



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Programmazione di Reti

Laboratorio 2

Andrea Piroddi

Dipartimento di Informatica, Scienza e Ingegneria

Programmazione di un Socket - Web Server



Socket

Un **socket** è un ***oggetto software*** che **permette l'invio e la ricezione di dati, tra host remoti** (tramite una rete) **o tra processi locali**.

Più precisamente, il concetto di **socket** si basa sul modello Input/Output su file di Unix, quindi sulle operazioni di ***open, read, write e close***; l'utilizzo, infatti, avviene secondo le stesse modalità, aggiungendo i parametri utili alla comunicazione, quali **indirizzi ip, numeri di porta e protocolli**.

Socket locali e remoti in comunicazione formano una **coppia** (pair), composta da **indirizzo e porta** di client e server; tra di loro c'è una connessione logica. Solitamente i sistemi operativi forniscono delle API per permettere alle applicazioni di controllare e utilizzare i socket di rete.



Socket

1. Creazione dei socket

Client e server creano i loro rispettivi **socket**, e il **server** lo pone in **ascolto** su una **porta**. Dato che il server può creare più connessioni con client diversi (ma anche con lo stesso), ha bisogno di una **coda** per gestire le varie richieste.

2. Richiesta di connessione

Il **client** effettua una **richiesta di connessione** verso il server.

Da notare che possiamo avere due numeri di porta diversi, perchè una potrebbe essere dedicata solo al traffico in uscita, l'altra solo in entrata; questo dipende dalla configurazione dell'host.

Il **server** riceve la richiesta e, nel caso in cui sia accettata, viene creata una **nuova connessione**.

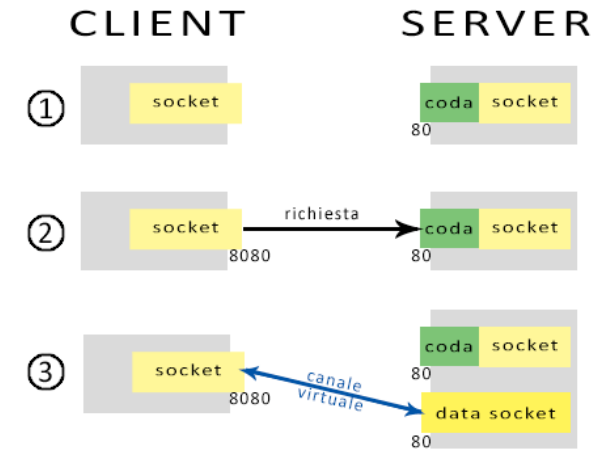
3. Comunicazione

Ora client e server comunicano attraverso un **canale virtuale**, tra il socket del primo, ed uno nuovo del server, creato appositamente per il flusso dei dati di questa connessione: **data socket**.

Coerentemente a quanto avete visto nella teoria, il server crea il data socket perchè il primo serve esclusivamente alla gestione delle richieste. È possibile, quindi, che ci siano molti client a comunicare con il server, ciascuno verso il **data socket** creato dal server per loro.

4. Chiusura della connessione

Essendo il TCP un protocollo orientato alla connessione, quando non si ha più la necessità di comunicare, il client lo comunica al server, che ne **deistanzia il data socket**. La connessione viene così chiusa.



Famiglie di socket

I tipi di **protocolli** utilizzati dal **socket**, ne definiscono la **famiglia** (o dominio).

Possiamo distinguere, ad esempio, due importanti famiglie:

- ***AF_INET***: comunicazione tra host remoti, tramite Internet;
- ***AF_UNIX***: comunicazione tra processi locali, su macchine Unix. Questa famiglia è anche chiamata *Unix Domain Socket*.



Esercizio 1: Programmazione di un Socket - Web Server

In questo laboratorio vediamo le basi della programmazione dei socket per le connessioni TCP in Python:

- come creare un socket
- associarlo a un indirizzo e una porta specifici,
- nonché inviare e ricevere un pacchetto HTTP.

Vedremo anche alcune nozioni di base sul formato dell'intestazione HTTP. Svilupperemo un server web che gestisce una richiesta HTTP alla volta.



- Il server Web deve accettare e analizzare la richiesta HTTP
- ottenere il file richiesto dal file system del server
- creare un messaggio di risposta HTTP costituito dal file richiesto preceduto da righe di intestazione e quindi inviare la risposta direttamente al client.
- Se il file richiesto non è presente nel server, il server deve inviare un messaggio **HTTP "404 non trovato"** al client.



Esercizio 1: Programmazione di un Socket - Web Server

Su IOL trovate nel Laboratorio il codice Python:

«Socket_Programming_Assignment_1.py» e un file «index.html».

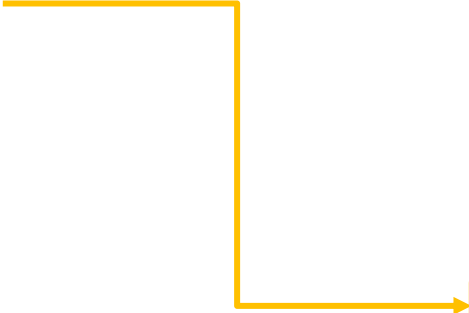
Nome	Ultima modifica	Tipo	Dimensione
 index	19/03/2020 22:18	Chrome HTML Do...	1 KB
 Socket_Programming_Assignment_1	19/03/2020 22:25	Python File	2 KB

Scaricate i due file (***index.html*** e ***Socket_Programming_Assignment_1***) sul vostro PC e metteteli nella stessa directory.




Esercizio 1: Programmazione di un Socket - Web Server

Aprirete con SPYDER il file Python e modificate il valore di porta all'interno del file python (esempio **serverPort=8080**)



```
1  # Corso di Programmazione di Reti - Laboratorio - Università di Bologna
2  # Socket_Programming_Assignment - WebServer - G.Pau - A. Piroddi
3
4
5  from socket import * #importiamo la <class 'type'> invece dell'intero modulo
6
7  # assegniamo alla porta del nostro server un valore a piacere es:82
8  serverPort=8080
9  # crea un socket INET di tipo STREAM
10 serverSocket = socket(AF_INET, SOCK_STREAM)
11 # associa il socket alla porta scelta
12 serverSocket.bind('',serverPort)
13
14 #listen(1) Definisce la lunghezza della coda di backlog, ovvero il numero di connessioni in entrata che sono state
15 #ma non ancora accettate dall'applicazione.
16 #Non ha nulla a che fare con il numero di connessioni simultanee che il server può gestire.
17 serverSocket.listen(1)
18
19
20 print ('the web server is up on port:',serverPort)
```

ed eseguite il codice



```
In [2]: runfile('/Users/apirodd/OneDrive - Alma Mater Studiorum Università di
Bologna/programmazione di reti/Lezioni/Laboratorio 4/codice python Laboratorio 4/
Socket_Programming_Assignment_1.py', wdir='/Users/apirodd/OneDrive - Alma Mater
Studiorum Università di Bologna/programmazione di reti/Lezioni/Laboratorio 4/
codice python Laboratorio 4')
the web server is up on port: 8080
Ready to serve...
```



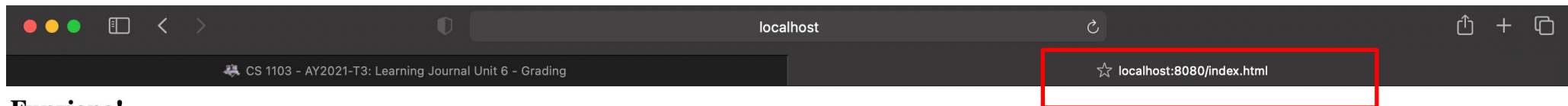
Esercizio 1: Programmazione di un Socket - Web Server

Determinare l'indirizzo IP dell'host che esegue il server (ad es. se è il vostro PC potete considerare come IP «localhost» o 127.0.0.1 indirizzo ip della loopback).

Aprire un browser sulla vostra macchina e digitate nella URL

http://localhost:8080/index.html

Dovrebbe aprirsi la pagina sotto



Funziona!

Corso di PROGRAMMAZIONE di RETI.

Laboratorio 4.



Esercizio 1: Programmazione di un Socket - Web Server

oppure determinate l'IP address associato alla vostra scheda di rete, e da un altro host, aprite un browser e inserite l'URL corrispondente ossia:

http:// «ipaddress-del-vostro-pc»:8080/index.html

«index.html» è il nome del file che avete inserito nella directory del server.

Notate anche l'uso del numero di porta dopo i due punti.

È necessario sostituire questo numero di porta con qualsiasi porta utilizzata nel codice del server. Nell'esempio sopra, abbiamo usato il numero di porta 8080. Il browser dovrebbe quindi visualizzare i contenuti di index.html. Se si omette ": 8080", il browser assumerà la porta 80 e si otterrà la pagina Web dal server solo se il server è in ascolto sulla porta 80.

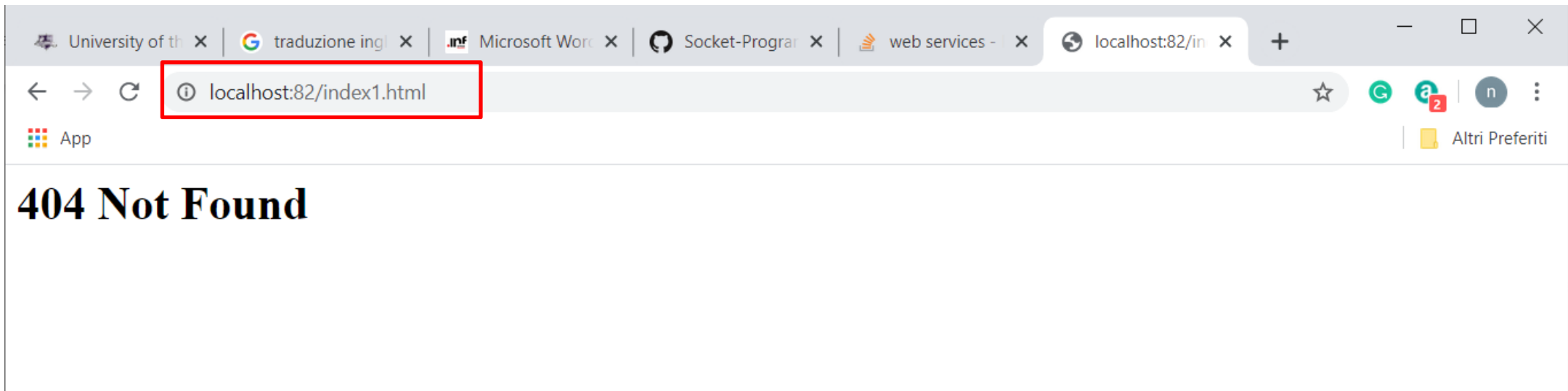


Esercizio 1: Programmazione di un Socket - Web Server

Quindi provate a ottenere un file che non è presente sul server.

Ossia provate a scrivere nella URL al posto di *index.html* il nome *index1.html*.

Dovreste ricevere un messaggio "404 Not Found".



Esercizio 1: Programmazione di un Socket - Web Server

Nel file ***Socket_Programming_Assignment_1*** ogni step è opportunamente commentato in modo da spiegare ogni singolo passaggio logico.

Analizzate il file e poi provate a fare il seguente esercizio:

Invece di utilizzare un browser, scrivete il vostro client HTTP per testare il server. Il client si connetterà al server utilizzando una connessione TCP, invierà una richiesta HTTP al server e visualizzerà la risposta del server come output. Si può presumere che la richiesta HTTP inviata sia un metodo GET.

Il client deve accettare gli argomenti della riga di comando specificando l'indirizzo IP o il nome host del server, la porta su cui è in ascolto il server e il percorso in cui l'oggetto richiesto è archiviato sul server. Di seguito è riportato un formato del comando di input per eseguire il client:

```
>>>client.py server_host server_port filename nome_file
```



Cosa dovreste vedere

Lato client

```
Prompt dei comandi
Microsoft Windows [Versione 10.0.18363.720]
(c) 2019 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\apirodd>f:

F:\>python client.py 127.0.0.1 82 index.html
```

```
Prompt dei comandi - python client.py 127.0.0.1 82 index.html
Microsoft Windows [Versione 10.0.18363.720]
(c) 2019 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\apirodd>f:

F:\>python client.py 127.0.0.1 82 index.html
b'HTTP/1.1 200 OK\r\n\r\n'
Input Host Address:
```

Lato server

```
*Python 3.8.0 Shell*
File Edit Shell Debug Options Window Help
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: F:\Socket_Programming_Assignment_1.py =====
the web server is up on port: 82
Ready to serve...
|
```

```
*Python 3.8.0 Shell*
File Edit Shell Debug Options Window Help
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: F:\Socket_Programming_Assignment_1.py =====
the web server is up on port: 82
Ready to serve...
b'GET /index.html HTTP/1.1' :: b'GET' : b'/index.html'
b'/index.html' || b'index.html'
<html><body><h1>FUNZIONA!</h1>
<p>Corso di PROGRAMMAZIONE di RETI</p>
<p>Esercitazione 2</p>
</body></html>
```



DNS



Domain Name System

DNS Protocol	
Purpose	Resolve hostnames by returning IP addresses
Standard	RFC 1034 and RFC 1035
Runs atop	UDP/IP and TCP/IP
Port number	53
Libraries	dnspython (<i>dnspython3</i> for python 3.x)



Cosa è il DNS ?

