



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Programmazione di Reti Laboratorio #8

Andrea Piroddi

Dipartimento di Informatica, Scienza e Ingegneria

Trasferimento di file crittografati tramite socket in Python



CRITTOGRAFIA AES (Advanced Encryption Standard)

L'algoritmo di crittografia simmetrica più popolare e ampiamente adottato è l'Advanced Encryption Standard (AES).

Le caratteristiche di AES sono le seguenti:

- Cifrario a blocchi simmetrico a chiave simmetrica
- Dati a 128 bit, chiavi a 128/192/256 bit



CRITTOGRAFIA AES (Advanced Encryption Standard)

AES è un cifrario iterativo. Si basa su "sostituzione-permutazione". Comprende una serie di operazioni collegate, alcune delle quali comportano la sostituzione di input con output specifici (sostituzioni) e altre implicano il mescolamento di bit (permutazioni).

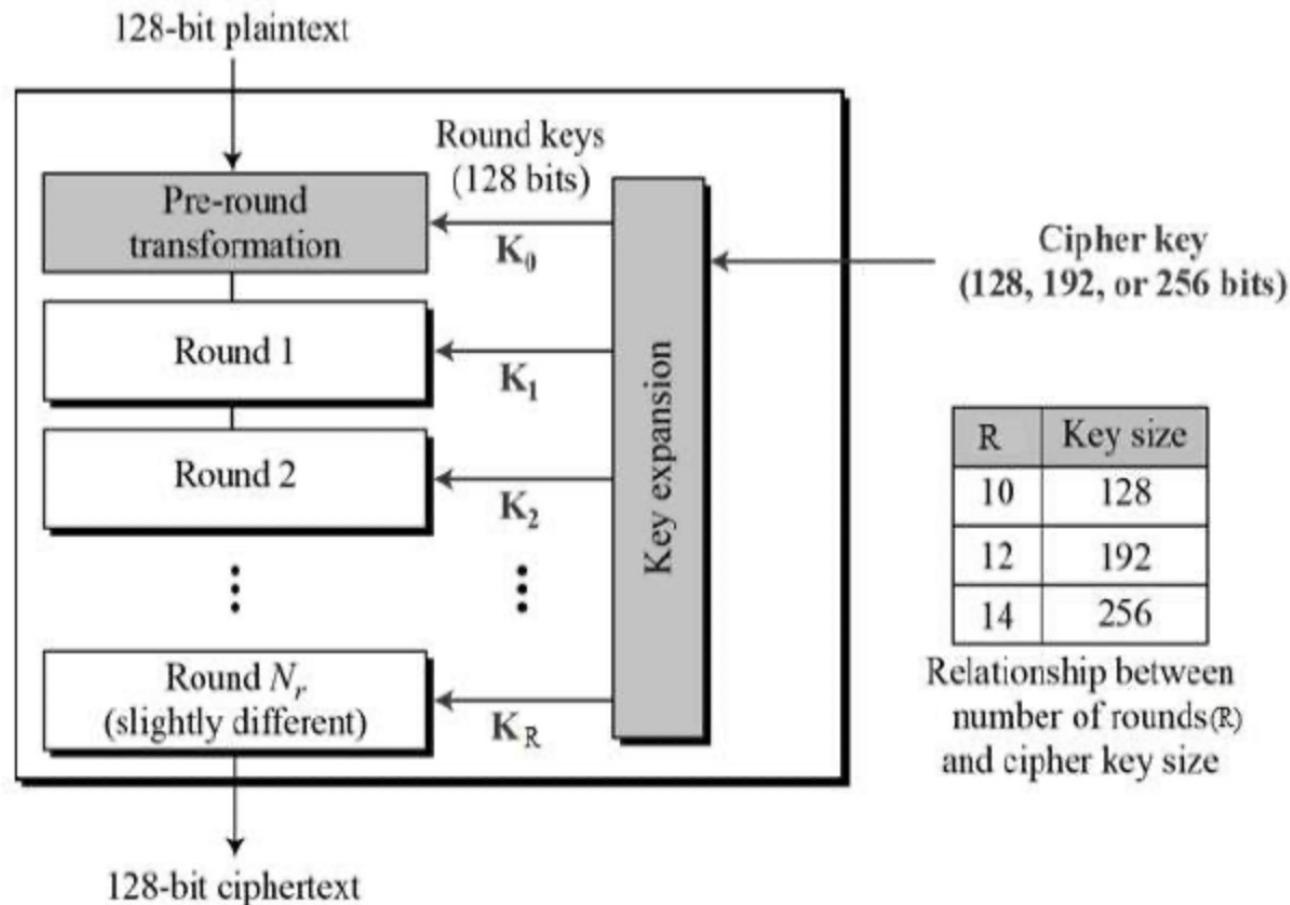
È interessante notare che AES esegue tutti i suoi calcoli su byte anziché su bit. Pertanto, AES tratta i 128 bit di un blocco di testo in chiaro come 16 byte. Questi 16 byte sono disposti in quattro colonne e quattro righe per l'elaborazione come matrice.

