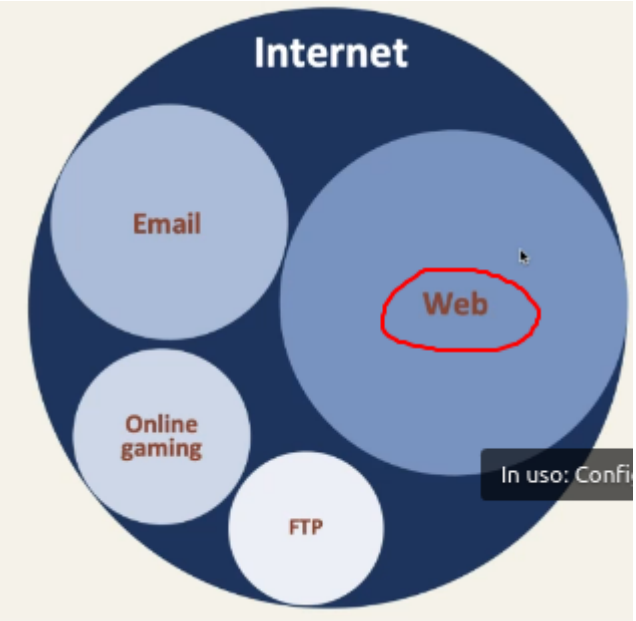
03/02/2023

Introduzione

- Docenti: Alessio Vecchio, Cimino
- alessio.vecchio@unipi.it
-

How the web works



Nasce intorno agli anni 90 i suoi creatori sono Tim Berners-Lee e Robert Cailiau.

I concetti fondamentali del web:

- URL: uniform resource locator modo in cui identifichiamo una risorsa all'interno del web, nella barra del browser quando inseriamo un URL stiamo identificando una pagina web
- HTTP: procollo che permette di scambiare informazioni in particolare permette di richiedere una certa risorsa è un protocollo basato un richiesta e risposta. Il server risponde con pagine web, possono essere immagini o frammenti audio/video
- Programma software lato server ovvero WEB SERVER che soddisfa le richieste da parte del client
- HTML è il linguaggio utilizzato per pubblicare documenti nel web, permette di definire la struttura del documento, quindi indicare che c'è una parte di intestazione della pagina, etc...
- Client che fa le richieste HTTP verso il web server e il client usa la risposta per far visualizzare all'utente la risposta

W3C

Nel '94 Berners-Lee fonda il World Wide Consortium che standardizza i meccanismi del web e guida lo sviluppo del web stesso. Tutte le tecnologie iniziali erano tecnologie libere non soggette a royalty questo è stato la base del successo del web all'inizio.

Applicazioni web vs Desktop App

Pro:

- Usufruibile da qualsiasi calcolatore collegato alla rete internet
- Nessuna necessità di installare software aggiuntivi
- Le webapp sono agnostiche dall'OS o dalle caratteristiche hardware del client
- Più semplice aggiornare il codice in quanto centralizzato ed è necessario installarlo solo sul web server
- Per sua natura centralizzata è installata in un server con dati centralizzati che può essere vantaggioso in certi domini applicativi

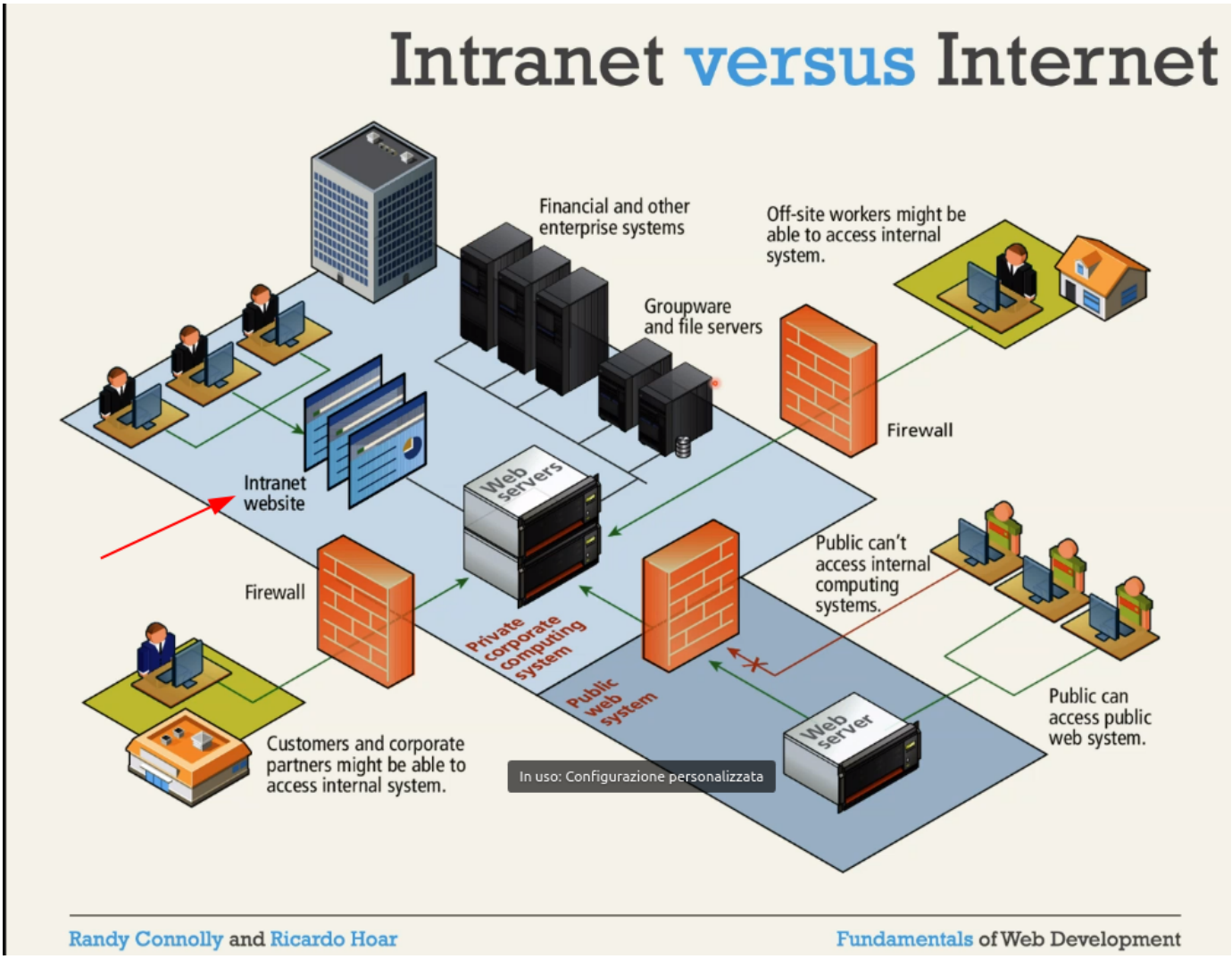
Contro:

- Necessario avere una connessione alla rete
- Esiste uno scambio continuo di informazioni che transitano su delle reti sconosciuti e chi siano i gestori delle reti attraversate
- In termini di licenze se pensiamo ad alcune regole imposte dall'Unione Europea che riguardano la gestione dei dati dei cittadini dell'unione, la posizione geografica è importante in quanto queste regole includono il concetto di confine geografico dell'UE ma se i dati escono dai confini per raggiungere i server dislocati altrove potrebbe essere un problema
- I browser non sono esattamente tutti uguali e quindi una webapp potrebbe funzionare meno bene con un altro browser
- La nostra applicazione quando viene eseguita su un browser viene eseguita lato client viene e quindi tutti gli strati software (OS, framework, etc...) possono aggiungere una limitazione. Ad esempio quando i browser hanno rimosso il supporto per Adobe Flash Player.
-

**

Intranet

Si intende una rete locale tipicamente interna ad un organizzazione o un azienda:

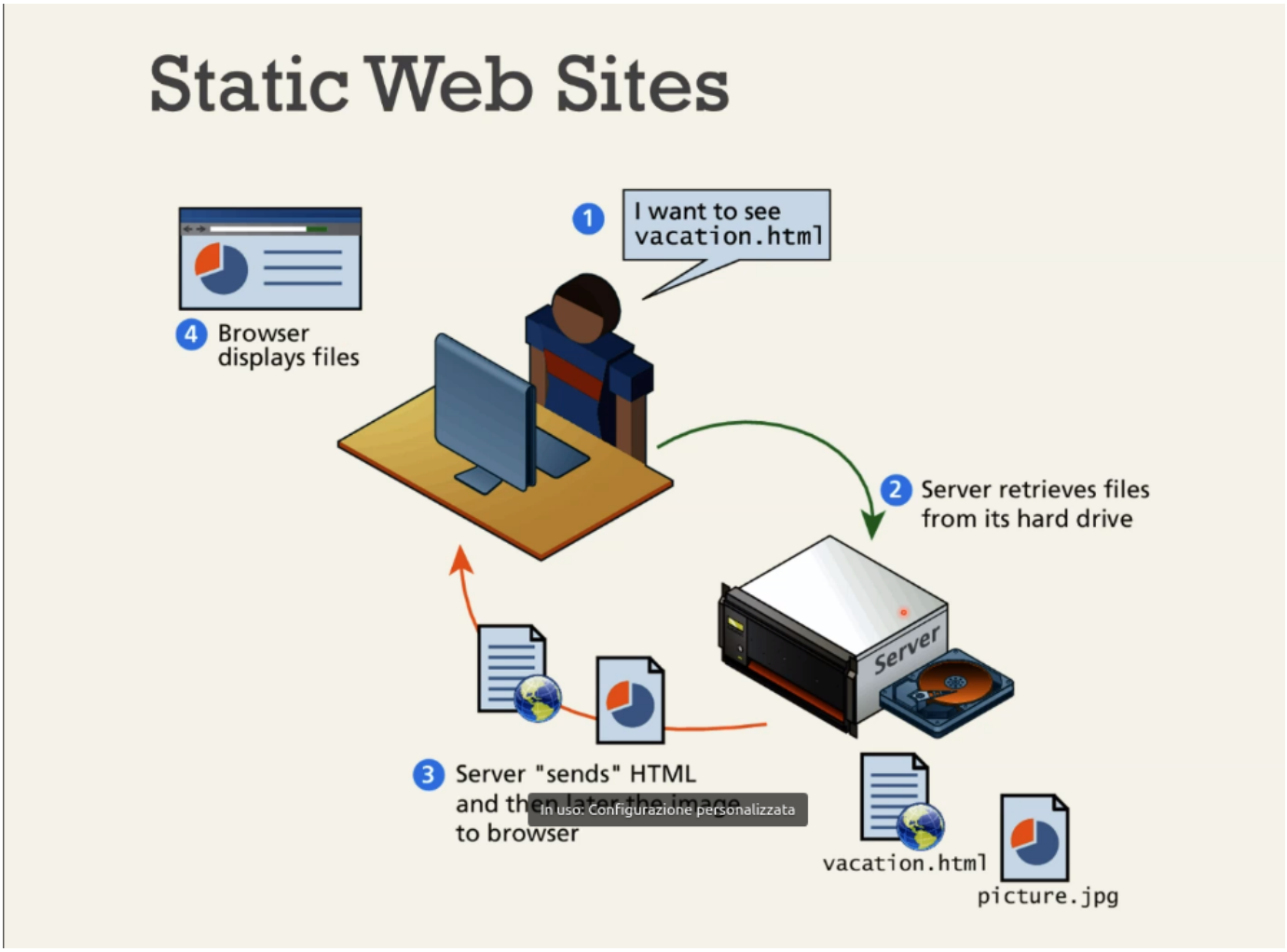


Anche nel caso di una rete locale è possibile adoperare le tecnologie web. In questo caso abbiamo una rete in azzurro chiaro più o meno isolata dal mondo esterno. Tipicamente si vuole che ad esempio i web server siano accessibili dal mondo esterno. I dipendenti del mondo esterno accedono al mondo interno aziendale passando per un firewall. Potrebbero esserci degli utenti che hanno bisogno di visitare delle pagine web messe a disposizione dall'azienda stessa, quindi il web server esterno fornirà le pagine ai visitatori esterni e può accedere ai sistemi interni sempre passando da un firewall.