“Il **Domain Name System, o DNS**, è uno degli elementi costitutivi fondamentali di Internet.

È il database di informazioni, globale, gerarchico e distribuito che è responsabile della traduzione dei nomi in indirizzi e viceversa, dell'instradamento della posta alla destinazione corretta e di molti altri servizi.

Il Domain Name System (DNS) è lo schema mediante il quale milioni di host Internet collaborano per rispondere alla domanda su come il nome dell'host si risolve in indirizzi IP.

Il DNS è il meccanismo che traduce nomi come **iana.org** in indirizzi IPv4 come 192.0.43.8 o 2001:500:88:200::8 se si sta già utilizzando IPv6.

```
verax@untamed ~ $ host iana.org
iana.org has address 192.0.43.8
iana.org has IPv6 address 2001:500:88:200::8
iana.org mail is handled by 10 pechora1.icann.org.
iana.org mail is handled by 10 pechora7.icann.org.
iana.org mail is handled by 10 pechora3.icann.org.
```



Terminologia DNS

RESOLVER (Risolutore)

Resolver è la parte *client* del sistema client/server DNS: pone le domande sui nomi degli host (hostnames).

Il **Resolver** è di solito una piccola libreria compilata in un qualsiasi linguaggio di programmazione che richiede i servizi DNS, e conosce quel tanto che basta per inviare query a un nameserver vicino.

I **Resolver** sono generalmente molto semplici, e si affidano ai server per fare il lavoro pesante.



Terminologia DNS

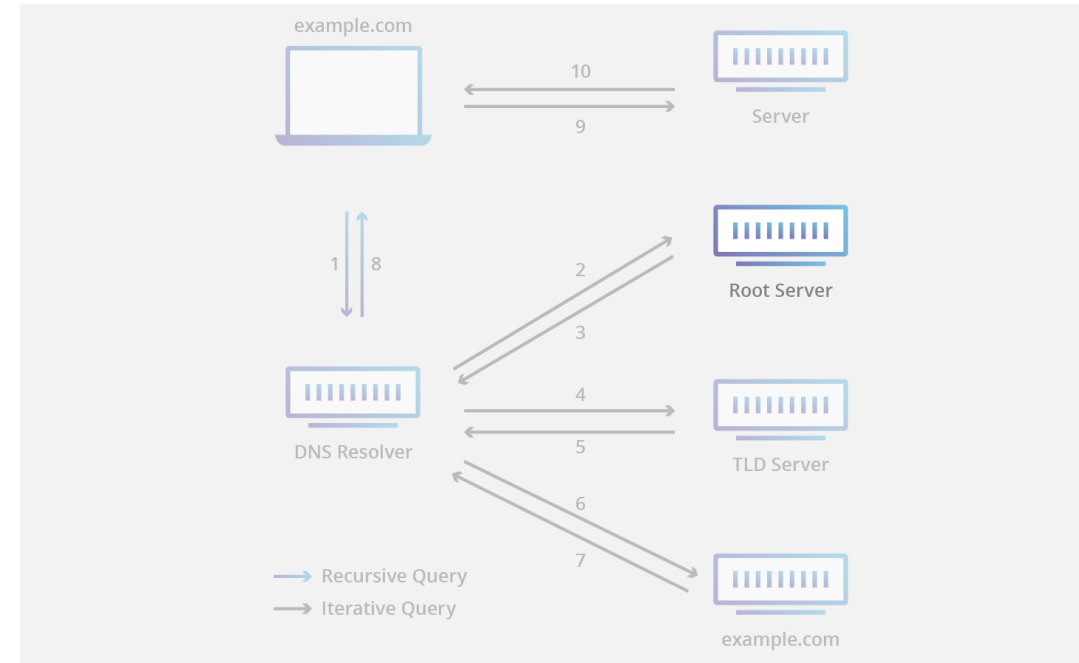
Recursive Nameserver

Un resolver ricorsivo (noto anche come DNS recursor) è la prima fermata in una query DNS.

Il resolver ricorsivo funge da intermediario tra un client e un nameserver DNS.

Dopo aver ricevuto una query DNS da un client Web, un resolver ricorsivo risponderà con i dati memorizzati nella cache o invierà una richiesta a un root nameserver, seguita da un'altra richiesta a un TLD nameserver e quindi un'ultima richiesta a un nameserver autoritativo.

Dopo aver ricevuto una risposta dal nameserver autoritativo contenente l'indirizzo IP richiesto, il resolver ricorsivo invia quindi una risposta al client.



Terminologia DNS

Nameserver

Questo è un server software che risponde alle query DNS.

A volte un nameserver conosce direttamente la risposta (se è "autoritativo" per la zona), altre volte deve andare su Internet e chiedere in giro per trovare la risposta (se è un nameserver ricorsivo).

Esiste un'ampia varietà di software che esegue questo servizio: BIND, PowerDNS ecc.



Terminologia DNS

Name server Autoritativo

Per ogni zona, qualcuno deve mantenere un file con l'associazione hostname e indirizzi IP.

Questa è generalmente una funzione amministrativa eseguita da un essere umano e nella maggior parte dei casi una macchina ha questo file.

È il ZONE MASTER.



Terminologia DNS

Registro delle risorse

Sebbene si pensi che il DNS fornisca solo la mappatura da nome host a IP, in realtà ci sono altri tipi di query che possiamo porre a un DNS, e questo evidenzia l'idea che il DNS sia in realtà un database di "record di risorse".

Il tipo più comune è l'indirizzo IP (un record "A"), ma esistono anche altri record: NS (nameserver), MX (mail exchanger), SOA (Start of Authority) e così via.

Record type	Purpose
A	IP Address record. Using a hostname to get an IPv4 adress.
AAAA	IP Address record. Using a hostname to get an IPv6 adress.
PTR	reverse DNS lookup. Using IP address to get hostname.
NS	Nameserver record responsible for the domain asked about.
MX	Mail Exchanger record. server responsible for handling email for the given domain.
SOA	Start of Authorities record describes some key data about the zone as defined by the zone administrator.
CNAME	Canonical Name or Alias, this allows providing an alternate name for a resource.
TXT	A generic Text record that provides descriptive data about domain.

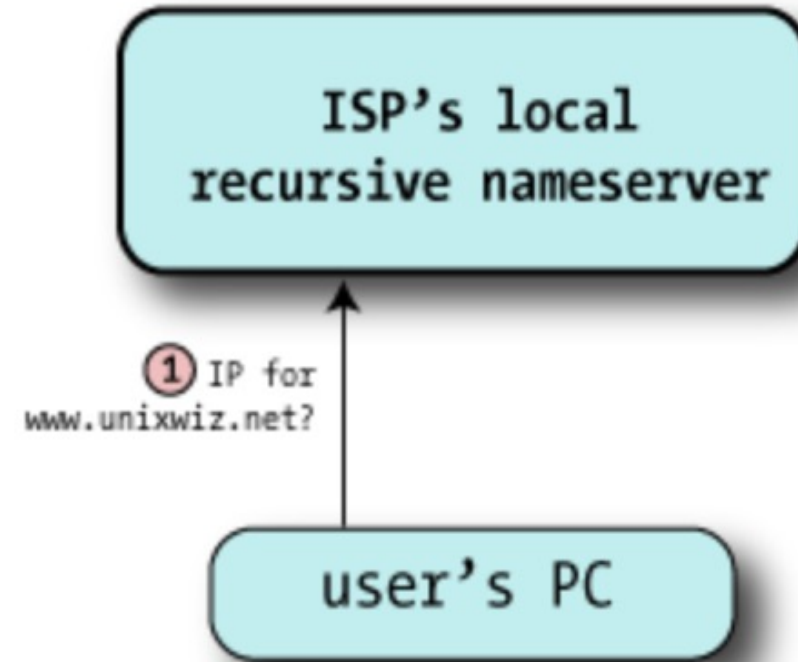


DNS - Query flow

Passo I

Il vostro sistema operativo tenta di risolvere l'indirizzo localmente (utilizzando /etc/hosts su Linux, cercando nella cache locale ecc.).

Se la risposta non è disponibile localmente, viene effettuata una richiesta al RECURSIVE SERVER.



DNS - Query flow

Passo II

Il RECURSIVE SERVER controlla la sua cache, se non trova il record, il server ricorsivo effettua una richiesta per vostro conto a uno qualsiasi dei 13 root server.

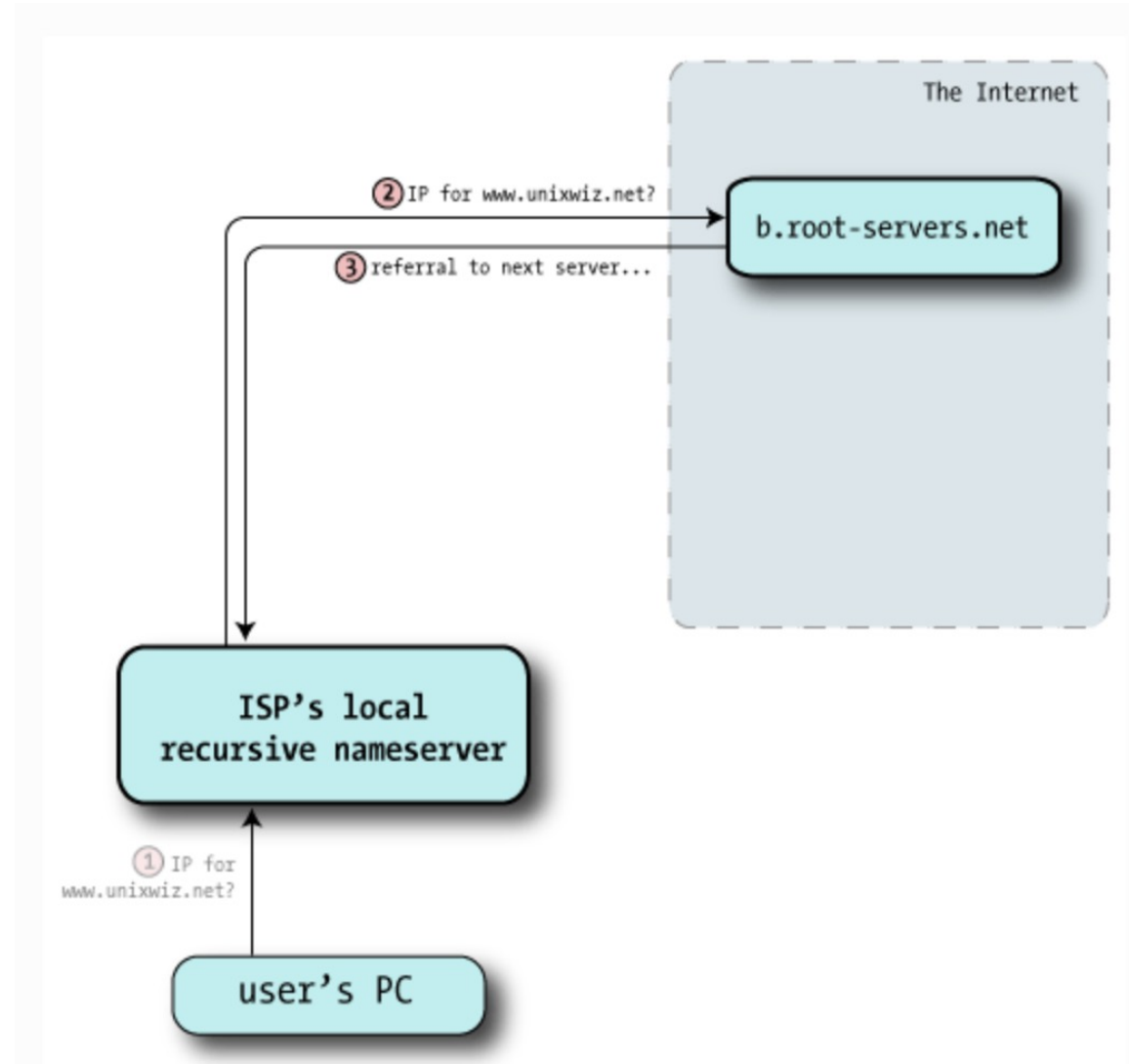
```
;; ADDITIONAL SECTION:
a.root-servers.net. 387599 IN A 198.41.0.4
a.root-servers.net. 403806 IN AAAA 2001:503:ba3e::2:30
b.root-servers.net. 401928 IN A 192.228.79.201
b.root-servers.net. 381816 IN AAAA 2001:500:84::b
c.root-servers.net. 374800 IN A 192.33.4.12
c.root-servers.net. 389084 IN AAAA 2001:500:2::c
d.root-servers.net. 403806 IN A 199.7.91.13
d.root-servers.net. 387112 IN AAAA 2001:500:2d::d
e.root-servers.net. 394211 IN A 192.203.230.10
e.root-servers.net. 390576 IN AAAA 2001:500:a8::e
f.root-servers.net. 400896 IN A 192.5.5.241
f.root-servers.net. 377160 IN AAAA 2001:500:2f::f
g.root-servers.net. 389862 IN A 192.112.36.4
h.root-servers.net. 373013 IN A 198.97.190.53
h.root-servers.net. 387632 IN AAAA 2001:500:1::53
i.root-servers.net. 119836 IN A 192.36.148.17
i.root-servers.net. 136570 IN AAAA 2001:7fe::53
j.root-servers.net. 403427 IN A 192.58.128.30
j.root-servers.net. 384850 IN AAAA 2001:503:c27::2:30
k.root-servers.net. 399148 IN A 193.0.14.129
k.root-servers.net. 394051 IN AAAA 2001:7fd::1
l.root-servers.net. 387865 IN A 199.7.83.42
l.root-servers.net. 378371 IN AAAA 2001:500:9f::42
m.root-servers.net. 381813 IN A 202.12.27.33
m.root-servers.net. 394206 IN AAAA 2001:dc3::35
```



DNS - Query flow

Passo III

Se il root server non conosce la risposta alla vostra richiesta, invia un record al vostro recursive server con un elenco dei **Global Top Level Domain (GTLD)** server responsabili di un dominio (.com, .net, .org , etc...). Questo è sotto forma di record NS.