Il numero di cicli in AES è variabile e dipende dalla lunghezza della chiave. AES utilizza 10 round per chiavi a 128 bit, 12 round per chiavi a 192 bit e 14 round per chiavi a 256 bit. Ciascuno di questi cicli utilizza una diversa chiave a 128 bit, che viene calcolata dalla chiave AES originale.



CRITTOGRAFIA AES (Advanced Encryption Standard)

Lo schema della struttura AES è riportato nell'illustrazione seguente:



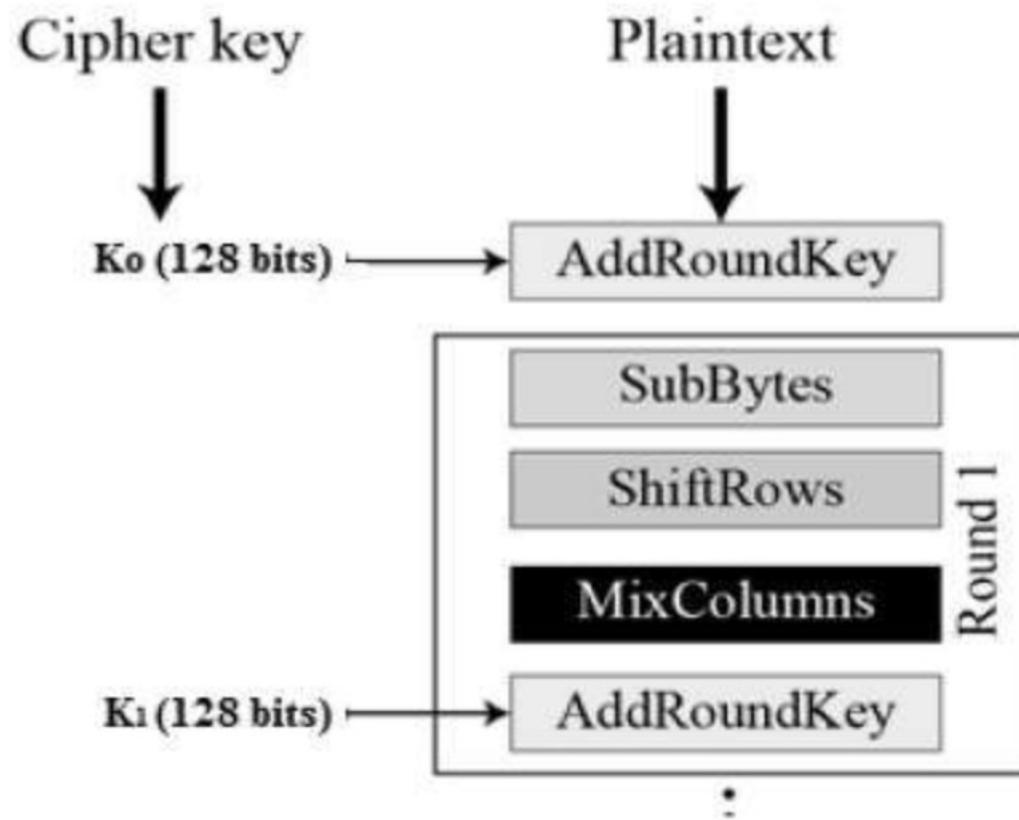
R	Key size
10	128
12	192
14	256

Relationship between
number of rounds(R)
and cipher key size



CRITTOGRAFIA AES (Advanced Encryption Standard)

Ogni round comprende quattro sottoprocessi. Il processo del primo round è illustrato di seguito



Trasferimento di file crittografati tramite socket in Python

Il trasferimento di file in modo sicuro è una funzionalità indispensabile in qualsiasi ambiente di sviluppo.