Static web sites

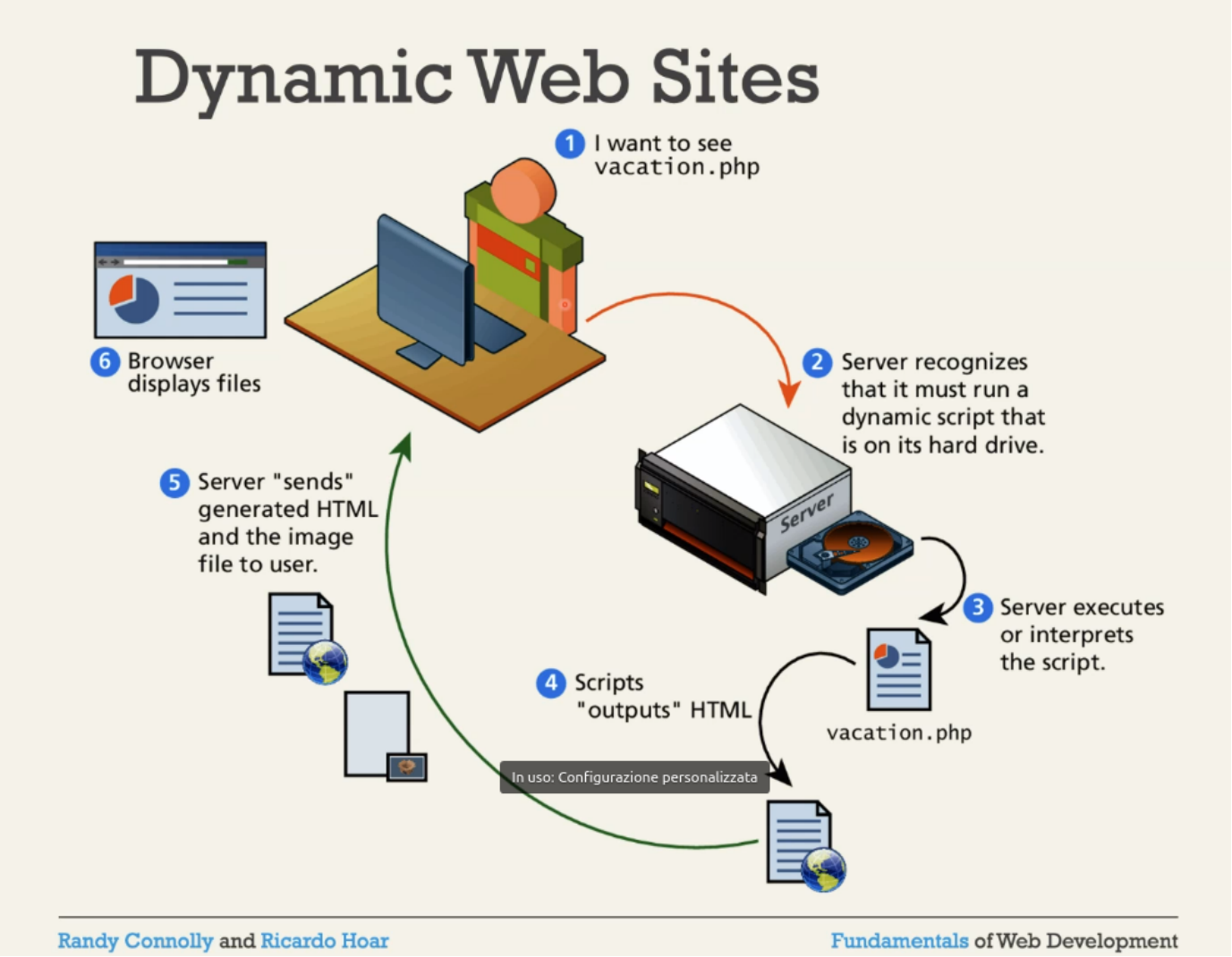
Una pagina web è sostanzialmente un documento HTML salvato su file che può incorporare delle immagini. Un tempo il webmaster scriveva i file HTML che erano poi ospitati in opportune cartelle del webserver e quindi erogati ai client.

Alcuni contenuti ancora oggi sono contenuti statici:



Dynamic web sites

I contenuti non sono memorizzati all'interno di file ma prodotti dinamicamente tramite esecuzione di programmi o script:



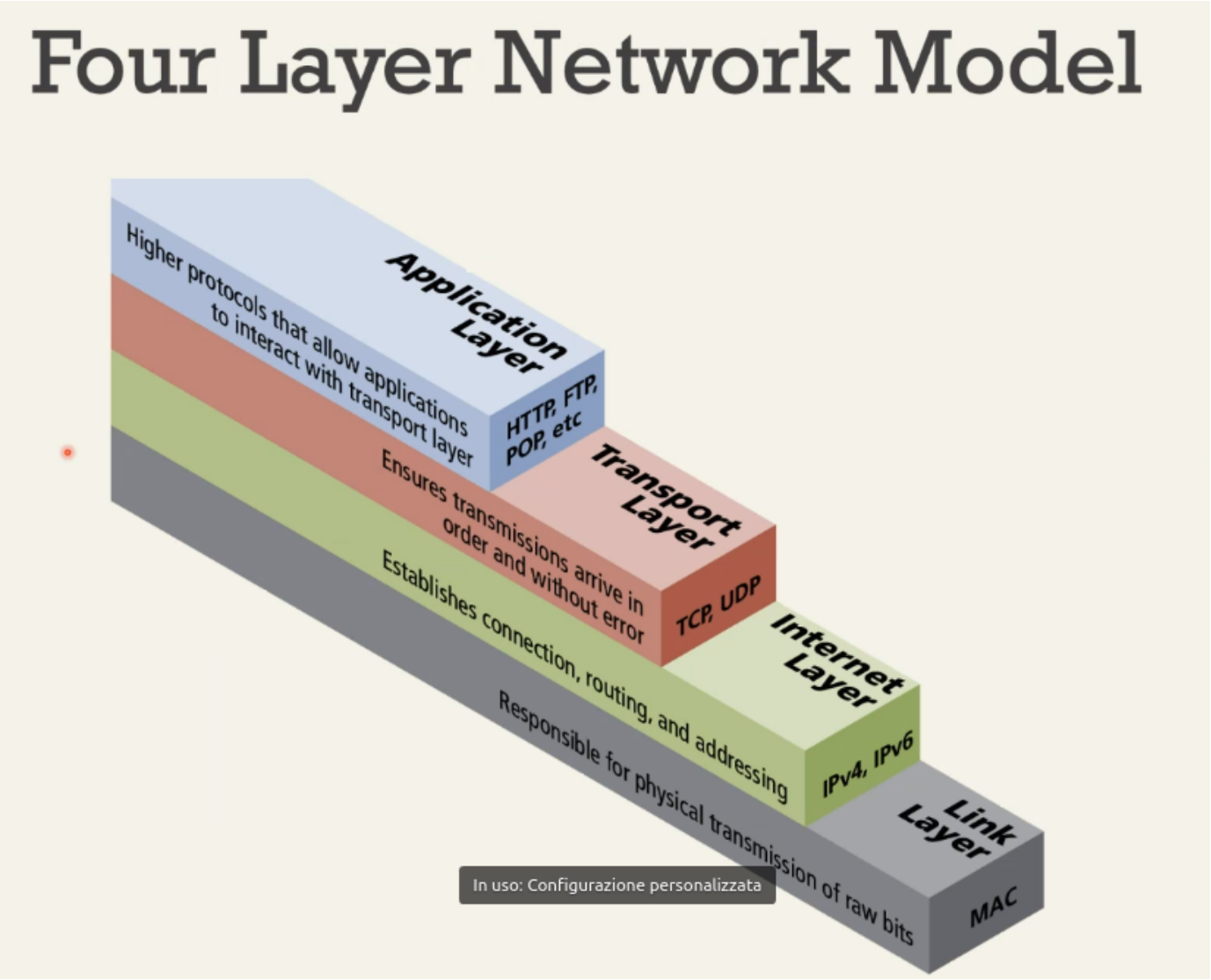
Quando il programma viene mandato in esecuzione può attingere a delle sorgenti di dati ulteriori ad esempio da un database customizzando il contenuto sulla base del contenuto del database.