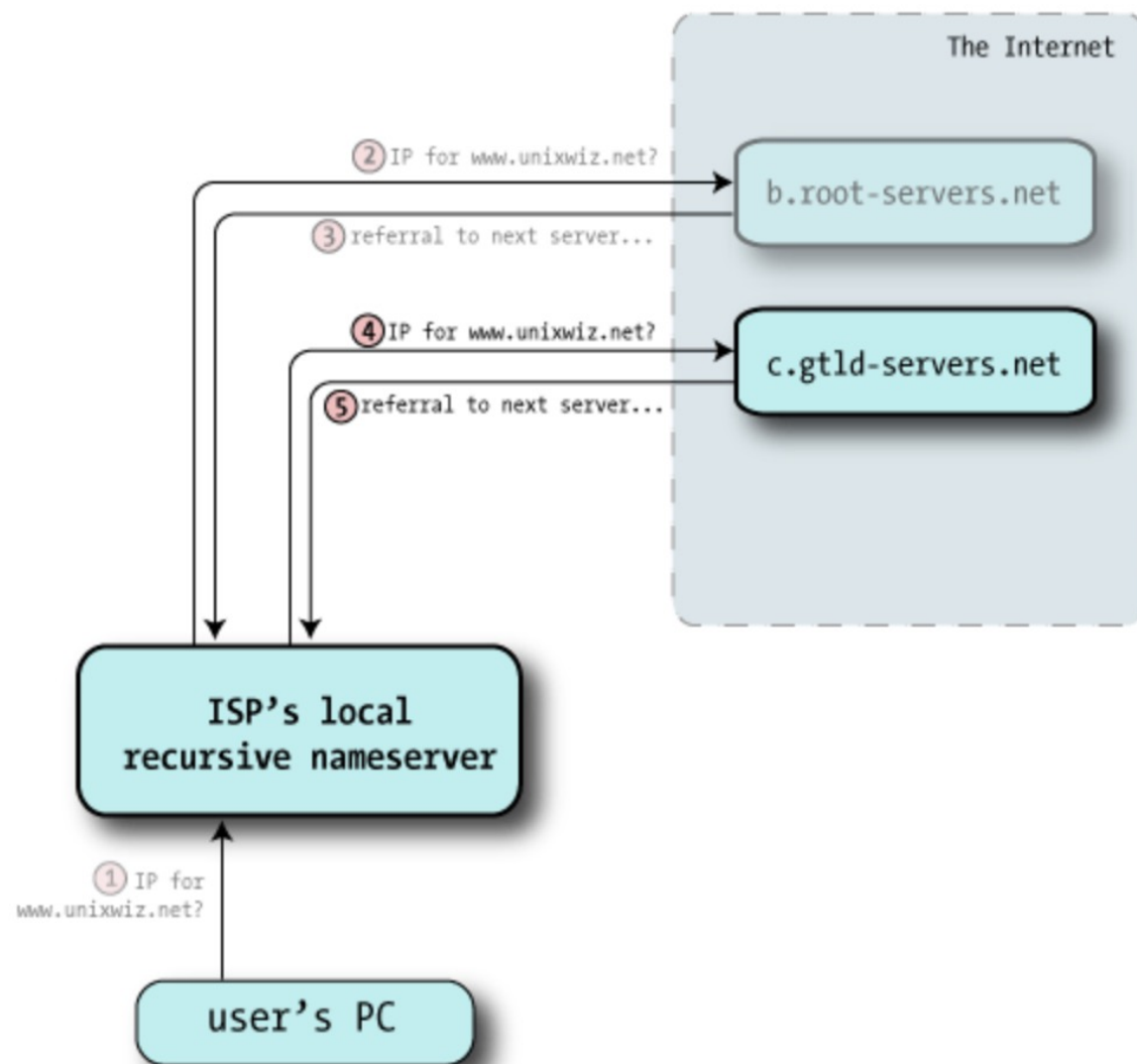


DNS - Query flow

Passo IV

Utilizzando la risposta del Root Server, il Recursive Server sceglie a caso uno dei server autoritativi (GTLD) e invia la stessa query.

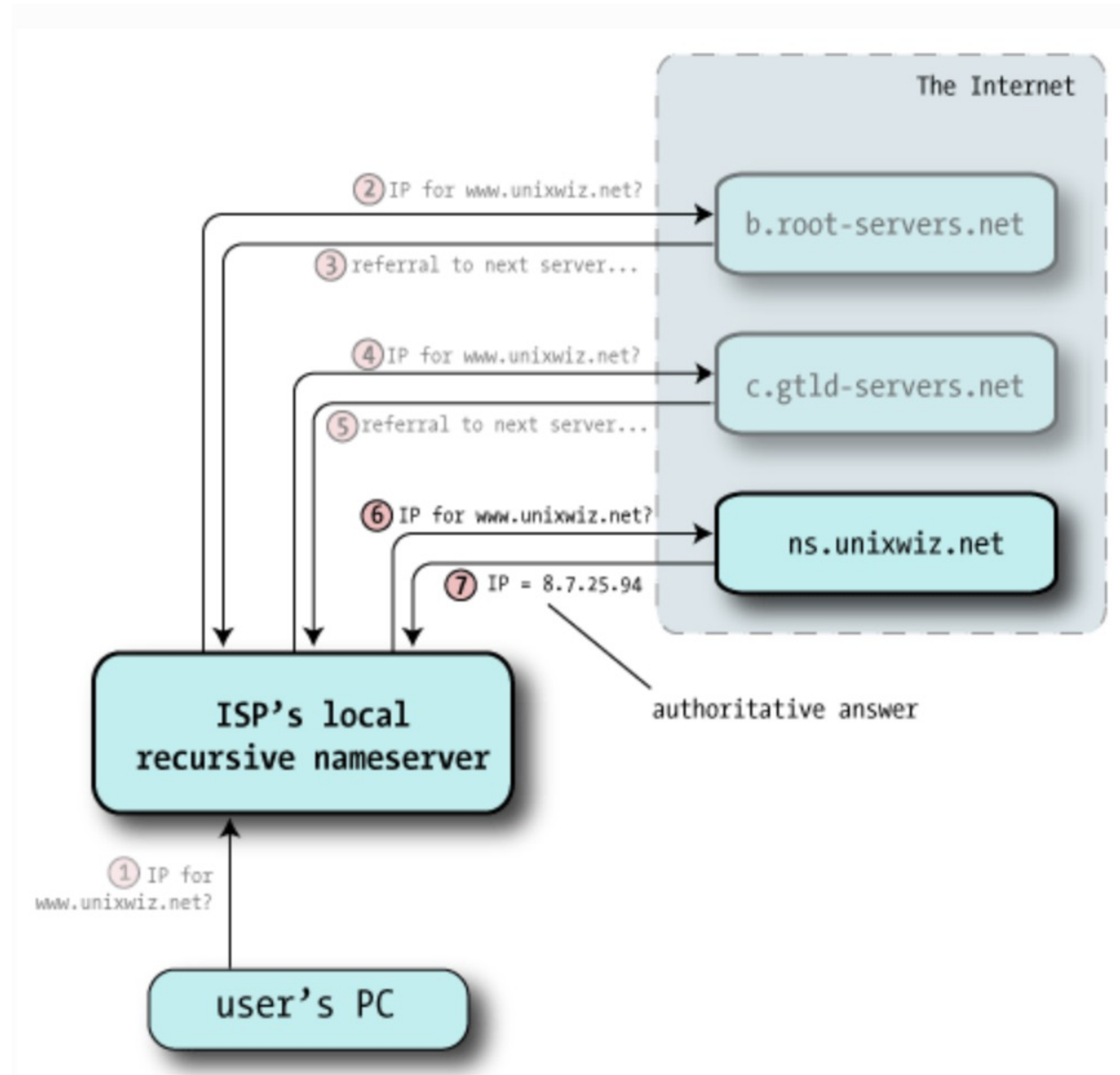
Se il server **GTLD** non conosce la risposta specifica alla nostra domanda restituisce un referral (un set di record NS) ad un server che con buona probabilità conosce la risposta.



DNS - Query flow

Passo V

Questa volta il Recursive Nameserver, seguendo una catena di risposte per conto del client, sceglie a caso uno dei nameserver e invia una terza query (la stessa delle altre due).

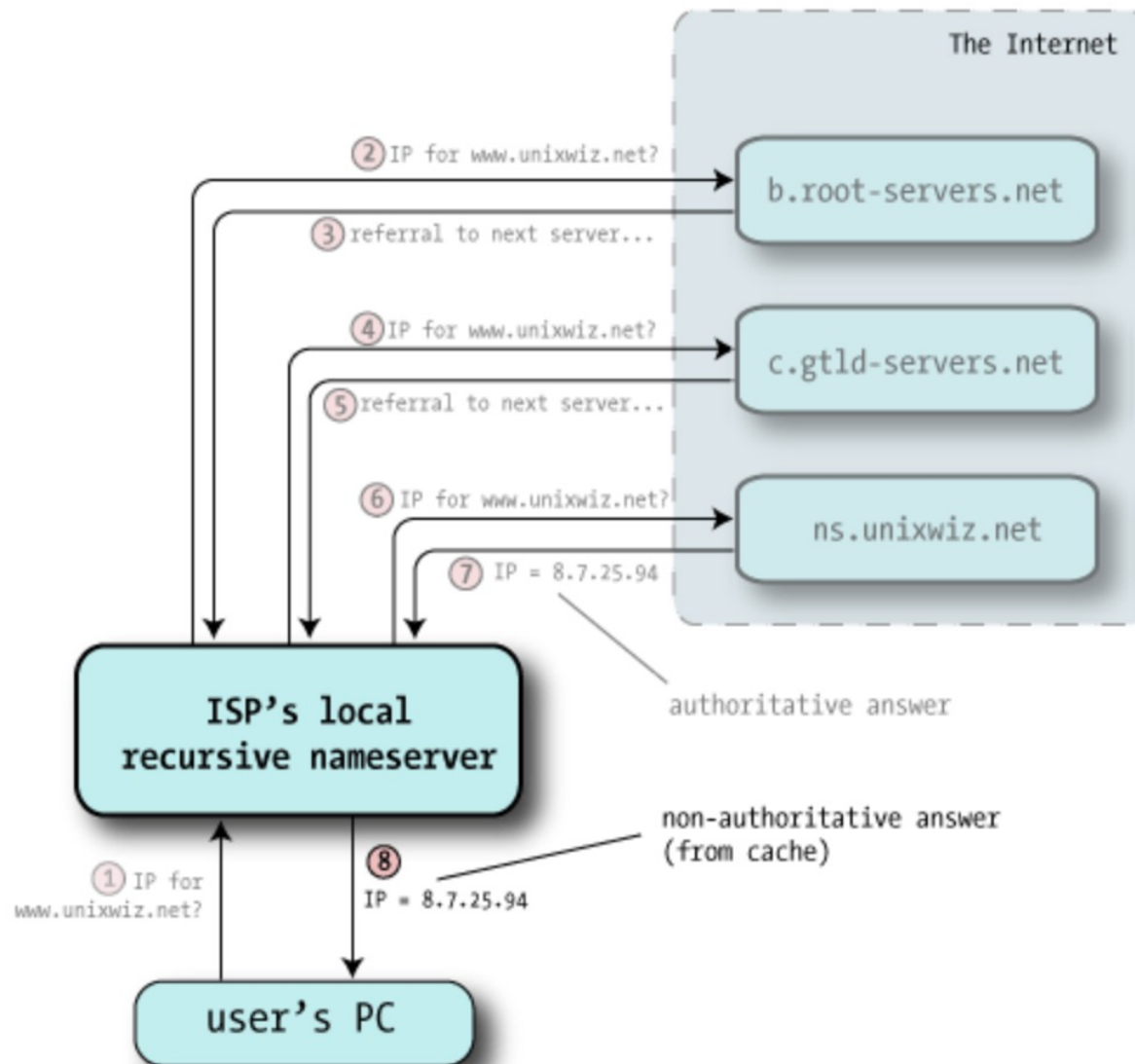


DNS - Query flow

Passo VI

Ora con la risposta in mano, il Recursive NameServer dell'ISP consegna la risposta al client e soddisfa la query.

