

HTML



Lezione 4:

HTML 5

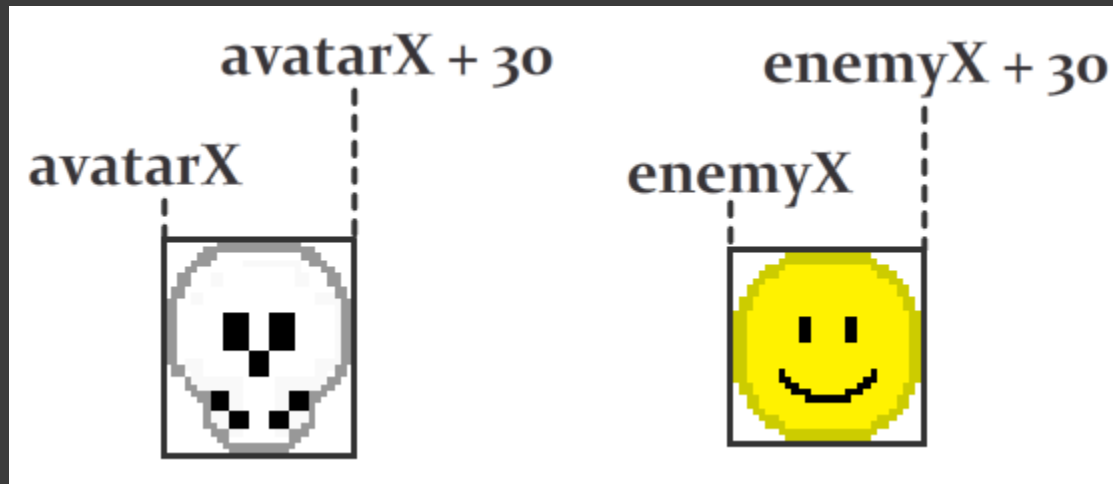
CANVAS

WEBSOCKET

Presentazioni:

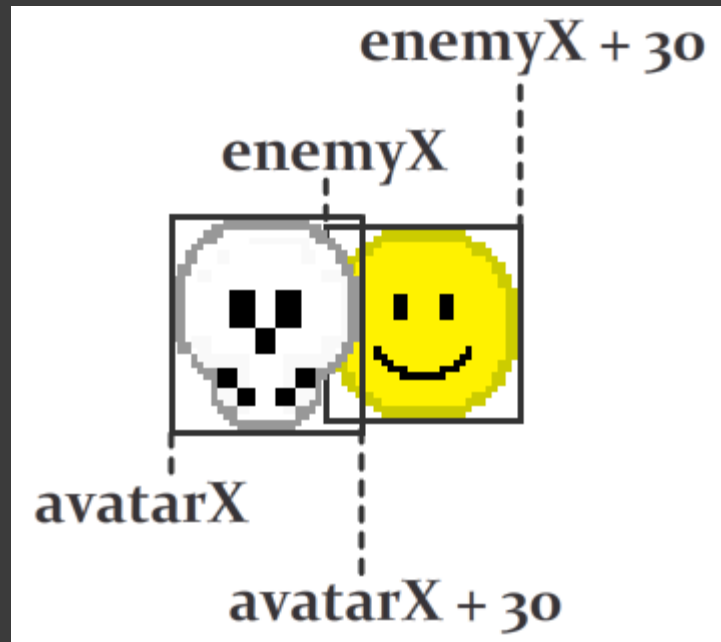
- ◎ Renato Mainetti (Grafica Programmazione Reti)
- ◎ Mail per info: renato.mainetti@gmail.com
(grande fantasia)
- ◎ Sito Web da cui scaricare le slide :
<http://tamberlo.altervista.org>

Collisioni: BoundingBox



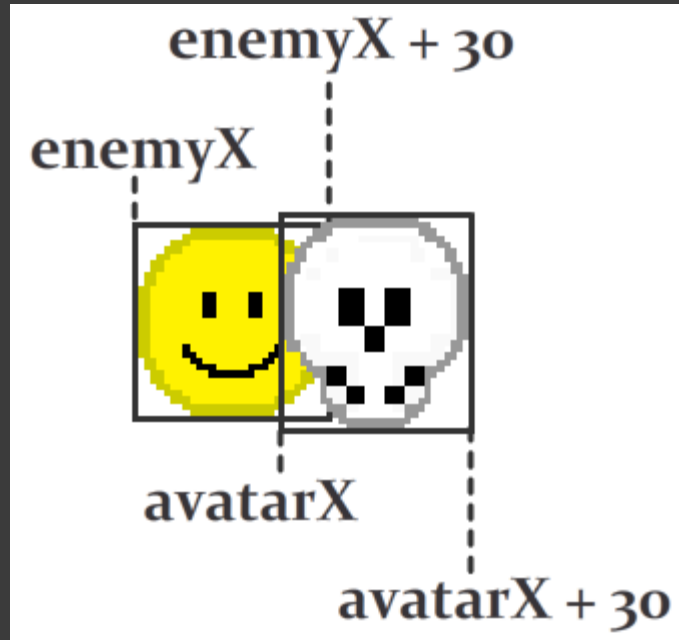
$\text{avatarX} < \text{avatarX} + 30 < \text{enemyX} < \text{enemyX} + 30$