La crittografia end-to-end garantisce che nemmeno il server possa leggere il contenuto dei dati.

Il trasferimento di file crittografati su socket in Python è un metodo per inviare in modo sicuro file da un computer a un altro.

Implica la creazione di una connessione socket tra il client e il server e la crittografia del file sul lato client prima di inviarlo al server.

Il server quindi decriptografa il file e lo salva nella posizione desiderata.



Trasferimento di file crittografati tramite socket in Python

Il processo può essere suddiviso in diverse fasi:

1. Il client stabilisce una connessione con il server utilizzando il modulo socket.
2. Il client legge il file da trasferire e lo crittografa utilizzando una libreria come ***pycrypto*** o ***cryptography***.
3. Il file crittografato viene inviato al server tramite la connessione socket stabilita.
4. Lato server, il file crittografato ricevuto viene decrittografato utilizzando la stessa libreria di crittografia e salvato nella posizione desiderata.



Trasferimento di file crittografati tramite socket in Python

È importante notare che la chiave di crittografia deve essere scambiata in modo sicuro tra il client e il server prima del trasferimento.

Ciò può essere ottenuto tramite un protocollo di scambio di chiavi sicuro come Diffie-Hellman.

È anche importante utilizzare un algoritmo di crittografia robusto come AES per garantire che il file sia protetto durante la trasmissione.

Inoltre, è anche importante gestire correttamente gli errori, i trasferimenti di file di grandi dimensioni e la gestione delle chiavi.



Trasferimento di file crittografati tramite socket in Python – Diffie-Hellmann

Si considera inizialmente un numero g , generatore del gruppo moltiplicativo degli interi modulo p , dove p è un numero primo.

Uno dei due interlocutori, ad esempio Alice, sceglie un numero casuale " a " e calcola il valore $A = g^a \text{ mod } p$ (dove *mod* indica l'operazione modulo, ovvero il resto della divisione intera) e lo invia attraverso il canale pubblico a Bob (l'altro interlocutore), assieme ai valori g e p .

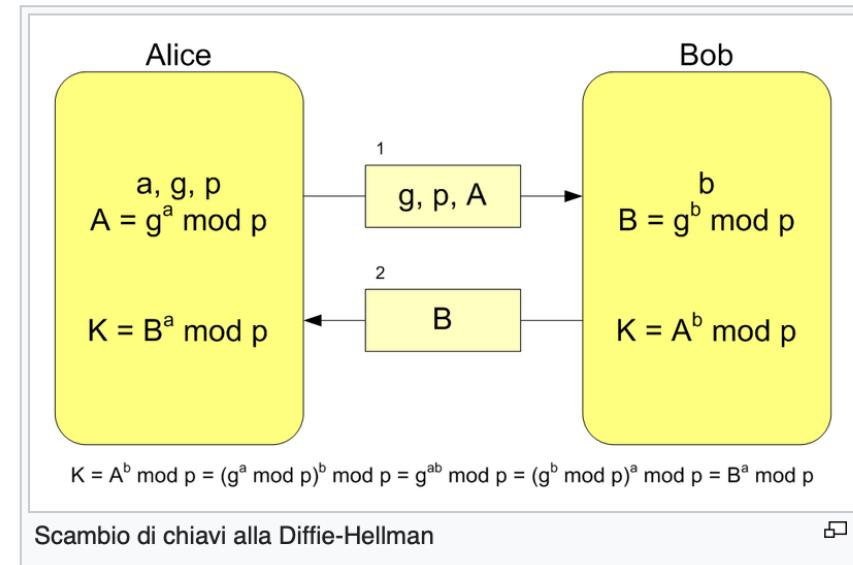
Bob da parte sua sceglie un numero casuale " b ", calcola $B = g^b \text{ mod } p$ e lo invia ad Alice.