Il Recursive NameServer archivia questa risposta nella propria cache nel caso in cui questo o qualche altro client effettui la stessa query in un secondo momento.



DNS – Installazione dnspython library

Installazione

Potete installare *dnspython* usando PIP.

```
$ pip install dnspython
```



DNS – Esempio QUERY

```
import dns.resolver as dns

name = input("inserisci il realm (esempio_: unibo.it) di cui vuoi conoscere ipaddress \n")

#Record_type = ['A', 'AAAA', 'MX', 'NS', 'TXT', 'SOA']

Record_type = ['A', 'AAAA', 'MX']
for qtype in Record_type:
    answer = dns.resolve(name,qtype, raise_on_no_answer=False)
    if answer.rrset is not None:
        print(answer.rrset)
```

```
inserisci il realm (esempio_: unibo.it) di cui vuoi conoscere ipaddress
unibo.it
unibo.it. 21512 IN A 137.204.24.147
unibo.it. 2518 IN MX 0 unibo-it.mail.protection.outlook.com.
```



DNS – QUERY DNS Inversa

Ricerca DNS inversa (record PTR)

La risoluzione DNS inversa (**rDNS**) è la determinazione di un nome di dominio associato a un indirizzo IP tramite l'interrogazione del DNS (il contrario della consueta ricerca DNS "in avanti" di un IP da un nome di dominio).

Le ricerche DNS inverse per gli indirizzi IPv4 utilizzano il dominio speciale **in-addr.arpa**. In questo dominio, un indirizzo IPv4 è rappresentato come una sequenza concatenata di quattro numeri decimali, separati da punti, a cui è aggiunto il suffisso di dominio di secondo livello **.in-addr.arpa**.

I quattro numeri decimali si ottengono suddividendo l'indirizzo IPv4 a 32 bit in quattro porzioni da 8 bit e convertendo ciascuna porzione da 8 bit in un numero decimale.

Questi numeri decimali vengono quindi concatenati nell'ordine: parte meno significativa a 8 bit per prima (più a sinistra), parte più significativa a 8 bit per ultima (più a destra)



DNS – QUERY DNS Inversa

```
from dns import reversename
from dns import resolver
domain_address = reversename.from_address('8.8.4.4')
print ("il domain address è:", (domain_address), "\n")
ip_address = reversename.to_address(domain_address)
print ("ip address è:", ip_address, "\n")
domain_name = str(resolver.resolve(domain_address, "PTR") [0])
print ("il nome di dominio è:", domain_name, "\n")
```

```
il domain address è: 4.4.8.8.in-addr.arpa.
ip address è: 8.8.4.4
il nome di dominio è: dns.google.
```



Esercizio 2 – Ritardi di Trasferimento



Esercizio 2 – Ritardi di Trasferimento

Supponiamo che un **router A** trasmetta un pacchetto verso un **router B**, che la frequenza di trasmissione del collegamento sia 1 Mbps e che la velocità di propagazione sia $2 \cdot 10^8 m/s$. Ipotizziamo in questo caso che il tempo di accodamento e il tempo di elaborazione siano trascurabili.

Quesito:

Si determini quanto deve essere al più la lunghezza D del collegamento affinché i primi 1000 bit del pacchetto arrivino a B dopo 2 ms.



Esercizio 2 – Ritardi di Trasferimento – soluzione 1/3

Sappiamo che il **tempo di Trasmissione** è dato da:

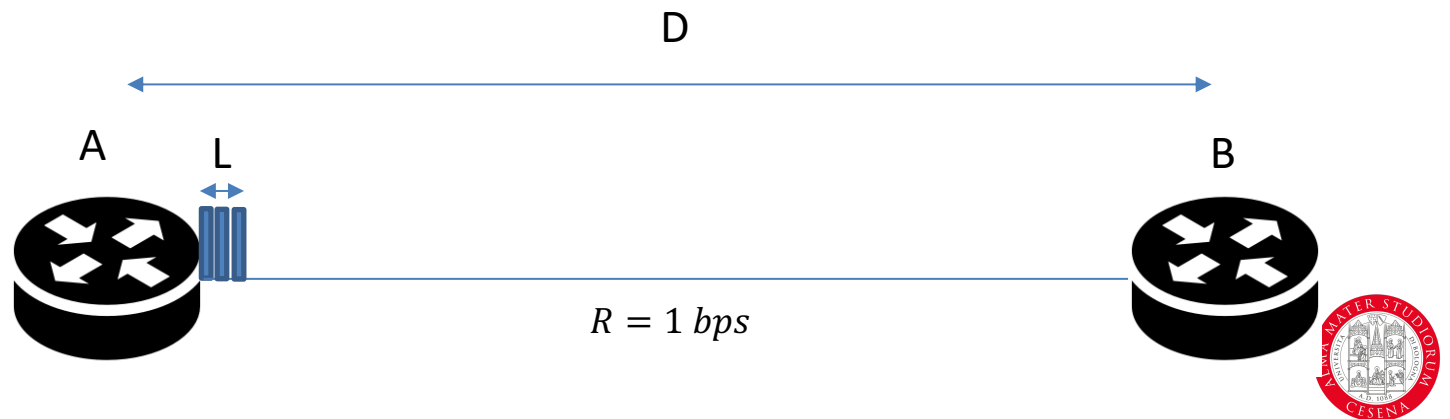
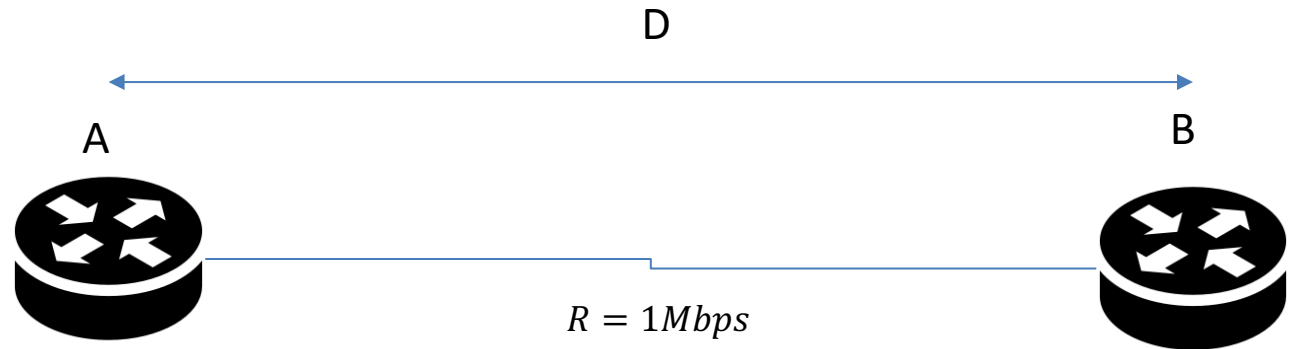
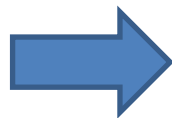
$$T = \frac{L}{R}$$

in cui L è la lunghezza del pacchetto (in bit) e R è il bit rate del Router Tx ossia il numero di bit trasmessi nell'unità di tempo.

Esempio:

Supponiamo di avere $L=3$ bit e $R=1$ bit/s

Il valore di $T=3$ s



Esercizio 2 – Ritardi di Trasferimento – soluzione 2/3

Sappiamo che il **tempo di Propagazione** è dato da:

$$\tau = \frac{D}{v}$$

in cui **D** è la lunghezza del collegamento tra A e B e **v** è la velocità di propagazione del segnale sul link fisico generalmente assunta uguale a $2 \cdot 10^8 m/s$



Esercizio 2 – Ritardi di Trasferimento – soluzione 3/3

Il tempo complessivo in questo caso è dato dalla somma del tempo di trasmissione e del tempo di propagazione:

$$T_{tot} = T_{trasm} + T_{prop}$$

Poniamo quindi la condizione che il tempo totale sia pari a 2 ms

$$2 \cdot 10^{-3} s = \frac{L}{R} + \frac{D}{v} = \frac{1000 \text{ bit}}{1 \cdot \frac{10^6 \text{ bit}}{s}} + \frac{D}{2 \cdot \frac{10^8 m}{s}} = 10^{-3} s + \frac{D}{2 \cdot 10^8} s$$

$\Rightarrow 10^{-3} = \frac{D}{2 \cdot 10^8}$

\Downarrow

$2 \cdot 10^5 m = D$

\Leftarrow

D = 200 Km



Esercizio 3 – Ritardi di Trasferimento



Esercizio 3 – Ritardi di Trasferimento

Un router A trasmette in modo continuato dati su un collegamento lungo **10 km** verso un router B, la cui velocità di propagazione è **$2 \cdot 10^8 \text{m/s}$** .

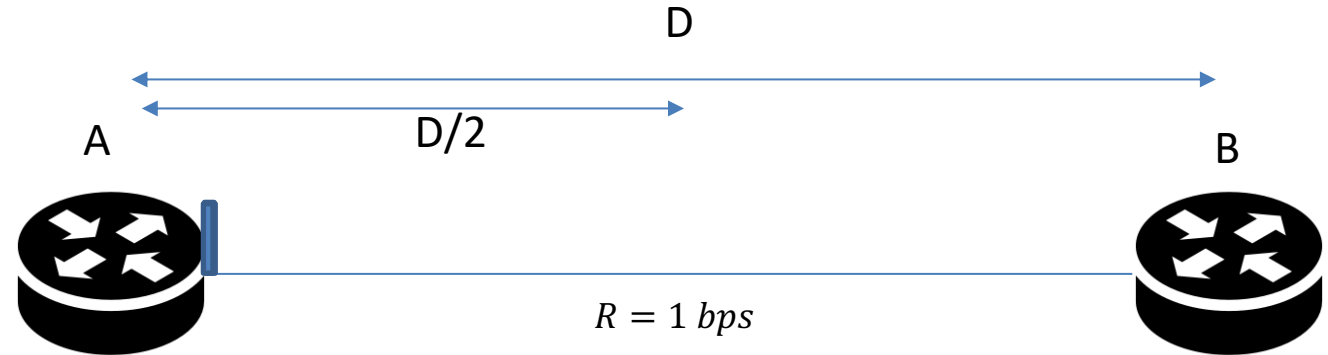
Quesito:

Si determini quale è la frequenza di trasmissione se al più 500 bit possono essere simultaneamente presenti nel collegamento.



Esercizio 3 – Ritardi di Trasferimento – soluzione 1/2


I bit che possono essere simultaneamente presenti nel collegamento sono quelli che il router riesce a trasmettere mentre il segnale si propaga nel canale.



Esempio

Se il router A trasmette ad una frequenza di **1 bit/s**= R e il bit ci impiega **2 s** = d_{prop} ad arrivare presso B abbiamo che al massimo sul collegamento possono essere presenti simultaneamente 2 bit.

Infatti quando il primo bit arriva a metà del percorso, il secondo bit parte dal router A.