Collisioni: BoundingBox



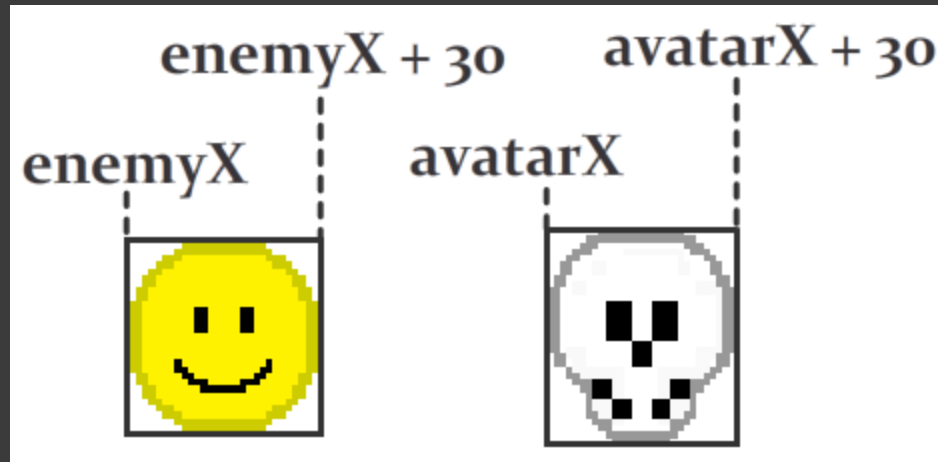
$$\text{avataarX} < \text{enemyX} < \text{avataarX} + 30 < \text{enemyX} + 30$$

Collisioni: BoundingBox



$\text{enemyX} < \text{avatarX} < \text{enemyX} + 30 < \text{avatarX} + 30$

Collisioni: BoundingBox



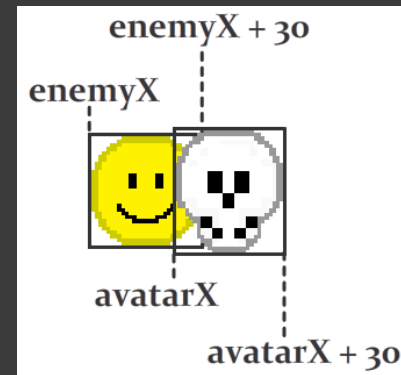
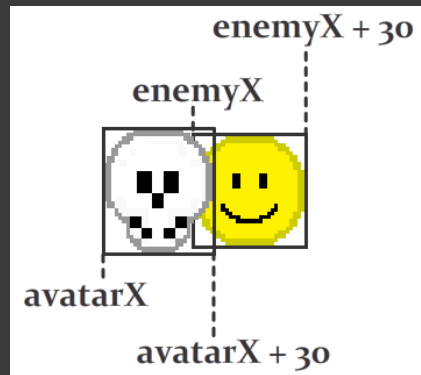
$\text{enemyX} < \text{enemyX} + 30 < \text{avatarX} < \text{avatarX} + 30$

Logica Bool: (Or e AND)

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Collisioni: BoundingBox



(`avatarX < enemyX` **AND** `enemyX < avatarX + 30`)

OR

(`enemyX < avatarX` **AND** `avatarX < enemyX + 30`)

Collisioni: BoundingBox

(avatarY < enemyY **AND** enemyY < avatarY + 30)

OR

(enemyY < avatarY **AND** avatarY < enemyY + 30)

Collisione: Finalmente!

