas finais**, e comunícanse mediante a rede. Por exemplo: un programa que fai de servidor Web comunícase cun programa navegador. Xa que logo, a comunicación para unha aplicación en rede ten lugar entre sistemas finais no nivel de aplicación.



Hai tres tipos de arquitecturas para as aplicacións en rede:

- Arquitectura Cliente/Servidor
- Arquitectura Peer-to-peer (P2P)
- Arquitectura híbrida entre cliente/servidor e P2P

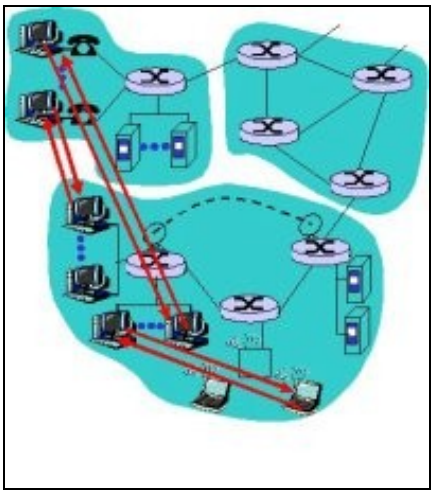
3.1 Arquitectura cliente/servidor


O modelo ou arquitectura cliente/servidor está formado por dúas partes.

- **O servidor**. Sempre está funcionando (24x7) e ten unha dirección IP permanente. Pódense instalar granxas ou clusters de servidores para dar servizo a un maior número de usuarios.
- **Os clientes**. Poden estar conectados intermitentemente. As súas direccións IP son dinámicas. Os clientes non se comunican directamente entre eles, senón que o fan co servidor

Moitas das aplicacións de uso frecuente de Internet, como a navegación Web ou o correo electrónico, usan este modelo.

3.2 Arquitectura P2P pura



Este modelo ten unhas características diferenciadas respecto á cliente/servidor pura. Estas diferenzas son as seguintes:

- Non existe un servidor acendido permanentemente
- Non se distingue entre clientes e servidores, senón que existen sistemas arbitrarios finais, chamados **peers**, que se comunican directamente
- Os peers están conectados intermitentemente e cambian a súa dirección IP con frecuencia
- Todos os nodos teñen a mesma función, peso e importancia dentro da rede

Un exemplo de programa que usa unha arquitectura P2P pura é o software de intercambio de ficheiros Gnutella, que usa protocolo de distribución de ficheiros entre pares, sen un servidor central.

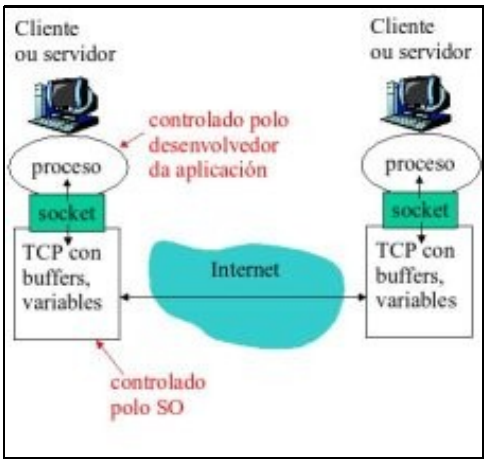
A principal vantaxe desta arquitectura fronte a anterior é que é moi **escalable**. Como contrapartida é máis **difícil de xestionar**.

3.3 Arquitecturas híbridas C/S e P2P

Son unha mistura das anteriores. De tal xeito que para algunhas funcións usan un servidor central e para outras fan unha comunicación directa con outro peer. Vexámolo con dúas aplicacións de exemplo:

- **eMule**. No programa de intercambio de ficheiros eMule a transferencia de ficheiros realízase mediante P2P. Con todo, a busca de ficheiros é centralizada xa que os pares, ou peers, rexistran contido nun servidor central (os ficheiros que comparten). Para localizar un contido, o pares consultan un servidor central ou varios servidores distribuídos, pero para a transferencia usan P2P.
- **Mensaxería instantánea**. As conversas que se manteñen entre dous usuarios realízase mediante P2P, pero o rexistro e localización é mediante C/S, é dicir, centralizada nun servidor. De feito, os usuarios rexistran a súa IP nun servidor cando arrincan o programa. Este servidor é un servidor central de contactos para IP de amigos.