Ossia $1 \text{ bit/s} \cdot 2 \text{ s} = 2 \text{ bit}$  $R \cdot d_{prop} = \text{num. di bit presenti simultaneamente nel canale trasmissivo}$



Esercizio 3 – Ritardi di Trasferimento – soluzione 2/2

Imponiamo quindi la condizione che al più sul collegamento possono essere presenti 500 bit simultaneamente

$$R \cdot d_{prop} = 500 \text{ bit}$$

$$R = \frac{500 \text{ bit}}{d_{prop}} = \frac{500 \text{ bit}}{10 \cdot 10^3 \text{ m} / 2 \cdot 10^8 \text{ m/s}} = \frac{1000 \cdot 10^5 \text{ bit}}{10 \text{ s}} = \frac{10^7 \text{ bit}}{\text{s}} = 10 \text{ Mbps}$$



Esercizio 4 – Ritardi di Trasferimento



Esercizio 4 – Ritardi di Trasferimento

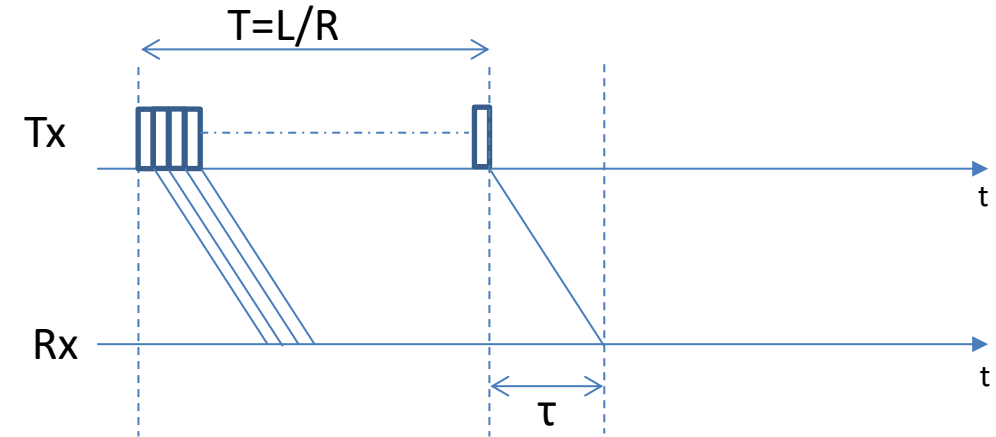
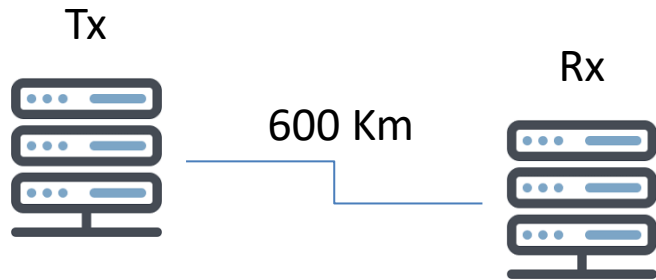
Un sistema trasmissivo avente velocità di 100 kb/s presenta una lunghezza complessiva tra i due terminali di 600 km. Si considerino trascurabili i tempi di elaborazione e di accodamento.

Quesito:

Si calcoli il tempo che intercorre fra la trasmissione del primo bit e la ricezione dell'ultimo bit di un pacchetto lungo 3000 bit, assumendo che il ritardo di propagazione sia di 5 $\mu\text{s}/\text{km}$.



Esercizio 4 – Ritardi di Trasferimento – Soluzione 1/2



Esempio

Se il router trasmette ad un rate di n bit/s e la distanza tra i due router è di 1 Km, avremo che l'ultimo bit del pacchetto di n bit arriva a destinazione dopo un tempo pari al tempo necessario a trasmettere gli n bit del pacchetto, ossia 1 secondo, a cui dobbiamo sommare il tempo impiegato dall'ultimo bit a percorrere il tragitto da A a B, pari al tempo di propagazione.

➡ $\frac{n \text{ bit}}{n \text{ bit/s}} + \text{tempo di propagazione}$



Esercizio 4 – Ritardi di Trasferimento – Soluzione 2/2

Il tempo di trasmissione è $T = \frac{L}{R} = \frac{3000 \text{ bit}}{100000 \text{ bit/sec}} = 3 \cdot 10^{-2} \text{ sec} = 30 \text{ ms}$

Il tempo di propagazione è $\tau = 5 \mu\text{s} / \text{Km} \cdot 600 \text{ Km} = 3000 \mu\text{s} = 3.0 \text{ ms}$

Il tempo che intercorre fra la trasmissione del primo bit e quello della ricezione dell'ultimo bit è:

$$T_{tot} = T_{trasm} + \tau_{prop} = 30 \text{ ms} + 3.0 \text{ ms} = 33 \text{ ms}$$



Esercizio 5 – Ritardi di Trasferimento



Esercizio 5 – Ritardi di Trasferimento

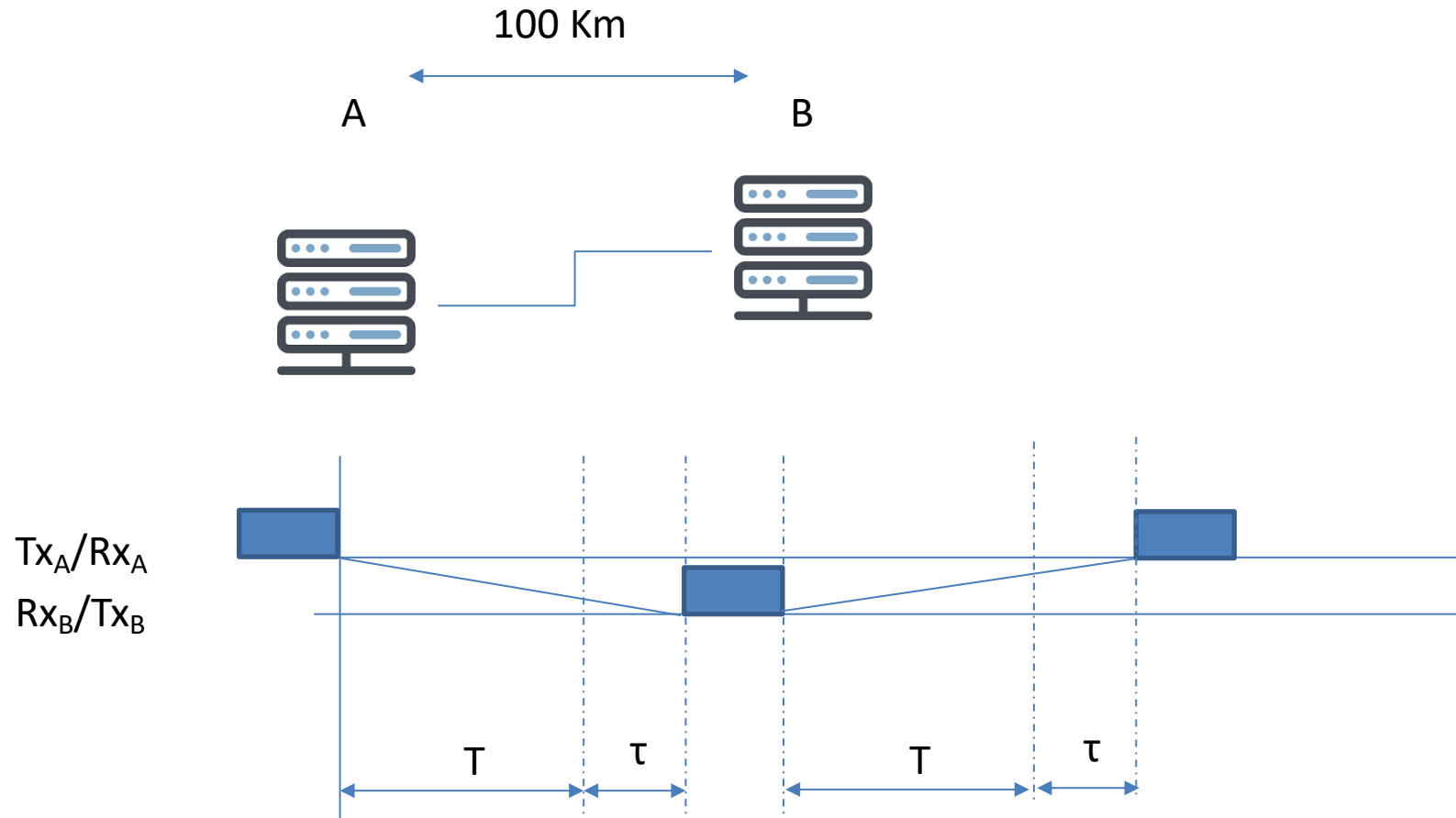
Un pacchetto di 10000 bit viene inviato dal server A alla velocità di 100 kb/s su un collegamento di 100 km. Il pacchetto viene ricevuto tutto in B e poi viene rimandato al mittente A alla stessa velocità di trasmissione.

Quesito:

Si calcoli l'intervallo di tempo che intercorre fra la **trasmissione** del primo bit in A e la **ricezione** dell'ultimo bit, sempre in A, assumendo che la velocità del segnale sia di 200000 km/s.



Esercizio 5 – Ritardi di Trasferimento – Soluzione 1/2



Esercizio 5 – Ritardi di Trasferimento – Soluzione 2/2

$$T_{tot} = T + \tau + T + \tau = 2 \cdot (T + \tau)$$

$$2 \cdot \left(\frac{L}{R} + \frac{\text{distanza}}{\text{velocità di propagazione}} \right) = 2 \cdot \left(\frac{10000 \text{ bit}}{100 \cdot \frac{10^3 \text{ bit}}{s}} + \frac{100 \text{ Km}}{200 \cdot \frac{10^3 \text{ Km}}{s}} \right)$$

$$2 \cdot (0.1 + 0.5 \cdot 10^{-3}) = 0.2 + 0.001 = 0.201 \text{ s} = 201 \text{ ms}$$



Esercizio 6 – Ritardi di Trasferimento



Esercizio 6 – Ritardi di Trasferimento

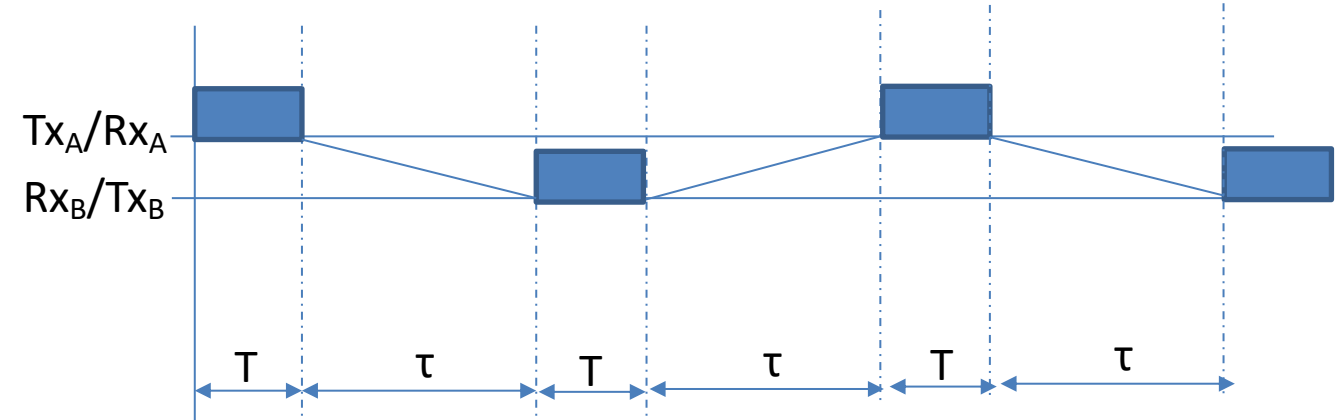
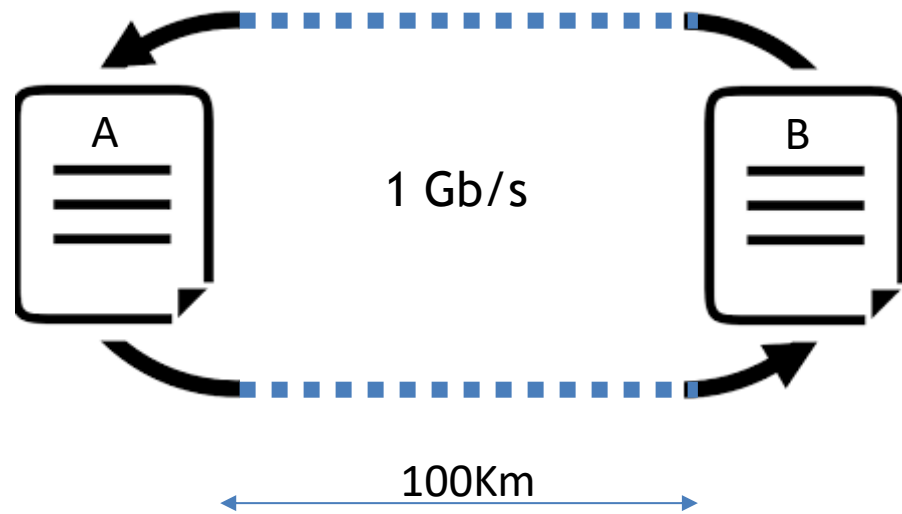
Due clienti di un internet service provider stanno utilizzando una connessione su un canale half-duplex lungo 100 km alla velocità di trasmissione di 1 Gb/s. I due utenti debbono trasferirsi l'un l'altro dei files, ed eseguono la trasmissione in *half-duplex* commutando il canale ogni 10000 bit ricevuti.

Quesiti:

- Assumendo che la commutazione del canale avvenga in un tempo nullo e che la velocità del segnale sia di 200000 km/s, a che velocità effettiva trasmettono i files?
- Si calcoli il totale tempo di trasmissione di un file di 1Gbyte.