((avatarX < enemyX **AND** enemyX < avatarX + 30)

OR

(enemyX < avatarX **AND** avatarX < enemyX + 30))

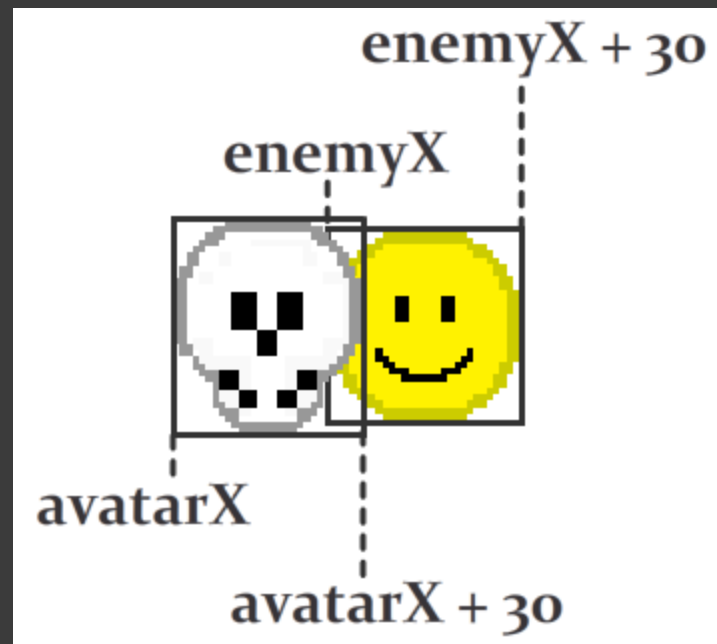
AND

((avatarY < enemyY **AND** enemyY < avatarY + 33)

OR

(enemyY < avatarY **AND** avatarY < enemyY + 30))

Esempio Chrome:

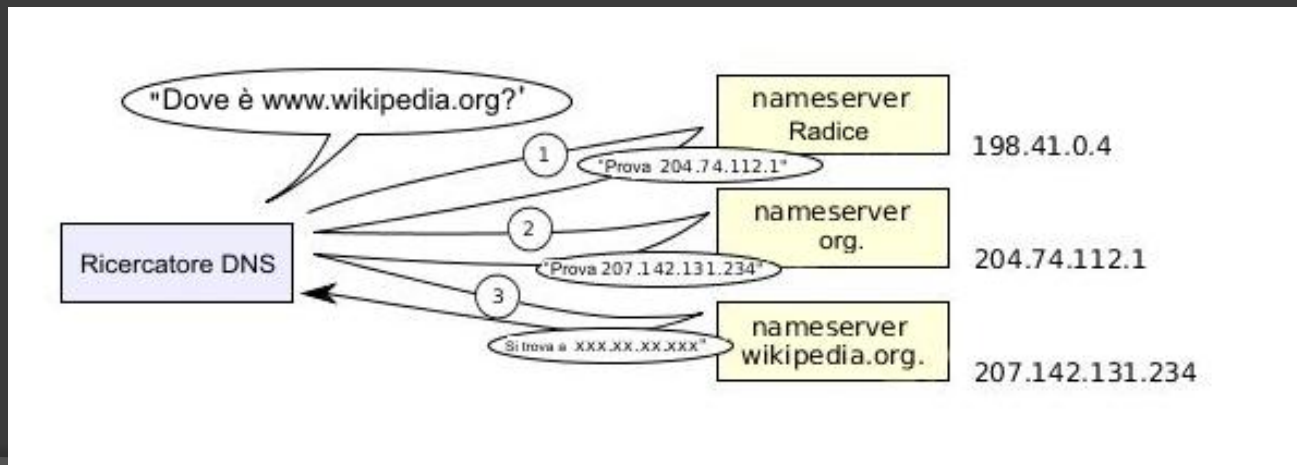


Server e Client: (perchè?)

- 24/7
- Condivisione
- Sincronizzazione
- Velocità(dipende)
- Sicurezza
- Affidabilità

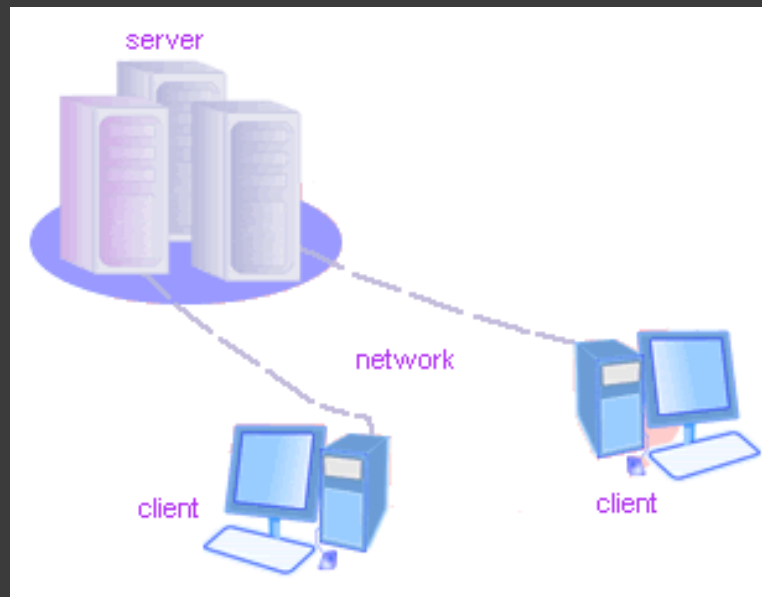
Breve intro sui DNS: (cosa sono?)