Web 2.0

Intorno alla metà degli anni 2000 si voleva indicare con tale termine un modificarsi di come i dati dovevano essere prodotti e consumati. Con le pagine statiche l'utente è solo un consumatore di contenuti e anche nel caso dinamico. Molti contenuti venivano invece prodotti dagli utenti stessi (si pensi ai social network), i contenuti nascono lato client e devono essere resi disponibili dagli utenti del social network. Questo è stato un momento importante in quanto richiedeva una logica lato client molto più complessa del semplice fruire i contenuti già pronti. Questo è avvenuto tramite Javascript che ci permette di eseguire dei programmi tramite il browser dell'utente.

Internet protocols

Un protocollo è un linguaggio comune che permette alle entità di parlare. Alcuni dei protocolli più importanti sono il TCP e l'IP e vediamo quali sono le funzionalità più importanti:

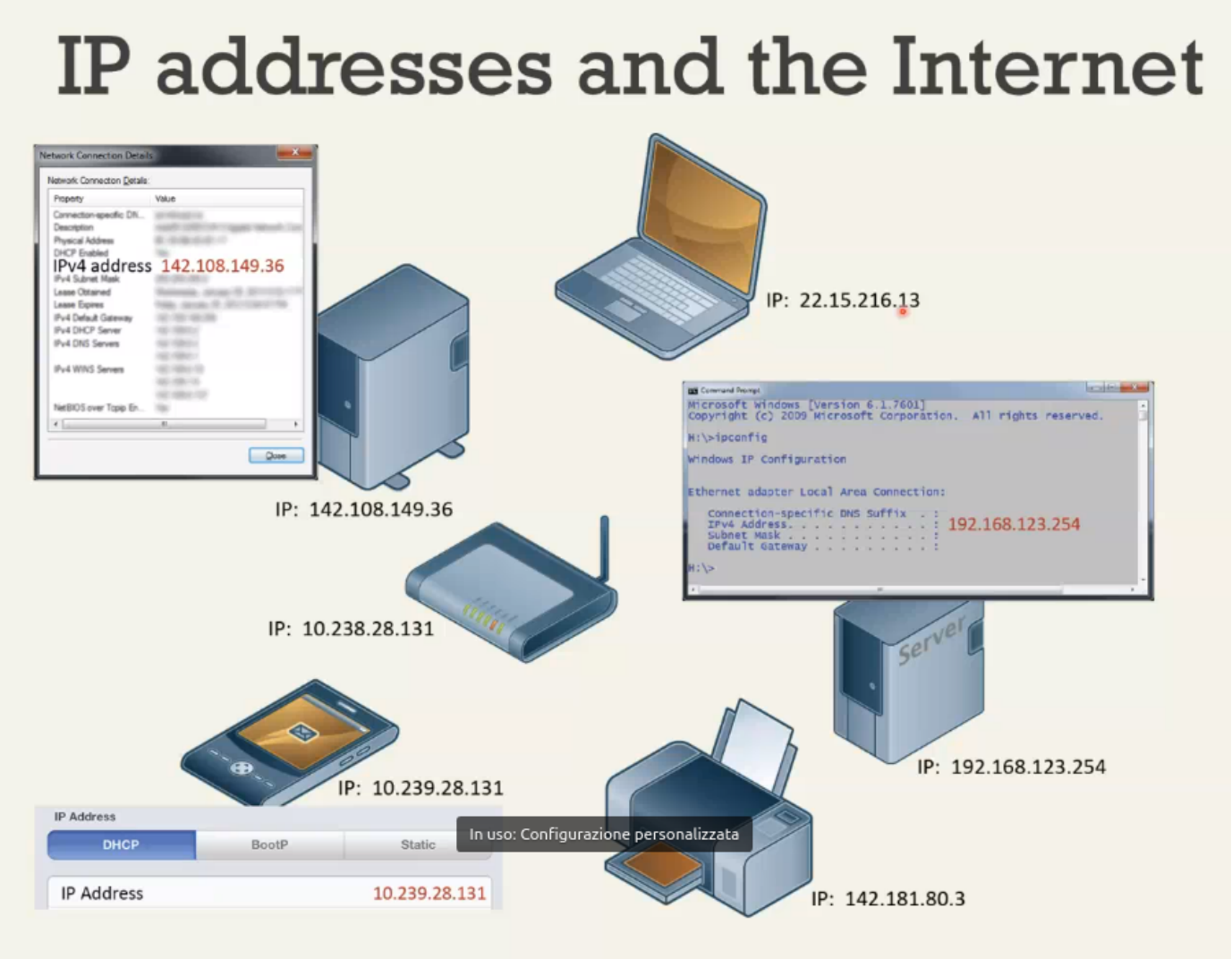


Ogni strato cerca di risolvere alcuni problemi che provengono dallo strato sottostante.

Link Layer: permette **a due calcolatori sulla stesse rete locale** di trasferire dei bit (es. la rete di dipartimento universitario o aziendale). Abbiamo la capacità di far parlare due calcolatori vicini tra di loro anche in termini fisici. Noi abbiamo la necessità di far parlare dei calcolatori dislocati in diverse parti del mondo. Per permettere la comunicazione tra due calcolatori qualunque sulla faccia della terra si usa il protocollo ***Internet Protocol***.

IP: del protocollo IP ne solo state realizzate alcune versioni diverse. Noi però vogliamo far parlare processi tra di loro. Ad esempio il processo del browser sul client con il processo web server sul server. Ecco che si parla del transport layer. IP è un protocollo non affidabile, il pacchetto viaggia attraverso i router della rete fino alla destinazione ma può succedere che non arrivi e non viene ritrasmesso. Non ci sono meccanismi di recupero a livello dell'IP. Può succedere per diversi motivi ad esempio il router potrebbe essere congestionato oppure spento oppure rotto etc... Dovendo far parlare due calcolatori

qualunque sulla faccia della terra dobbiamo identificarlo con il suo indirizzo IP (versione 4 usa 32 bit).