Esercizio 6 – Ritardi di Trasferimento – soluzione 1/2



Tempo di Trasmissione $T = \frac{L}{R} = \frac{10000 \text{ bit}}{10^9 \text{ bit/s}} = 10^{-6} \text{ s} = 10 \mu\text{s}$

Tempo di Propagazione $\tau = \frac{100 \text{ km}}{200 \cdot 10^3 \text{ km/s}} = 0.5 \cdot 10^{-3} = 500 \mu\text{s}$

«A» trasmette il suo blocco di dati, aspetta che sia arrivato, commuta il canale ed aspetta il blocco trasmesso da «B»

Ogni utente trasmette 10000 bit in un periodo di $2 \times 510 \mu\text{s} = 1,02 \text{ ms}$

Il rate di trasferimento effettivo è: $R = 10 \text{ kbit}/1,02 \text{ ms} = 9,8 \text{ Mbit/s}$



Esercizio 6 – Ritardi di Trasferimento – soluzione 2/2

File da trasferire 1Gbyte = 8Gbit

Tempo complessivo necessario a trasferire il file da 1 Gbyte

$$\Rightarrow T = 8 \text{ Gbit} / 9.8 \cdot 10^{-3} \text{ Gbit/s} = 0.816 \cdot 10^3 \text{ s} = 816 \text{ s}$$

Oppure, si considerino il numero complessivo di trasmissioni necessarie per inviare un file di 8Gbit

$$N = \frac{8 \cdot 10^9 \text{ bit}}{10 \cdot 10^3 \text{ bit}} = 0.8 \cdot 10^6 = 800000$$

Ogni trasmissione dura 1,02ms

$$\Rightarrow T = 800000 \cdot 1.02 \text{ ms} = 816000 \text{ ms} = 816 \text{ s}$$



Esercizio 7 – Ritardi di Trasferimento



Esercizio 7 – Ritardi di Trasferimento

Un sistema trasmissivo della velocità di 100 kb/s presenta una lunghezza di 600 km tra Tx ed Rx. Fra i due elementi di testa sono presenti due router, ciascuno dei quali presenta una latenza (latency), ossia un tempo di accodamento in uscita, pari al tempo di trasmissione, ed una capacità di 100kb/s. Si consideri trascurabile il tempo di elaborazione.

Quesiti:

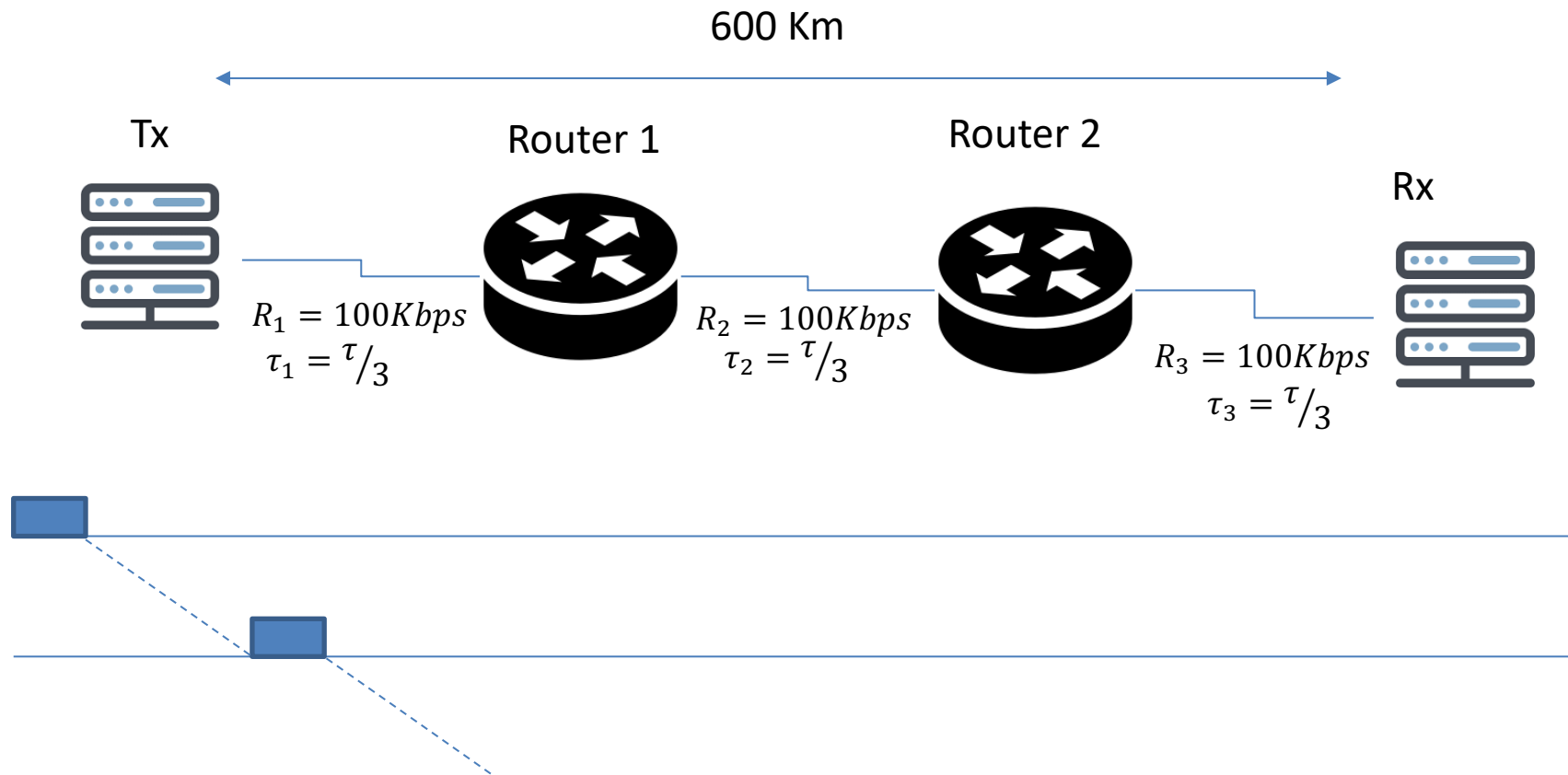
Si chiede di calcolare il ritardo totale nella trasmissione di un pacchetto di 3000 bit, assumendo un ritardo di propagazione di 5 μ s: nei due casi in cui nei router si applichi:

- la modalità *store and forward*
- la modalità *cut-through*, considerando che la lunghezza dell'*header* sia di 200 bit (a parità di dimensione totale del pacchetto).



Esercizio 7 – Ritardi di Trasferimento – Soluzione 1/5

Scenario: STORE & FORWARD → il pacchetto deve essere completamente ricevuto prima di essere ritrasmesso

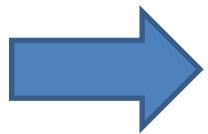


Esercizio 7 – Ritardi di Trasferimento – Soluzione 2/5

Scenario: STORE & FORWARD → il pacchetto deve essere completamente ricevuto prima di essere ritrasmesso

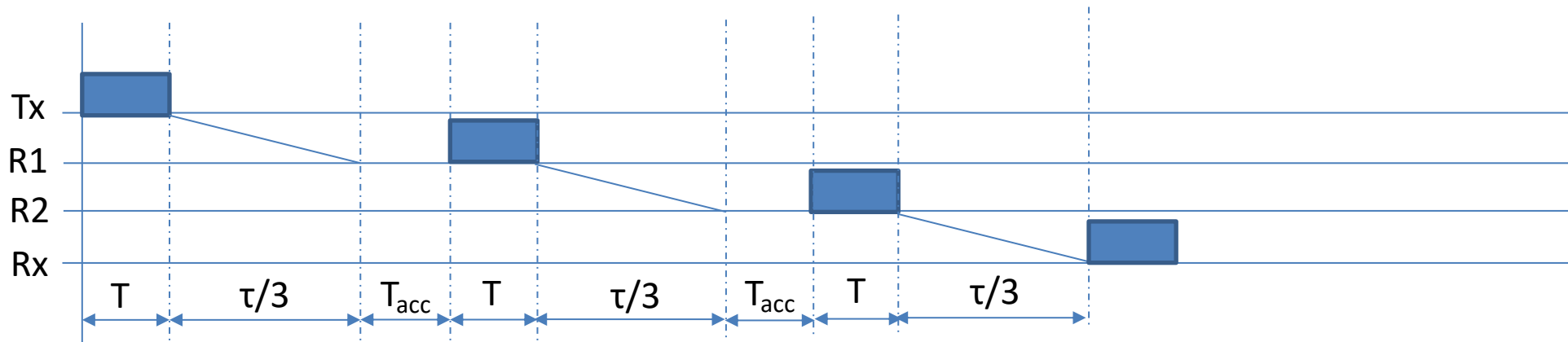
Osservazioni preliminari:

- Abbiamo 3 elementi trasmissivi quindi 3 tempi di trasmissione → $3 \cdot T = 3 \cdot \frac{L}{R} = 3 \cdot \frac{3000 \text{ bit}}{100 \cdot 10^3 \text{ bps}} = 3 \cdot 30 \cdot 10^{-3} \text{ s} = 90 \text{ ms}$
- Abbiamo 2 elementi che introducono latenza per accodamento e ritrasmissione → $2 \cdot T_{acc} = 2 \cdot T = 60 \text{ ms}$
- Abbiamo il tempo di propagazione fisico → $\tau = 5 \mu\text{s} / \text{Km} \cdot 600 \text{ Km} = 3000 \mu\text{s} = 3.0 \text{ ms}$



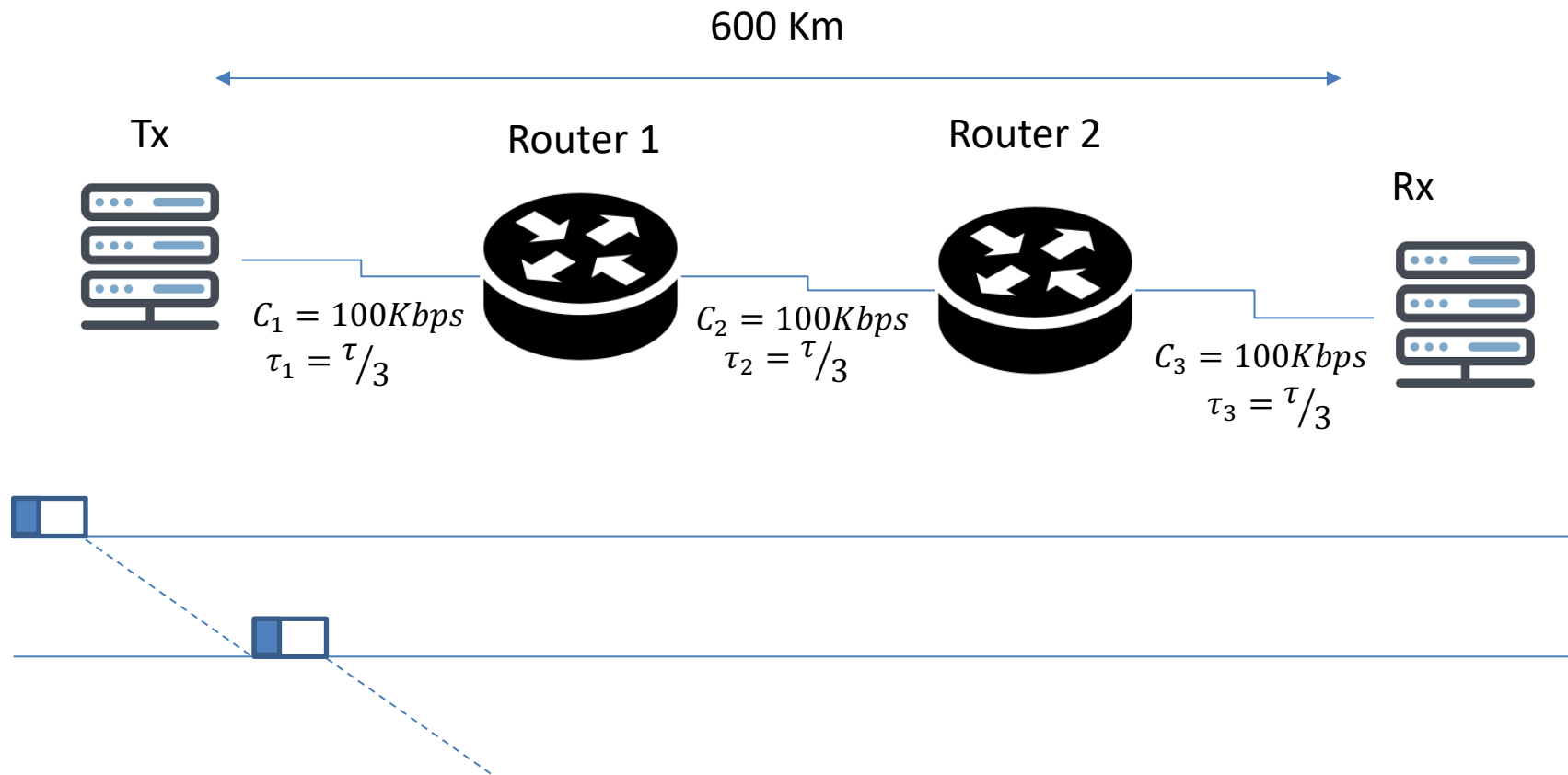
Sappiamo che il tempo di accodamento $L = T$, quindi facendo la somma otteniamo che:

$$T_{tot} = 3 \cdot T + 2 \cdot T + \tau = 5 \cdot T + \tau = 5 \cdot 30 \text{ ms} + 3 \text{ ms} = 153 \text{ ms}$$