In inglese **Domain Name System** (spesso indicato con **DNS**), è un sistema utilizzato per la risoluzione di nomi dei nodì della rete (in inglese host) in indirizzi IP e viceversa. Il servizio è realizzato tramite un database distribuito, costituito dai server DNS.



Server e Client: (questi sconosciuti)

Un'applicazione **client-server** è un tipo di applicazione di rete nel quale un computer **client** istanzia l'interfaccia utente di un'applicazione connettendosi ad una server application o ad un sistema di database.



Server e Client: (come?)

Ciascun server può essere di tipo:

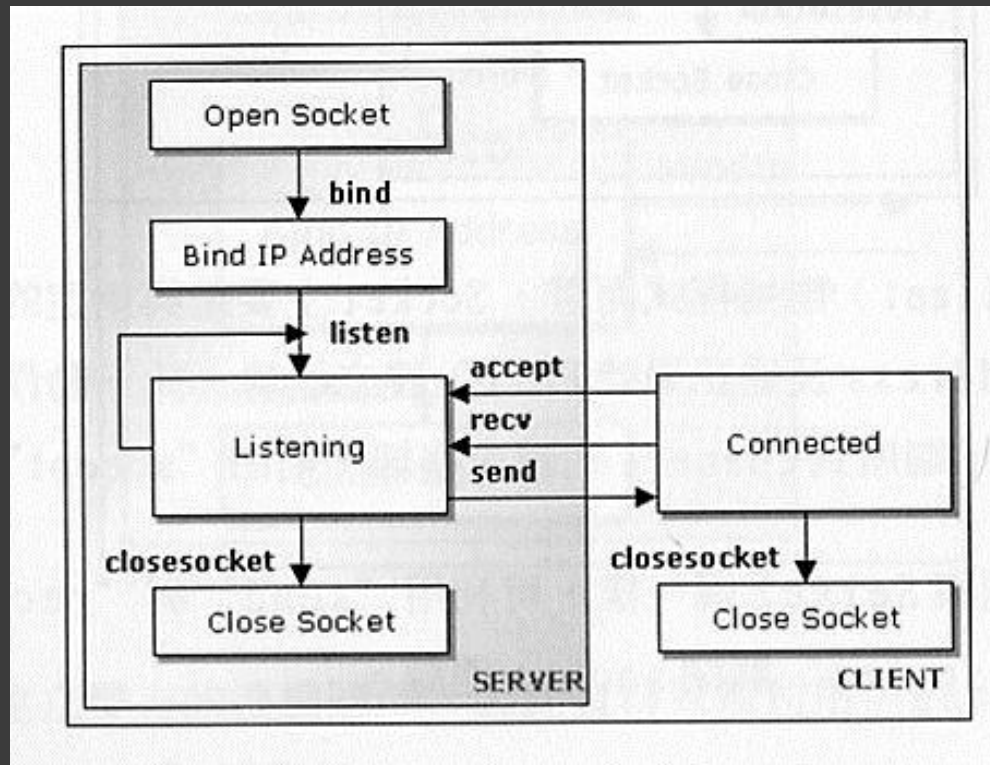
iterativo, cioè accogliere e soddisfare sequenzialmente una sola richiesta di servizio alla volta da parte dei client con una tipica procedura a coda di attesa dei processi da gestire (caso di pochi client: situazione ai primordi della rete Internet);

concorrente, ovvero soddisfare più richieste di servizio da parte di più client attraverso procedure tipiche del multithreading (situazione diventata ora comune nella rete Internet).

Server e Client: (come?)

Tipicamente l'espletamento del servizio per il client è preceduta da una fase di definizione di un socket e la successiva instaurazione della connessione con il server tramite i protocolli di rete TCP e UDP con il server che possiede un indirizzo IP statico a causa dei legami intrinseci con il nome di dominio (che è inevitabilmente fisso) attraverso il DNS. La creazione di applicazioni di rete, per quanto riguarda la connessione, ricade all'interno degli ambiti della cosiddetta programmazione socket. Tipicamente tale ambito di programmazione fa uso di opportune chiamate di sistema o API Socket al sistema operativo del server e del client per realizzare la connessione affidabile sia in modalità iterativa sia concorrente gestendo anche tutti i possibili errori o eccezioni.