4 Comunicación de procesos



Un proceso é un programa que se executa nun computador. Un **proceso cliente** é o que inicia a comunicación. Un **proceso servidor** é o que espera a ser contactado. Por exemplo: O programa `dhclient` lanza un proceso cliente que solicita unha dirección IP. O programa `dhcpd` espera a recibir peticións para respostar con direccións IP dinámicas aos clientes.

As aplicacións están formadas por varios procesos. No caso das arquitecturas P2P as aplicacións teñen procesos clientes e procesos servidores ao mesmo tempo.

Os procesos pertencen a un usuario que, normalmente, os lanza ou provoca que se lancen. Para ver os procesos dun usuario que se están executando nun computador para podes teclear:

```
ps --u nome_usuario
```

Nun computador pode haber varios procesos executándose polo que se identifican mediante un número de **porto**. Os procesos comunícanse uns cos outros a través dos **sockets**. Un socket especifica a dirección, o porto e o protocolo de transporte que se usa na comunicación. Por exemplo: (**192.168.100.10**, **80**, **TCP**).

5 Protocolos do nivel de aplicación

Os procesos comunícanse enviándose **mensaxes**. As características destas mensaxes están determinadas no protocolo, o cal define:

- Os **tipos** de mensaxes que se intercambian, i.e: mensaxes de solicitude e resposta
- A **sintaxe** dos tipos de mensaxe, i.e: campos que terán as mensaxes
- A **semántica** dos campos, é dicir, o que significa a información dos campos das mensaxes
- As **regras** sobre cando e como os procesos poden enviar e respostar mensaxes

Os protocolos do nivel de aplicación poden ser de dominio público ou propietarios. No primeiro caso, están definidos nos RFC (*Request For Comments*). Ao estar publicamente accesibles permiten a interoperabilidade entre aplicacións de distintos fabricantes. Exemplos deste tipo de protcolos son HTTP (*HyperText Transfer Protocol*) ou o SMTP (*Single Mail Transfer Protocol*). No segundo caso é o fabricante dun determinado software quen o define e a interoperabilidade non está garantida. Un exemplo deste tipo de protocolos é o que usa a aplicación Skype.

5.1 O HTTP

5.1.1 Un pouco de historia

O HTTP naceu no 1989 no CERN (Centro Europeo de Investigación Nuclear). A Web xurdiu pola necesidade de lograr que os equipos de investigadores dispersos internacionalmente colaborasen, usando un conxunto sempre cambiante de informes, planos, debuxos, fotos e outros documentos.

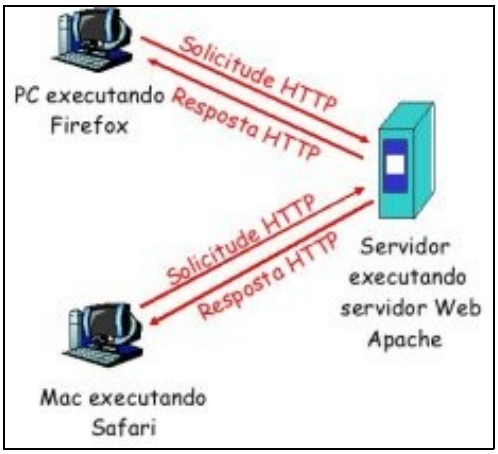
A proposta inicial dunha rede (web) de documentos vinculados xurdiu do físico do CERN **Tim Berners-Le**. O primeiro prototipo estaba baseado só en texto. En decembro de 1991 fíxose unha demostración pública e o desenvolvemento continuou durante o seguinte ano, culminando coa liberación da primeira interface gráfica, Mosaic, no ano 1993. Mosaic tivo tanto éxito que, un ano despois, o seu autor, Marc Andreeseen formou a compañía, Netscape Communications Corp.

En 1994, o CERN e o M.I.T. asinaron un acordo para establecer o World Wide Web Consortium, unha organización adicada ao desenvolvemento da Web, a estandarización de protocolos e o fomento da interoperabilidade entre as instalacións. Berners-Le converteuse no director.