Esercizio 7 – Ritardi di Trasferimento – Soluzione 3/5

Scenario: Cut & Through → il pacchetto viene ritrasmesso alla completa ricezione dell'header

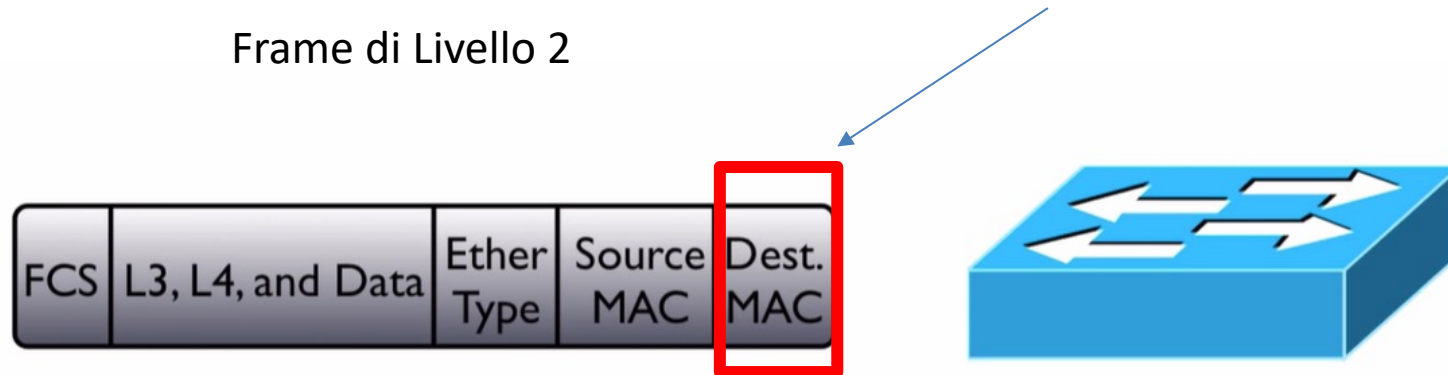


Esercizio 7 – Ritardi di Trasferimento – Soluzione 4/5

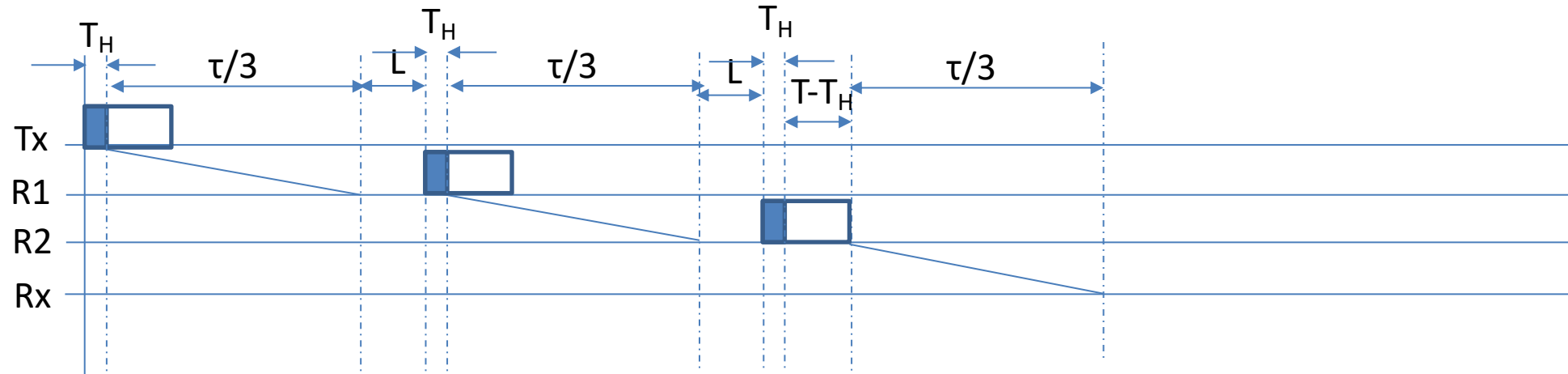
Scenario: Cut & Through → il pacchetto viene ritrasmesso alla completa ricezione dell'header

Osservazioni preliminari:

Nel caso *cut-through* il tempo di trasmissione dell'*header* si paga tre volte (l'*header* è sempre trasmesso con modalità *store and forward*) mentre il tempo di trasmissione del resto si paga una sola volta. Il tempo totale è dunque



Esercizio 7 – Ritardi di Trasferimento – Soluzione 5/5



$$T_{tot} = \underbrace{T_H + (T - T_H)}_{\text{Server 1}} + \underbrace{T_H + (T - T_H)}_{\text{router 1}} + \underbrace{T_H + (T - T_H)}_{\text{router 2}} + \tau$$

Labels for the equation components:

- Tempo di trasmissione dell'Header (points to T_H)
- Tempo di trasmissione del resto del pacchetto (points to $T - T_H$)
- Tempo di accodamento dell'Header (points to T_H)
- Tempo di ritrasmissione del resto del pacchetto (points to $T - T_H$)
- Tempo di ritrasmissione dell'Header (points to T_H)
- Tempo di accodamento dell'Header (points to T_H)
- Tempo di ritrasmissione del resto del pacchetto (points to $T - T_H$)
- Tempo di ritrasmissione dell'Header (points to T_H)



Esercizio 7 – Ritardi di Trasferimento – Soluzione 5/5

$$\cancel{T_H} + \boxed{T} - \cancel{T_H} + \cancel{T_H} + \boxed{T} - \cancel{T_H} + \boxed{T_H} + \cancel{T_H} + \boxed{T} - \cancel{T_H} + \boxed{T_H} + \tau$$

$$T = \frac{L}{R} = \frac{3000 \text{ bit}}{100 \cdot 10^3 \text{ bps}} = 30 \cdot 10^{-3} \text{ s} = 30 \text{ ms}$$

$$\tau = 5 \mu\text{s/Km} \cdot 600 \text{ Km} = 3000 \mu\text{s} = 3.0 \text{ ms}$$

$$T_{tot} = \boxed{3 \cdot T} + 2 \cdot T_H + \tau = 90 \text{ ms} + 4 \text{ ms} + 3 \text{ ms} = 97 \text{ ms}$$

$$T_H = \frac{200 \text{ bit}}{100 \cdot 10^3 \text{ bit/sec}} = 2 \cdot 10^{-3} \text{ sec} = 2 \text{ ms}$$

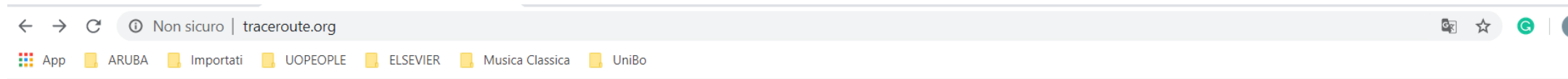


Esercizio 8 – UTILIZZO DI TRACEROUTE



Esercizio 8 – Utilizzo di Traceroute

TRACEROUTE è un Programma diagnostico: <http://www.traceroute.org>



...traceroute.org...

Maintained by [Thomas Kernen](#)

Please feel free to send me updates, links, corrections, extra info

Note that I'm unable to provide support for the linked web pages

sponsored by



Selezionate il paese di appartenenza del server originante

By country:

[Argentina](#)
[Brasil](#)
[Cyprus](#)
[Germany](#)
[Iran](#)
[Latvia](#)
[New Zealand](#)
[Saint Kitts](#)
[Taiwan](#)
[Uzbekistan](#)

[Armenia](#)
[Bulgaria](#)
[Czech Republic](#)
[Greece](#)
[Ireland](#)
[Lithuania](#)
[Pakistan](#)
[Saint Vincent](#)
[Thailand](#)

[Australia](#)
[Canada](#)
[Denmark](#)
[Grenada](#)
[Israel](#)
[Luxembourg](#)
[Paraguay](#)
[Singapore](#)
[Togo](#)

[Austria](#)
[Chile](#)
[Ecuador](#)
[Hong Kong](#)
[Italy](#)
[Macau](#)
[Philippines](#)
[Slovakia](#)
[Turkey](#)

[Bangladesh](#)
[China](#)
[Estonia](#)
[Hungary](#)
[Japan](#)
[Malaysia](#)
[Poland](#)
[South Africa](#)
[Turkmenistan](#)

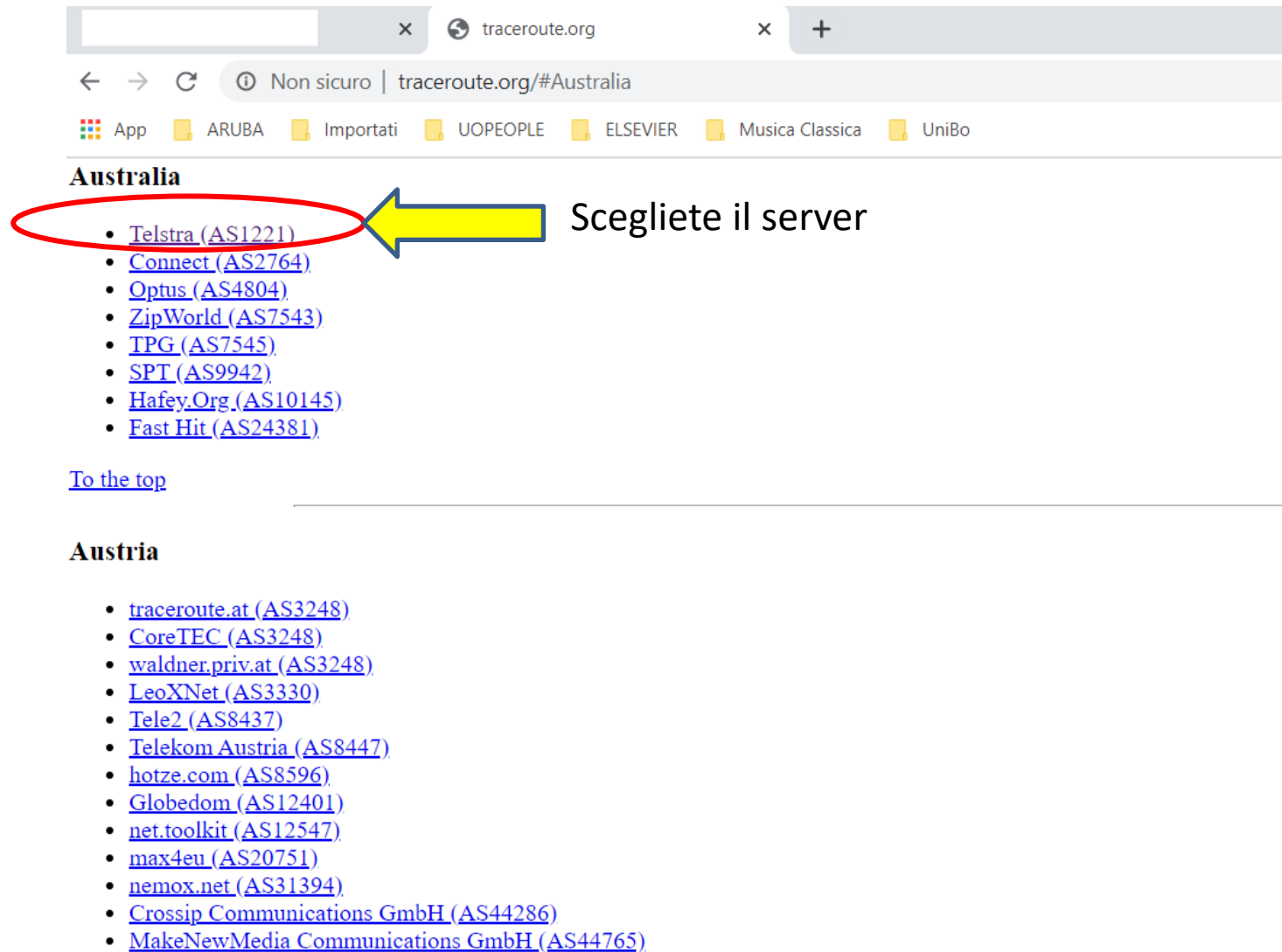
[Belarus](#)
[Colombia](#)
[Faroe Islands](#)
[Iceland](#)
[Kenya](#)
[Malta](#)
[Portugal](#)
[Spain](#)
[Ukraine](#)

[Belgium](#)
[Costa Rica](#)
[Finland](#)
[India](#)
[Korea](#)
[Mexico](#)
[Romania](#)
[Sweden](#)
[United Kingdom](#)

[Bolivia](#)
[Croatia](#)
[France](#)
[Indonesia](#)
[Kyrgyzstan](#)
[The Netherlands](#)
[Russia](#)
[Switzerland](#)
[USA](#)



Esercizio 8 – Utilizzo di Traceroute



The screenshot shows the traceroute.org website with the URL traceroute.org/#Australia in the address bar. The page displays a list of servers for Australia, with the first server, [Telstra \(AS1221\)](#), circled in red. A yellow arrow points to this server with the text "Scegliete il server". Below the list of servers, there is a link "To the top".