Server e Client: (come?)



A **socket address** is the combination of an IP address (the location of the computer) and a port (which is mapped to the application program process) into a single identity.

IP Statici per forza?



Anche NO! 😊

Passato, Presente e Futuro:



Polling, Long-Polling, and Streaming

Vs.

WebSocket

Polling, Long-Polling, and Streaming: Headache 2.0

Quando un browser richiede una pagina web, viene effettuata una richiesta HTTP al WebServer che ospita quella pagina.

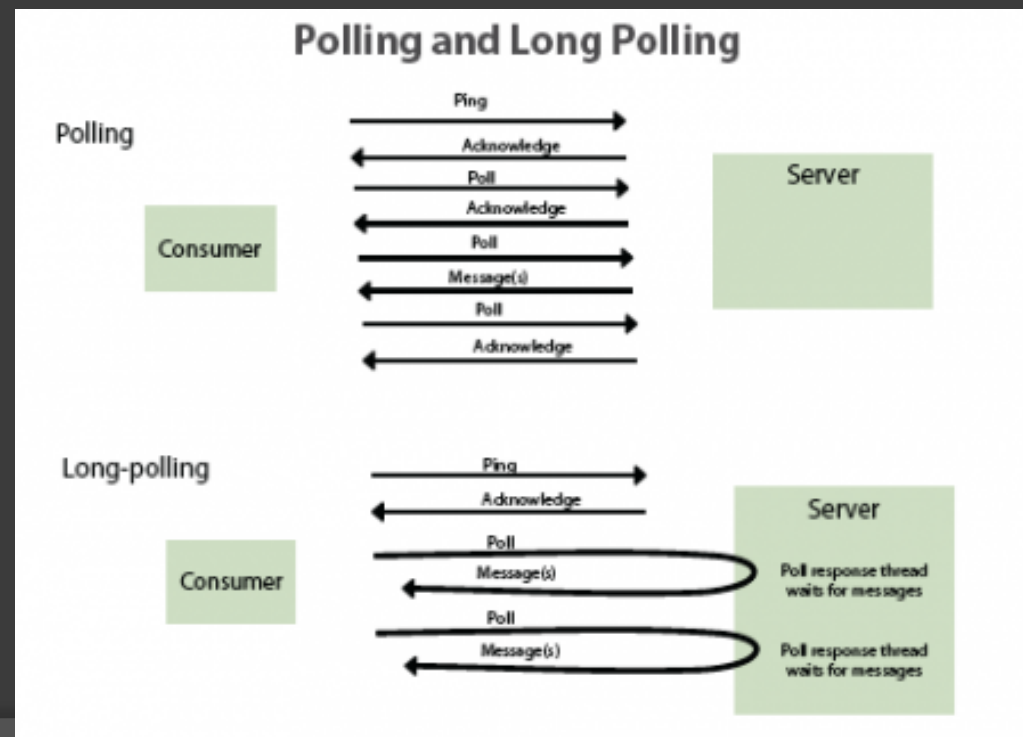
Il server accetta la richiesta e rispedisce al client la pagina. **Per info Realtime...refresh**



Polling, Long-Polling, and Streaming: Headache 2.0

Per info Realtime...refresh! Ma se non vogliamo fare refresh? Magari non ci sono nuovi dati...

[http://en.wikipedia.org/wiki/Comet_\(programming\)](http://en.wikipedia.org/wiki/Comet_(programming))



Polling, Long-Polling, and Streaming: Headache 2.0

- ◉ With *polling*, the browser sends HTTP requests at regular intervals and immediately receives a response. This technique was the first attempt for the browser to deliver real-time information. Obviously, this is a good solution if the exact interval of message delivery is known, because you can synchronize the client request to occur only when information is available on the server. However, real-time data is often not that predictable, making unnecessary requests inevitable and as a result, many connections are opened and closed needlessly in low-message-rate situations.
- ◉ With *long-polling*, the browser sends a request to the server and the server keeps the request open for a set period. If a notification is received within that period, a response containing the message is sent to the client. If a notification is not received within the set time period, the server sends a response to terminate the open request. It is important to understand, however, that when you have a high message volume, long-polling does not provide any substantial performance improvements over traditional polling. In fact, it could be worse, because the long-polling might spin out of control into an unthrottled, continuous loop of immediate polls.
- ◉ With *streaming*, the browser sends a complete request, but the server sends and maintains an open response that is continuously updated and kept open indefinitely (or for a set period of time). The response is then updated whenever a message is ready to be sent, but the server never signals to complete the response, thus keeping the connection open to deliver future messages. However, since streaming is still encapsulated in HTTP, intervening firewalls and proxy servers may choose to buffer the response, increasing the latency of the message delivery. Therefore, many streaming Comet solutions fall back to long-polling in case a buffering proxy server is detected.