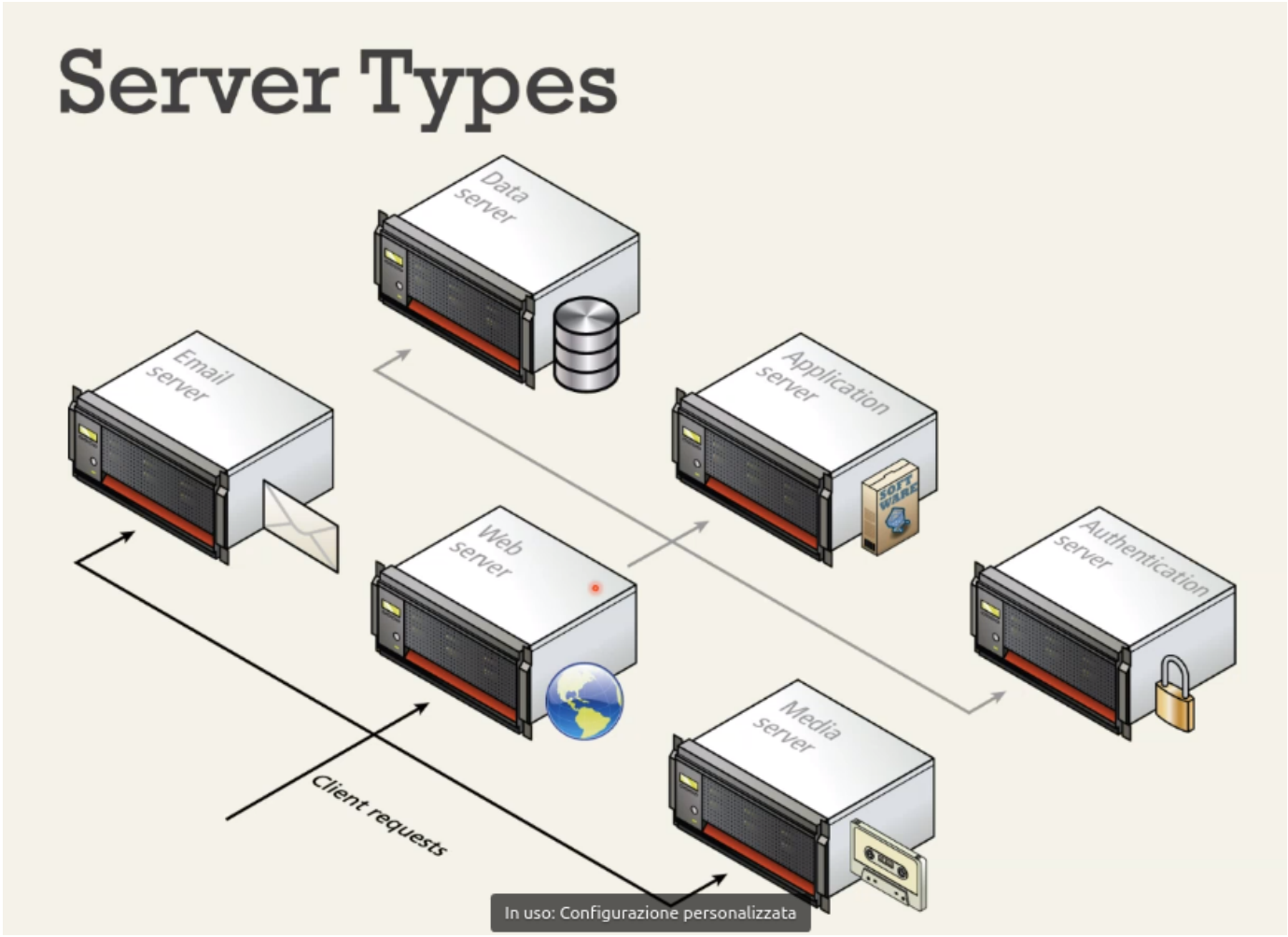
Transport layer: il meccanismo di comunicazione tra processi viene realizzato attraverso il livello transporto. Quelli più utilizzati sono TCP e UDP. Quello interessante per questo corso è il TCP perchè le richieste e le risposte del protocollo HTTP fanno uso delle chiamante TCP sottostanti. Realizza il canale di comunicazione tra un processo e un altro. Attraverso il "canale" creato i dati fluiscono tra i due processi. Il TCP oltre alla comunicazione tra processi (e non più calcolatori) fa anche altro. Il protocollo TCP mette in atto dei meccanismi di recupero per colmare la mancanza del protocollo IP. Fa si che che il canale sia affidabile, assicurandosi la comunicazione corretta. Per identificare con quale processo vogliamo parlare dobbiamo usare la PORTA, un numero intero espresso su 16 bit, che si mette in ascolto sulla porta specificata, e quando arriva una comunicazione dal mondo esterno sarà diretta verso quella porta.

Application layer: il protocollo applicativo per il web è l'HTTP, poi c'è IMAP e POP per la posta elettronica etc...

Server Types

Nel caso in cui l'applicazione web venga adoperata da migliaia di utenti, l'applicazione web verrà replicata su una molteplicità di macchine. Nella seguente immagine le richieste del client verranno gestite da un pool di macchine aventi diversi ruoli Web Server, Mail Server, Database Server,