5.1.2 Conceptos previos

As **páxinas Web** están formadas por **obxectos**. Os obxectos poden ser ficheiros HTML, imaxes, ficheiros de audio, vídeo, etc. Unha páxina web é un **ficheiro HTML base** que inclúe varias referencias aos obxectos. A cada obxecto accédese mediante un **URL** (*Uniform Resource Locator*) que ten a seguinte forma:

5.1.3 Características



O HTTP é un protocolo do nivel de aplicación para a Web. Segue o modelo ou **arquitectura cliente/servidor**, tal e como se ve na figura. O cliente é un navegador (*browser*) que solicita, recibe, e mostra obxectos Web. O servidor Web envía obxectos como resposta ás peticións do cliente. Hai dúas versións do protocolo, o HTTP 1.0, especificada no RFC 1945, e o HTTP 1.1, especificado no RFC 2068.

HTTP **utiliza TCP como protocolo de transporte**. Os clientes inician conexións TCP (crean un socket) co servidor, no porto 80. Os servidores aceptan conexións TCP dos clientes. Unha vez establecida a conexión intercámbianse mensaxes HTTP (mensaxes do protocolo do nivel de aplicación) entre o cliente (cliente HTTP) e o servidor Web (servidor HTTP). Finalmente, péchase a conexión TCP.

HTTP non ten **?estados?** (*stateless*). Isto quere dicir que o servidor non mantén información sobre solicitudes anteriores dos clientes. Deseñouse así porque os protocolos que manteñen o estado son complexos, xa que hai que manter un histórico do que acontece (estado). Se o cliente ou o servidor fallan pode haber inconsistencias e deberán volver ao estado anterior, co correspondente consumo de recursos.

5.1.4 Tipos de conexións

As conexións HTTP poden ser de dous tipos:

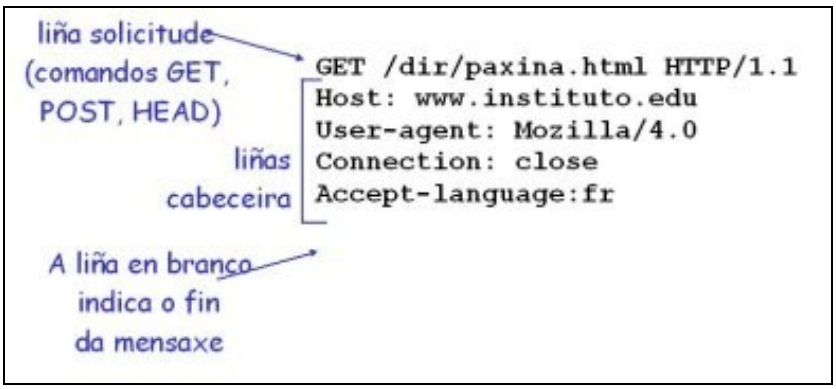
- **Conexión non persistentes**. Nelas, envíase un obxecto como máximo en cada conexión TCP. Para cada nova petición do cliente precísase unha nova conexión TCP. HTTP/1.0 utiliza HTTP non persistente. Por exemplo, se unha páxina Web ten referencias a 10 obxectos, abriranse 10 conexións TCP.
- **Conexións persistentes**. Envíanse varios obxectos nunha única conexión TCP entre o cliente e o servidor. HTTP/1.1 utiliza conexións persistentes por defecto. As conexións persistentes supoñen menos sobrecarga para o SO xa que os navegadores non abren varias conexións TCP paralelas para conseguir os obxectos referenciados. O servidor deixa a conexión TCP aberta despois de enviar unha resposta, xa que logo, as mensaxes HTTP entre o mesmo cliente e servidor usan a mesma conexión TCP.

5.1.5 Mensaxes

En HTTP hai mensaxes de **solicitude** e mensaxes **resposta**. Ambas as dúas mensaxes están en formato ASCII, polo que poden verse e lerse se capturamos unha sesión HTTP cun analizador de rede como, por exemplo, o Wireshark.