HTML5 Web Sockets to the Rescue!

Defined in the Communications section of the HTML5 specification, HTML5 Web Sockets represents the next evolution of web communications—a full-duplex, bidirectional communications channel that operates through a single socket over the Web. HTML5 Web Sockets provides a true standard that you can use to build scalable, real-time web applications

HTML5 Web Sockets to the Rescue!

WebSocket

is a technology providing for bi-directional, [full-uplex](#) communications channels, over a single [Transmission Control Protocol](#) (TCP) [socket](#).

It is designed to be implemented in [web browsers](#) and [web servers](#), but it can be used by any client or server application. The WebSocket API is being standardized by the [W3C](#), and the WebSocket protocol is being standardized by the [IETF](#)

For the client side, WebSocket was implemented in [Firefox 4](#), [Google Chrome 4](#), [Opera 11](#), and [Safari 5](#), as well as the mobile version of Safari in [iOS 4.2](#).^[1] Also, the BlackBerry Browser in OS7 supports WebSocket.^[2] However, although present, support was disabled by default in Firefox 4 and 5 and Opera 11 because of concerns over security vulnerabilities.^{[3][4]} The new -07 version of the WebSocket protocol, which fixes the protocol bug, is implemented and enabled by default in Firefox 6 ^[5] and in Chrome 14.^[6]

Simple-Half-FullDuplex? WTF?

- **Simplex**
In un canale di trasmissione *Simplex* i dati viaggiano al suo interno solo e sempre in un'unica direzione.
- **Half-duplex**
In un canale di trasmissione *Half-duplex* invece i dati possono passare in entrambi i sensi ma solo alternativamente. Ciò significa che mentre il trasmettore (Tx) manda dei dati, il ricevitore (Rx) sta ad ascoltare. Un canale half-duplex è bidirezionale in quanto Tx e Rx possono scambiarsi i ruoli ma ricordando comunque che non lo possono fare contemporaneamente.
Una analogia, giusto per capire, potrebbe essere quella di due radioamatori: mentre uno sta parlando l'altro ascolta; al contrario, quando quest'ultimo risponderà sarà l'altro ad ascoltare e via dicendo.
- **Full-duplex**
Il *Full-duplex* invece, come vi aspetterete, è in grado di mandare e ricevere dati in entrambi i sensi e simultaneamente. E' ovviamente il tipo di canale di trasmissione piu' usato nella stra grande maggioranza dei servizi del giorno d'oggi.

HTML5 Web Sockets to the Rescue!

To establish a WebSocket connection, the client and server upgrade from the HTTP protocol to the WebSocket protocol during their initial handshake, as shown in the following example:

WebSocket handshake (browser request and server response)