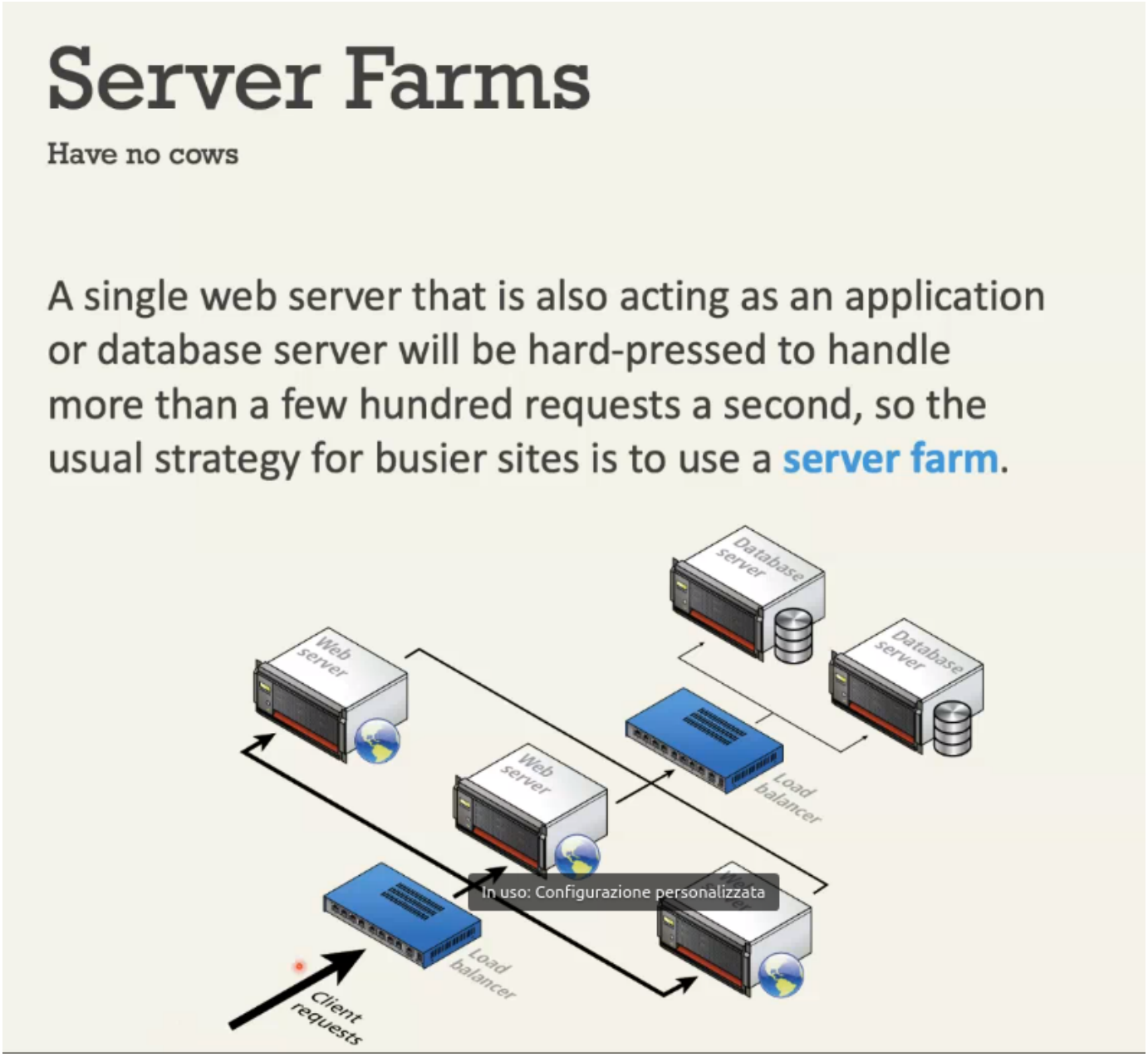
Authentication server...



Server farms

Un singolo calcolatore potrebbe non essere sufficiente a gestire questi volumi di traffico quindi potremmo adoperarne più di uno. Le richieste che arrivano dai client arrivano in modo bilanciato

mediate da un Load Balancer:



La logica con cui le richieste vengono smistate possono essere di diverse tipo (round robin, FIFO, LIFO).

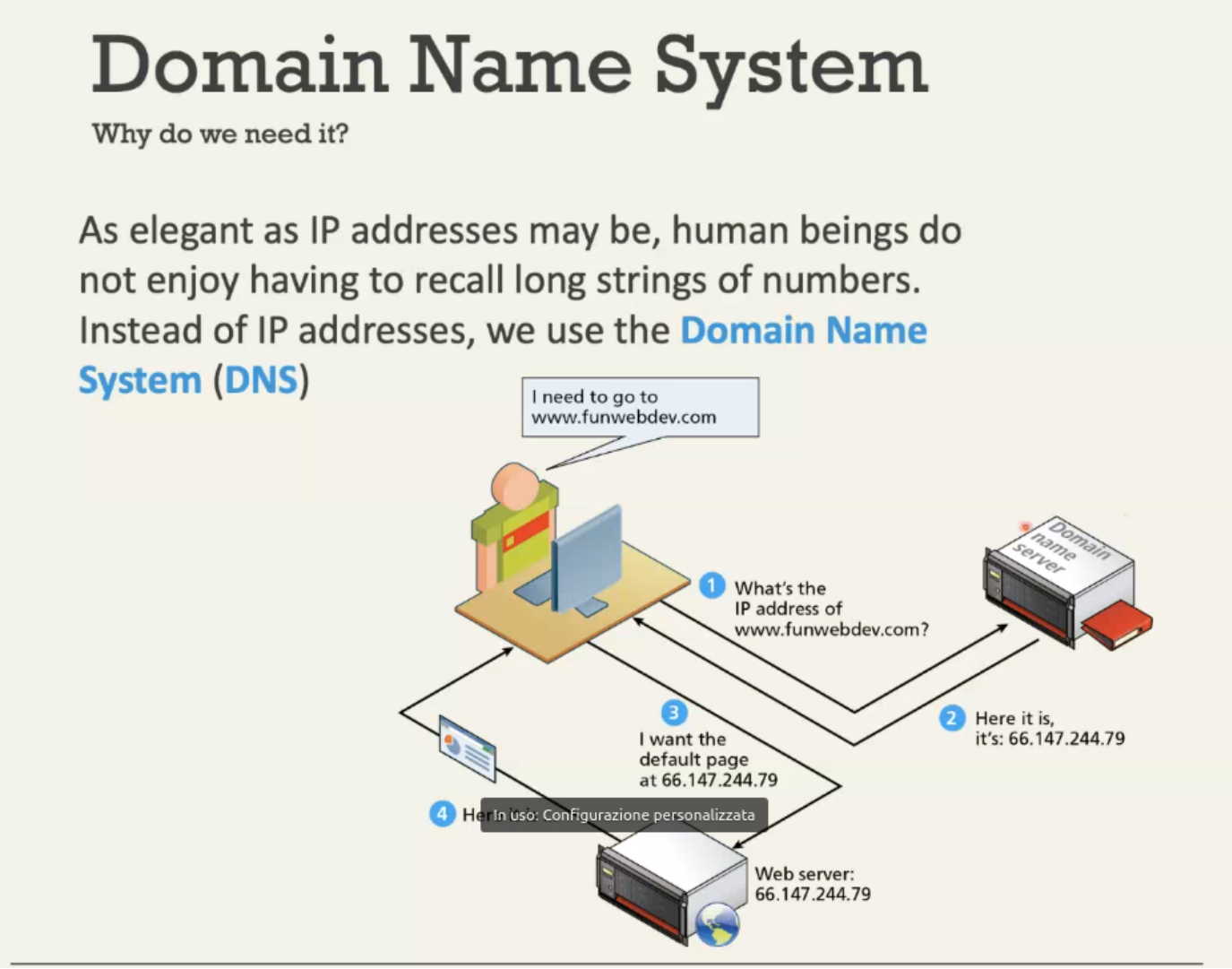
Un load balancer ha anche un effetto positivo in termini di tolleranza ai guasti. Nel caso una macchina si rompa è possibile avere un failover verso i 2 server funzionanti rispetto al terzo, quindi ci sarà un degradamento delle prestazioni ma l'applicazione continua a funzionare.