5.1.5.1 Solicitudes

Unha mensaxe HTTP de solicitude ten o seguinte formato:



Se queremos facer unha solicitude de envío de datos (por exemplo, cando facemos unha consulta nun buscador enviámoslle ao servidor a palabra a buscar, ou cando enviamos un formulario) podemos usar dous métodos:

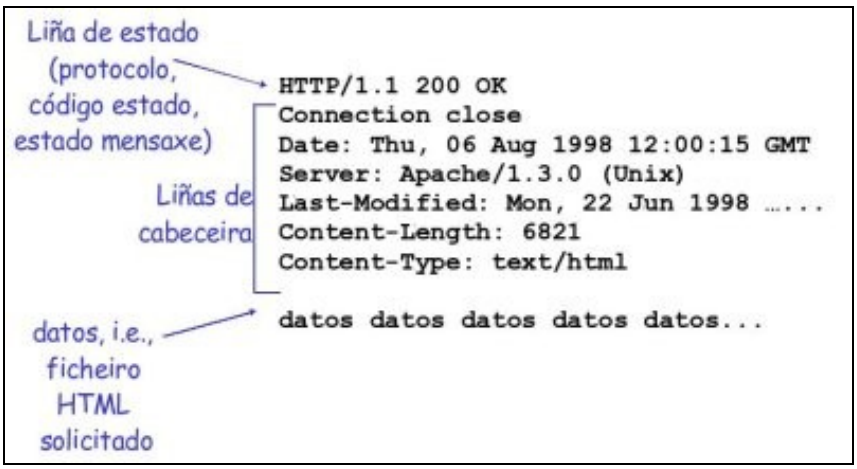
- **Método POST**, no que os datos se envían ao servidor dentro da mensaxe de solicitude HTTP.
- **Método URL**, tamén chamado método GET, no que os datos se envía na URL da liña de solicitude, por exemplo `www.unsite.com/buscaAnimal?mono&banana`

Dependendo da versión do protocolo que se use están soportadas distintas mensaxes HTTP (tamén coñecidas como métodos):

- **Para HTTP 1.0**: están soportadas as mensaxes **GET** e **POST** xa vistas. Tamén está soportada a mensaxe **HEAD** que só devolve a cabeceira, non o obxecto referenciado
- **Para HTTP 1.1**: están soportadas as tres mensaxes anteriores e tamén **PUT**, que permite subir o ficheiro especificado no URL ao servidor, e **DELETE**, que permite borrar o ficheiro especificado no URL.

5.1.5.2 Respostas

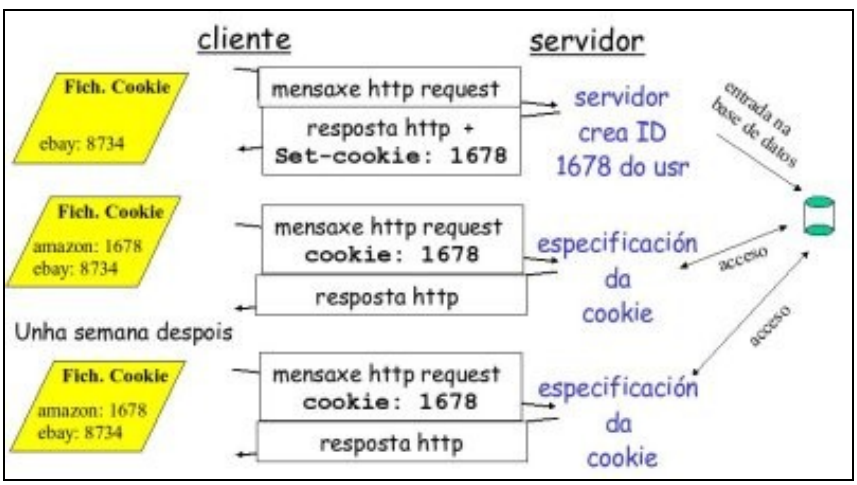
As mensaxes HTTP de resposta teñen un formato similar ás de solicitude:



As mensaxes de resposta inclúen un código dependendo do significado da mesma. Dito código vai na primeira liña da mensaxe (liña de estado). Algúns exemplos son os seguintes:

```
200 OK
  Solicitude correcta, envíarase o obxecto solicitado nesta mensaxe

301 Moved Permanently
  Obxecto solicitado noutro URL, especifícase a nova localización nesta mensaxe (Location:)

400 Bad Request
  Mensaxe non entendida polo servidor

404 Not Found
  Obxecto solicitado non atopado no servidor

505 HTTP Version Not Supported
```