GET /text HTTP/1.1

Upgrade: WebSocket

Connection: Upgrade

Host: www.websocket.org

HTTP/1.1 101 WebSocket Protocol Handshake

Upgrade: WebSocket

Connection: Upgrade

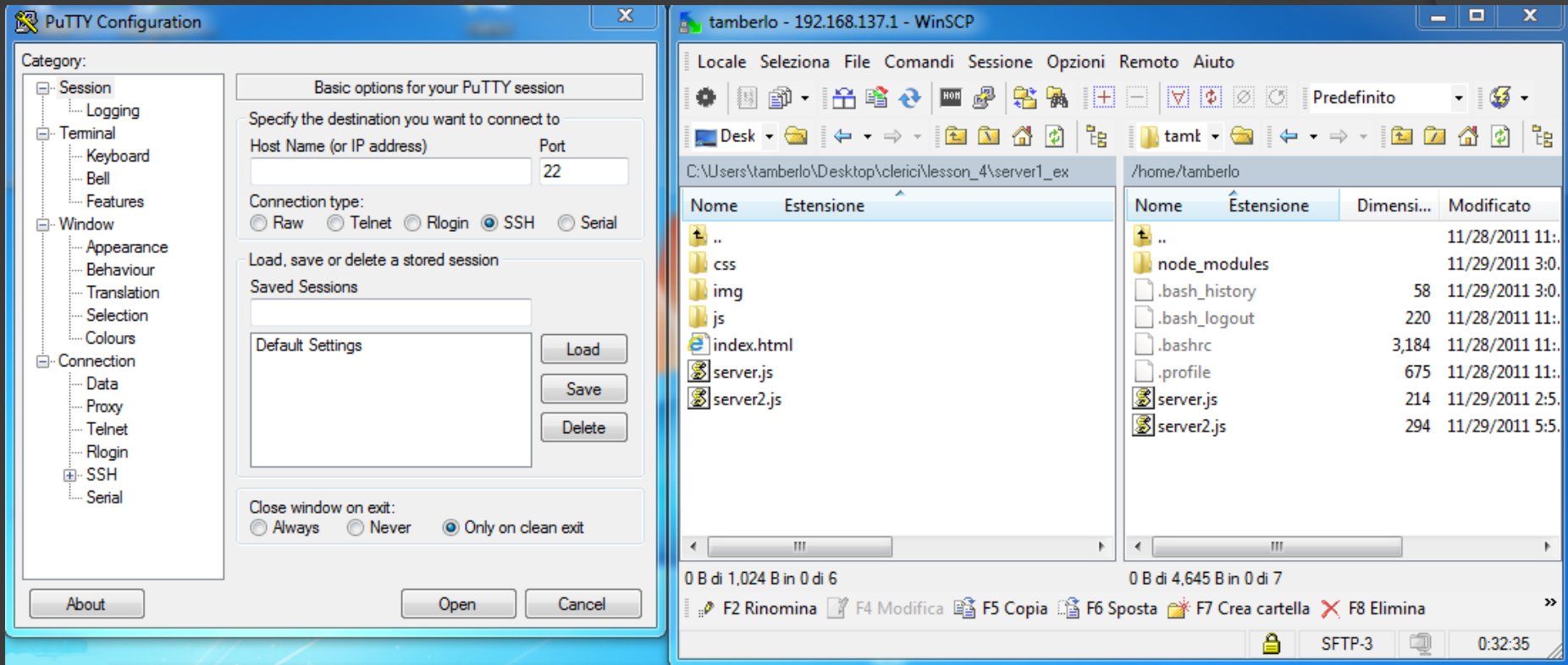
SSH:

SSH (Secure SHell, shell sicura)

è un protocollo di rete che permette di stabilire una sessione remota cifrata tramite interfaccia a riga di comando con un altro host di una rete informatica.

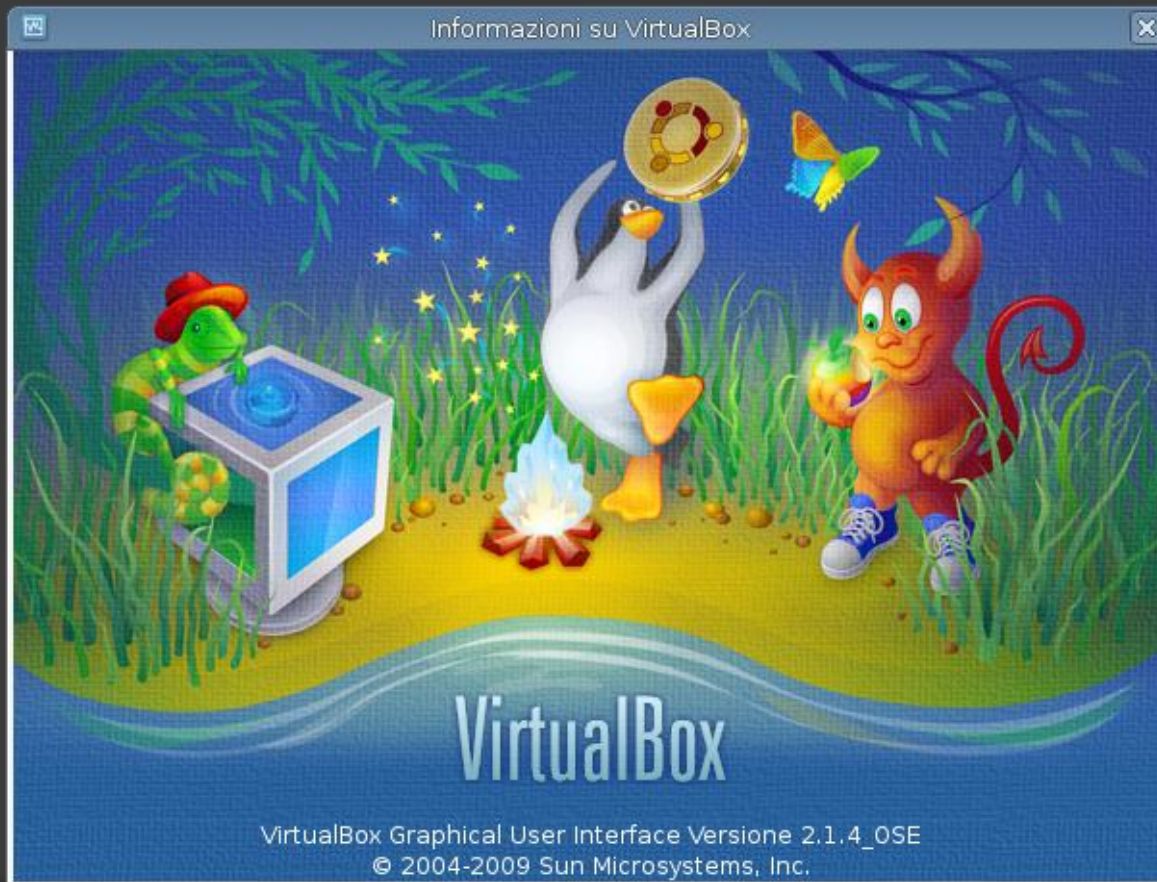
E' il protocollo che ha sostituito l'analogo ma insicuro Telnet.