Australia

- [Telstra \(AS1221\)](#)
- [Connect \(AS2764\)](#)
- [Optus \(AS4804\)](#)
- [ZipWorld \(AS7543\)](#)
- [TPG \(AS7545\)](#)
- [SPT \(AS9942\)](#)
- [Hafey.Org \(AS10145\)](#)
- [Fast Hit \(AS24381\)](#)

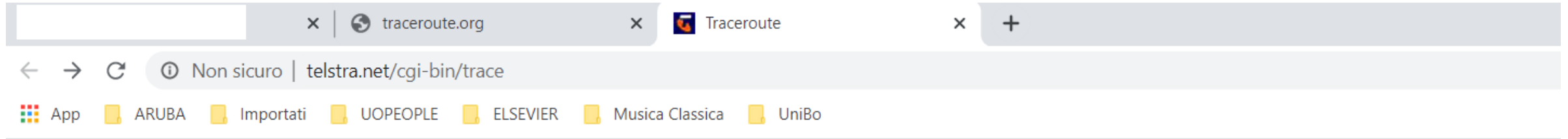
[To the top](#)

Austria

- [traceroute.at \(AS3248\)](#)
- [CoreTEC \(AS3248\)](#)
- [waldner.priv.at \(AS3248\)](#)
- [LeoXNet \(AS3330\)](#)
- [Tele2 \(AS8437\)](#)
- [Telekom Austria \(AS8447\)](#)
- [hotze.com \(AS8596\)](#)
- [Globedom \(AS12401\)](#)
- [net.toolkit \(AS12547\)](#)
- [max4eu \(AS20751\)](#)
- [nemox.net \(AS31394\)](#)
- [Crossip Communications GmbH \(AS44286\)](#)
- [MakeNewMedia Communications GmbH \(AS44765\)](#)



Esercizio 8 – Utilizzo di Traceroute



Traceroute

This traceroute commences from www.telstra.net, within AS 1221.

Enter the desired destination host.domain or IPv4 or IPv6 address:

8.8.8.8

Inserite IP address che volete testare

There are other traceroute sites listed [here](#).

The traceroute CGI source can be found via:



Una volta inserito l'ip address dell'host di destinazione e dato invio, il server **AS1221** invia un certo numero di pacchetti **speciali** verso tale destinazione; questi pacchetti passano attraverso vari router. Quando un router ne riceve uno invia un breve messaggio che torna all'origine. **Questo messaggio contiene il nome e ip address del router che l'ha inviato.**



Esercizio 8 – Utilizzo di Traceroute

Prima di eseguire un comando traceroute, cerchiamo di comprendere il meccanismo di rete chiamato **"time to live" (TTL)**.

TTL limita la durata della "vita" dei dati in una rete IP. Ad ogni pacchetto di dati viene assegnato un valore TTL. Ogni volta che un pacchetto di dati raggiunge un salto, il valore TTL viene diminuito di uno.

Un altro elemento chiave da comprendere è il "tempo di andata e ritorno" (**RTT – Round Trip Time**).

Traceroute garantisce che ogni salto nel percorso verso un dispositivo di destinazione rilasci un pacchetto e restituisca un messaggio di errore ICMP.

Ciò significa che traceroute può misurare il tempo che intercorre tra il momento in cui i dati vengono inviati e il momento in cui il messaggio ICMP viene ricevuto di nuovo per ogni hop, fornendo il valore RTT per ogni hop.



Esercizio 8 – Utilizzo di Traceroute

Per illustrare meglio questo aspetto, supponiamo di eseguire un traceroute e di specificare un massimo di 30 hop. Traceroute invierà pacchetti con un TTL pari a uno al server di destinazione. Il primo dispositivo di rete attraverso cui passano i dati ridurrà il TTL al valore zero e verrà inviato un messaggio che informa che i pacchetti sono stati ignorati. Questo ci dà l'RTT per il salto numero uno.

Da lì, i pacchetti di dati vengono inviati al server di destinazione con un TTL di due. Quando i pacchetti passano attraverso il primo hop, il TTL diminuisce a uno. Quando passano attraverso il secondo salto, diminuisce a zero. Il messaggio viene inviato di nuovo. Questo ci dà l'RTT per il salto numero due.

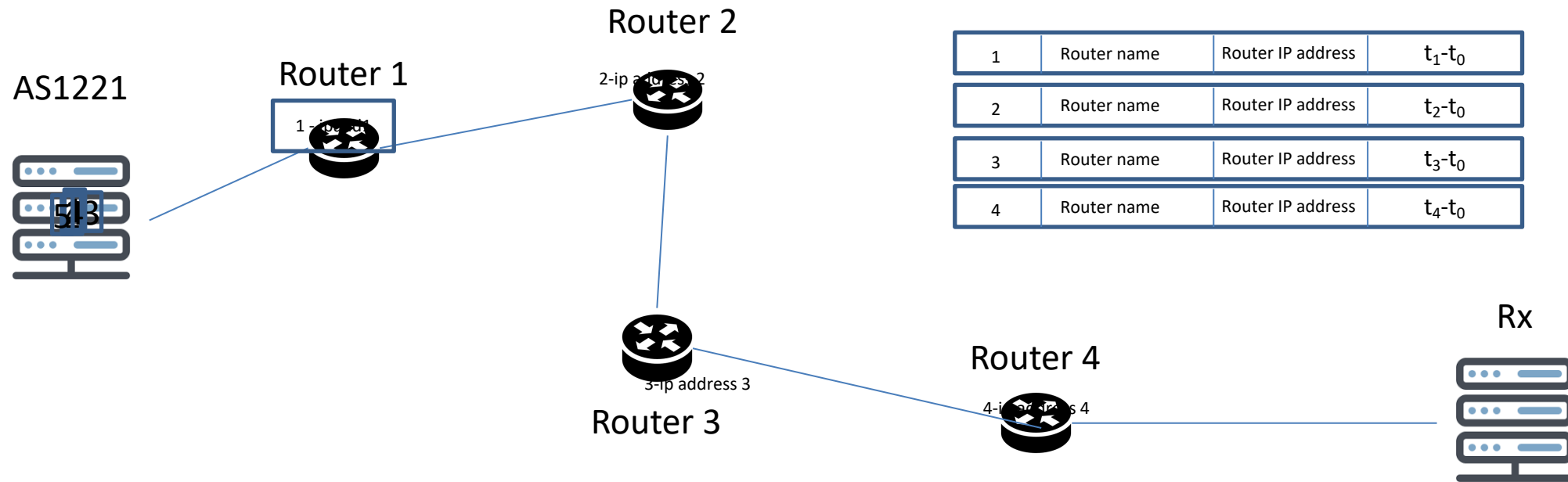
Questo processo si ripeterà fino a quando i pacchetti di dati **raggiungeranno il dispositivo di destinazione o raggiungeranno il numero massimo di hop.**

Alla fine di questo test, avremo il numero di hop al dispositivo di destinazione, la lunghezza RTT per ogni hop e il nome del dispositivo e l'indirizzo IP per ogni hop.



Esercizio 8 – Utilizzo di Traceroute

Supponiamo che tra il server Tx e la destinazione vi siano N-1 router (nel nostro esempio sono 4), l'AS1221 invia N pacchetti speciali nella rete (ossia 5). I pacchetti sono etichettati da 1 a N in sequenza. Quando l'i-esimo router riceve il pacchetto marcato con i , invece di mandarlo avanti, invia un messaggio verso l'origine. Quando l'host di destinazione riceve il pacchetto speciale N-esimo, anche esso restituisce un messaggio all'origine.



L'Host di origine quindi registra il tempo intercorso tra l'invio di un pacchetto e la ricezione del corrispondente messaggio di ritorno e memorizza anche il nome e l'indirizzo del router (o del destinatario) che restituisce il messaggio. In questo modo, l'A1221 può ricostruire il percorso intrapreso dai pacchetti ed è inoltre in grado di determinare i ritardi di andata e ritorno di tutte le tratte.



Esercizio 8 – Utilizzo di Traceroute

Traceroute

Traceroute ripete l'operazione per tre volte consecutive, motivo per cui il risultato prevede 6 colonne e non 4.

This traceroute commences from www.telstra.net, within AS 1221.

Enter the desired destination host.domain or IPv4 or IPv6 address:

Valore dell'etichetta
assegnata al pacchetto
ossia il numero del
router lungo il
percorso

Nome del router

Indirizzo IP
del router

Prima misura del
ritardo di andata e
ritorno

1	gigabitethernet3-3.exi2.melbourne.telstra.net	(203.50.77.53)	0.368 ms	0.201 ms	0.239 ms
2	bundle-ether3-100.win-core10.melbourne.telstra.net	(203.50.80.129)	2.609 ms	1.727 ms	2.240 ms
3	bundle-ether12.ken-core10.sydney.telstra.net	(203.50.11.122)	12.856 ms	11.848 ms	12.610 ms
4	bundle-ether1.ken-edge903.sydney.telstra.net	(203.50.11.173)	11.732 ms	11.723 ms	11.857 ms
5	72.14.212.22	(72.14.212.22)	12.479 ms	12.474 ms	12.482 ms

There are other traceroute sites listed [here](#).

The traceroute CGI source can be found via:

 **carpeNet**

Se l'origine riceve meno di tre messaggi da ogni router
Traceroute pone un asterisco subito dopo il numero del router

Esercizio 8 – Utilizzo di Traceroute

Traceroute è presente come comando sia su Windows che su Linux e Mac

Su **Windows** è fruibile aprendo una shell di DOS,
E lanciando il comando «tracert xxx.xxx.xxx.xxx»



```
Prompt dei comandi
Microsoft Windows [Versione 10.0.18363.592]
(c) 2019 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\apirodd>tracert 8.8.8.8

Traccia instradamento verso dns.google [8.8.8.8]
su un massimo di 30 punti di passaggio:

 1  4 ms   5 ms   5 ms  DESKTOP-9SK0BHP [10.0.0.2]
 2  6 ms   6 ms   6 ms  10.0.0.1
 3  *      *      *      Richiesta scaduta.
 4 12 ms  12 ms  11 ms  172.17.54.60
 5 12 ms  12 ms  12 ms  172.17.54.144
 6 12 ms  12 ms  12 ms  172.19.177.42
 7 29 ms  25 ms  26 ms  172.19.177.4
 8 26 ms  24 ms  24 ms  etrunk49.milano50.mil.seabone.net [195.22.205.116]
 9 21 ms  21 ms  22 ms  72.14.221.64
10 25 ms  24 ms  24 ms  74.125.245.241
11 24 ms  23 ms  23 ms  172.253.79.35
12 24 ms  24 ms  24 ms  dns.google [8.8.8.8]

Traccia completata.

C:\Users\apirodd>
```

Su **Linux e Mac** è fruibile aprendo una Command Line Interface,
E lanciando il comando «traceroute xxx.xxx.xxx.xxx»



```
apirodd@ubuntunet2008:~$ traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  10.0.0.2 (10.0.0.2)  25.464 ms  25.822 ms  26.342 ms
 2  10.0.0.1 (10.0.0.1)  34.868 ms  35.367 ms  35.895 ms
 3  * * *
 4 172.17.54.56 (172.17.54.56)  37.117 ms 172.17.54.72 (172.17.54.72)  38.336 ms 172.17.54.58 (172.17.54.58)  38.979 ms
 5 172.17.54.154 (172.17.54.154)  40.417 ms 172.17.54.144 (172.17.54.144)  40.940 ms 172.17.54.146 (172.17.54.146)  41.496 ms
 6 172.19.177.36 (172.19.177.36)  42.593 ms 11.310 ms 172.19.177.42 (172.19.177.42)  33.373 ms
 7 172.19.177.2 (172.19.177.2)  43.640 ms 172.19.177.4 (172.19.177.4)  48.458 ms 172.19.177.2 (172.19.177.2)  46.018 ms
 8 etrunk49.milano50.mil.seabone.net (195.22.205.116)  46.524 ms etrunk49.milano1.mil.seabone.net (195.22.205.98)  41.217 ms 44.553 ms
 9 74.125.51.148 (74.125.51.148)  50.457 ms 74.125.146.168 (74.125.146.168)  51.004 ms 72.14.195.206 (72.14.195.206)  49.847 ms
10 * 108.170.245.81 (108.170.245.81)  53.499 ms 108.170.245.65 (108.170.245.65)  52.928 ms
11 172.253.79.29 (172.253.79.29)  47.588 ms dns.google (8.8.8.8)  51.396 ms 53.892 ms
```