5.1.6 As cookies



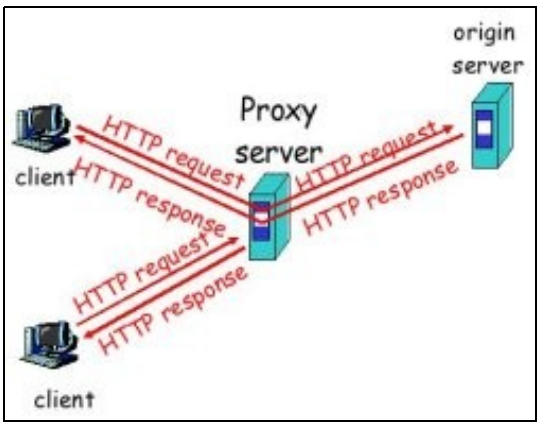
Diximos que HTTP é un protocolo sen estado. Isto, ás veces, pode non ser o desexado. Por exemplo, supoñamos que temos unha web na que as súas páxinas están restrinxidas a determinados usuarios. Se non mantemos o estado do cliente teremos que estar solicitándolle para cada páxina que queira ver o seu nome de usuario e clave. HTTP resolve este escenario coas cookies.

As cookies están definidas no RFC 2109 e permiten monitorizar a navegación do usuario. Moitos sites usan cookies. Son útiles, por exemplo, para validar un usuario, escoller opcións personalizadas (idioma, etc.), rexistrar hábitos de navegación, etc. Están formadas por catro compoñentes:

1. Liña de cabeceira da cookie na resposta HTTP.
2. Liña de cabeceira da cookie na solicitude HTTP.
3. Ficheiro que se mantén no computador do cliente, xestionado polo navegador.
4. Base de datos no servidor.

Un exemplo de funcionamento das cookies podémolo ver na figura.

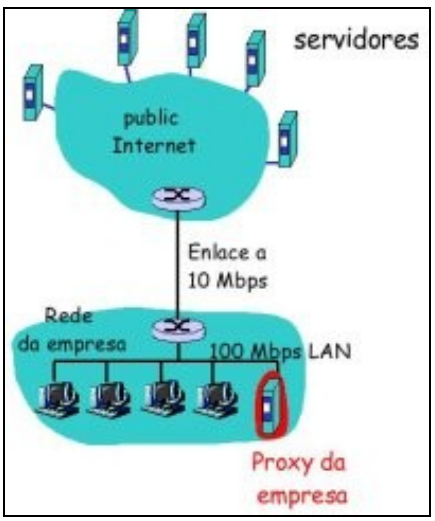
5.1.7 Servidores Proxy



Dun modo xenérico, pódese definir un proxy como un software que fai de intermediario entre un cliente e un servidor. Os proxies en HTTP úsanse, entre outras cousas, para facer de caché das páxinas visitadas. É o que se coñece como un **Web cache** e o seu obxectivo é respostar aos clientes sen interactuar co servidor, gañando en velocidade, xa que o proxy está normalmente na mesma rede ou máis cercano que o servidor Web que contén a páxina orixinal.

O proxy configúrase no navegador. É o navegador quen envía as solicitudes HTTP ao proxy. Se o obxecto está na cache o proxy é quen o devolve. Se non, o proxy solicita o obxecto ao servidor orixinal que á súa vez é enviado ao cliente. O proxy ten, xa que logo, unha parte cliente e outra servidor. Normalmente instálanse en universidades, compañías, ISP, etc., para axilizar o tráfico. Entre as vantaxes de usar un servidor proxy están as seguintes:

- Reducir o tempo de resposta das solicitudes dos clientes.
- Reducir o tráfico, e polo tanto ampliar o largo de banda.
- Filtrar contido.



Un exemplo de software de servidor proxy é o Squid.

5.1.8 Instalación e configuración do Apache

Instalación e configuración do servidor Web Apache

5.2 O File Transfer Protocol

O FTP

5.3 O DNS

O DNS

5.4 O correo electrónico

O correo-e

--Arribi 11:12 27 ene 2009 (GMT)