Putty & SCP (SSH) :



Virtual-Box (Oracle):

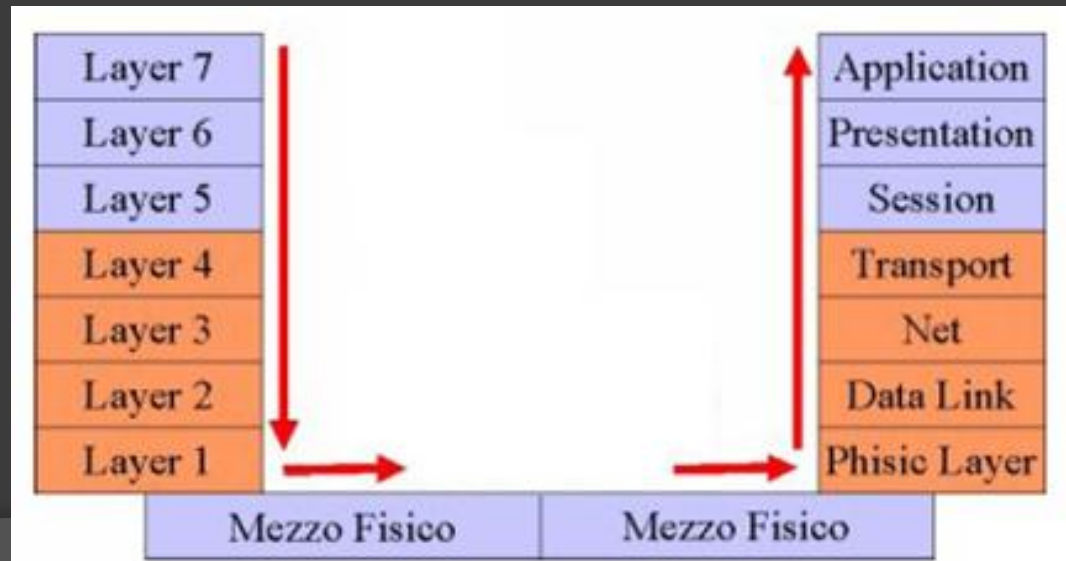
<https://www.virtualbox.org/>



Esempio...

Debian e Port Forwarding: NAT

il **network address translation** o **NAT**,
ovvero *traduzione degli indirizzi di rete*
Il **NAT** è una tecnica che consiste nel
modificare gli indirizzi IP dei pacchetti in
transito su un sistema che agisce
da router.



Socket IO:



What is Socket.IO?

Socket.IO aims to make realtime apps possible in every browser and mobile device, blurring the differences between the different transport mechanisms. It's care-free realtime 100% in JavaScript.

- Cenno alle collisioni con semplice esempio funzionante chrome
- Definizione server client schemi etc etc
- DNS cosa sono...
- Spiegazione socket e accenno ai protocolli usati
- Spiegazione websocket differenza col vecchio modo ajax xmlhttprequest
- Macchina virtuale spiegazione
- Ssh spiegazione
- Putty SCP example
- Inoltro porte perchè debian sotto nat
- Pacchetto socket.io
- Creiamo un server in javascript semplice
- Facciamolo girare su debian
- Creiamo un client semplice
- Colleghiamoci tutti al server debian e vediamo il risultato

Domande?
Chiarimenti?
Proposte?