Data centers & web hosting

Può accadere che più applicazioni girino in contemporanea sulla stessa macchina.
Alcune compagnia di web hosting (GODaddy, Dreamhost, etc...) ospitano il sito web su un unica macchina fisica insieme ad altri siti web. Inutile dedicare una macchina fisica a quel sito web solo per quella specifica applicazione web di quella specifica azienda o persona.

Domain Name System

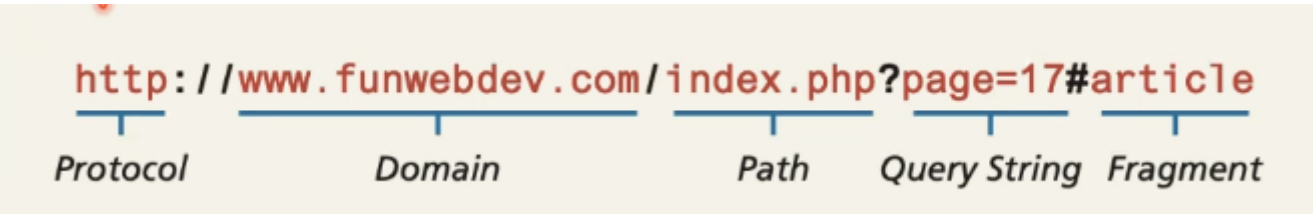
Il protocollo IP si basa su indirizzi IP composti da 32 bit ovvero indirizzi numerici che sono difficili da ricordare ed è per questo che usiamo gli indirizzi simbolici e quindi usiamo un meccanismo di traduzione denominato DNS. Tramite questa slide capiamo solo cosa fa ma non COME lo fa:



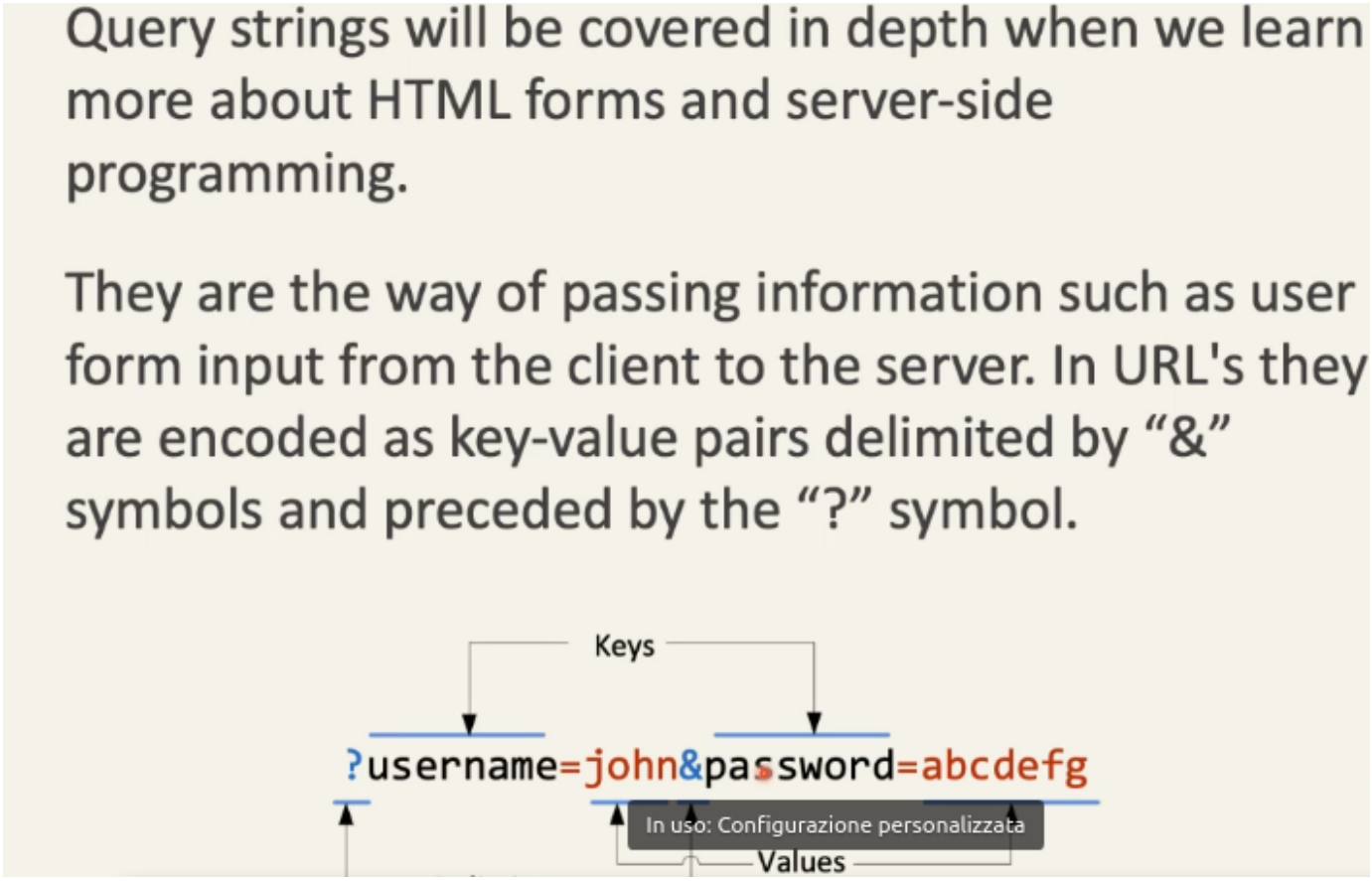
URLs

Uniforme resource locator ovvero il modo in cui identifichiamo una risorsa ovvero un immagine, un file HTML, un contenuto all'interno di una pagina web...

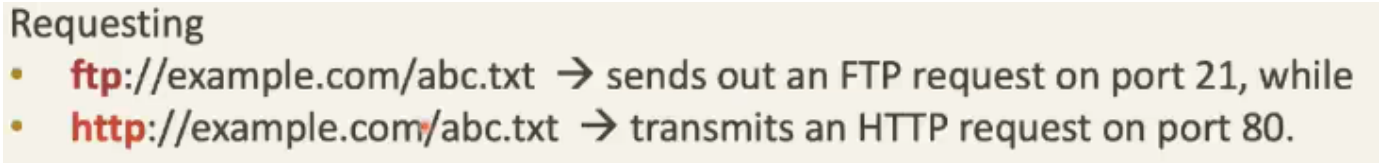
La struttura di un URL è la seguente:



Le query string permettono di passare informazioni provenienti ad esempio dall'utente:

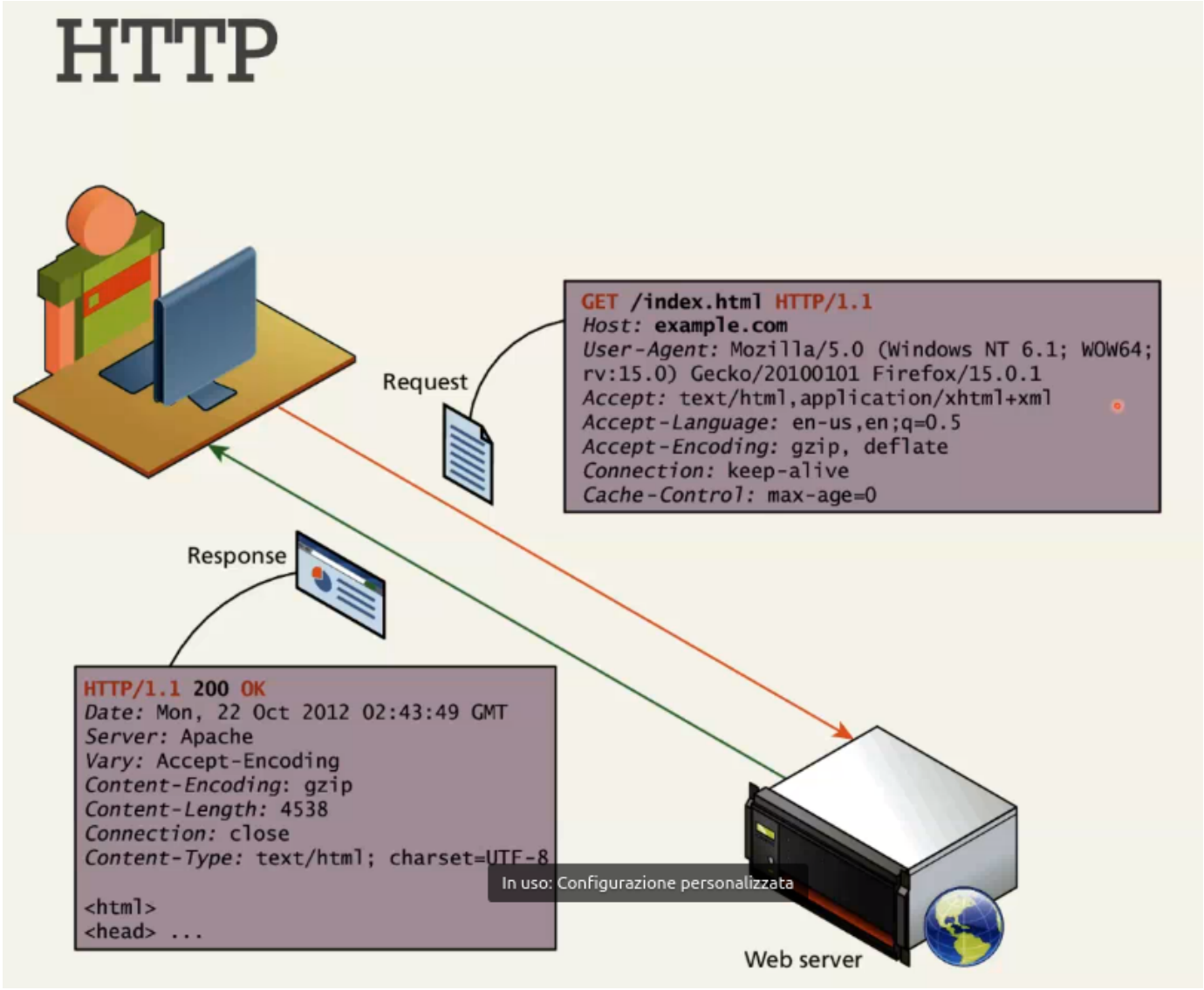


Un URL può fare riferimento anche ad altri protocolli come ad esempio FTP per il trasferimento di file. Il protocollo HTTP implicitamente userà una porta TCP ovvero la 80 (**TCP/80**). Il web server di default si mette in ascolto su questa porta, possiamo anche configurarlo per dirgli di mettersi in ascolto sulla 5000. Il fatto che la porta 80 sia ben nota è utile perchè quando il client fa una richiesta verso il web server in HTTP NON deve specificare la porta 80:



Ad esempio per l'FTP la porta well-known è la 21.

HTTP



Tale protocollo può essere usato anche al di là di questo specifico contesto per far parlare i programmi client con quelli server e quindi usare l'HTTP come protocollo di scambio di informazioni. Non necessariamente il client deve essere un browser e il server un web server.

La richiesta in questo esempio è una richiesta GET, che è una delle richieste possibili, ed è il tipo di richiesta può frequentemente usata. La risorsa richiesta è quella specificata con `/index.html`. Le versioni più importanti di HTTP sono la 1.1, la 1.2 e recentemente la versione 3.0. Alcuni colossi come Google erogano contenuti attraverso HTTP 3.0.
L'HTTP 3.0 rispetto alle precedenti introduce delle differenze sostanziali in quanto non usa il TCP ma il protocollo Quick introdotto da Google e basato su UDP con meccanismi custom di recupero degli errori.

L'header HTTP della REQUEST è quello in viola nella richiesta in cui troviamo delle coppie chiave-valore che ci permettono di customizzare la richiesta verso il server. Stiamo dicendo che la richiesta è diretta all'`example.com` ,quale browser stiamo usando tramite lo `User-Agent`. Con la proprietà `Accept` il client indica che la risorsa che riceverà gli va bene sia in formato di testo html che in formato application xhtml+xml. `Accept-Language` indica che gli va bene sia in inglese americano che in inglese standard. Queste preferenze sono separate da virgola e possiamo esprimere anche qual'è l'ordine con cui vorremmo ottenere la risorsa tramite il `q` che può assumere un valore tra 0.1 e 0.5. Lo stesso vale in termini di compressione dei dati `Accept-Encoding`. Con il campo `Connection` il client

chiede che al termine di questo ciclo richiesta-risposta la connessione TCP venga mantenuta attiva `keep-alive` in modo tale da poterci trasferire sopra nuvoe richieste e risposte. Questo è utile in quanto la creazione della connessione TCP richiede dei tempi e delle risorse, in quanto non sarà necessario ricostruire la connessione. Infine il `Cache-control` serve a specificare l'eventuale funzionamento di meccanismi di cache. Con `max-age=0` la risorsa non verrà cachata.

L'header HTTP della RESPONSE include la versione dell'HTTP e un codice numerico `200 OK` ovvero il server riesce ad evadere la richiesta del client. Il tipo di server che sta rispondendo `Server: Apache` e che la lunghezza della risposta è `Content-Length: 4538`. A seguire c'è il body della risposta contenente l'index.html. Il `Connection: close` nella risposta da parte del server indica che:
I don't support your keep-alive request and will just close the connection when I am finished.