A questo punto Alice calcola $K_A = B^a \text{ mod } p$, mentre Bob calcola $K_B = A^b \text{ mod } p$.

I valori calcolati sono gli stessi, in quanto $B^a \text{ mod } p = A^b \text{ mod } p$.

A questo punto i due interlocutori sono entrambi in possesso della chiave segreta e possono cominciare ad usarla per cifrare le comunicazioni successive.

Un attaccante può ascoltare tutto lo scambio, ma per calcolare i valori a e b avrebbe bisogno di risolvere l'operazione del logaritmo discreto, che è computazionalmente onerosa e richiede parecchio tempo, in quanto sub-esponenziale (sicuramente molto più del tempo di conversazione tra i 2 interlocutori).



Trasferimento di file crittografati tramite socket in Python

Per questo trasferimento di file crittografato utilizzeremo la crittografia simmetrica, ciò significa sostanzialmente che utilizzeremo la stessa chiave per la crittografia e per la decrittografia.

Questo metodo è diverso dalla crittografia RSA in cui abbiamo chiavi pubbliche e private, dove la chiave pubblica è utilizzata per la crittografia mentre la chiave privata è utilizzata per la decrittografia



Trasferimento di file crittografati tramite socket in Python

Criptography :

Criptography è un pacchetto che fornisce primitive crittografiche agli sviluppatori Python. Supporta Python 3.7+ e PyPy3 7.3.11+.

Criptography include sia interfacce di alto livello che interfacce di basso livello per algoritmi crittografici comuni come cifrari simmetrici, digest di messaggi e funzioni di derivazione di chiavi.



Trasferimento di file crittografati tramite socket in Python

TQDN:

TQDN è una libreria Python che fornisce una barra di avanzamento rapida ed estensibile per loop e tabelle. È progettato per fornire un feedback visivo sull'avanzamento di un ciclo o di un'iterazione e può essere utilizzato per tenere traccia dell'avanzamento di attività quali trasferimenti di file, elaborazione dati e altro.



Trasferimento di file crittografati tramite socket in Python

Ora andiamo a creare due script uno per il mittente e uno per il destinatario.

Step 1: importare la libreria richiesta

Step 2: creare la chiave

Step 3: crea un codice

Step 4: creare un socket TCP per la comunicazione Internet

Step 5: connettersi all'host locale in cui è ospitato il server ricevente

Step 6: Calcola la dimensione dei file da inviare

Step 7: caricare i dati sotto forma di byte

Step 8: crittografare i dati

Step 9: inviare al destinatario il nome del file, la dimensione del file, il file crittografato e un tag di chiusura per indicare che il file è stato ricevuto completamente.



Trasferimento di file crittografati tramite socket in Python - SERVER

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  Created on Tue May  7 10:55:48 2024
5
6  @author: apirodd
7  """
8
9  import socket
10 import tqdm
11 import os
12 from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
13 from cryptography.hazmat.primitives import hashes
14 from cryptography.hazmat.backends import default_backend
15
16 # definiamo le chiavi
17 key = b"TheBestRandomKey"
18 nonce = b"TheBestRandomNce"
19
20 # crea un oggetto AES cipher
21 backend = default_backend()
22 cipher = Cipher(algorithms.AES(key), modes.GCM(nonce), backend=backend)
23
24 filename1 = "Cartel3.csv"
25 # IP address del server ricevente
26 SERVER_HOST = "127.0.0.1"
27 SERVER_PORT = 5001
28 # riceve 4096 bytes ogni tranne
29 BUFFER_SIZE = 1024 * 4
30 SEPARATOR = "<SEPARATOR>"
31 # creiamo il socket del server
32 # TCP socket
33 s = socket.socket()
34 # facciamo il bind del socket all'ip address locale
35 s.bind((SERVER_HOST, SERVER_PORT))
36
37 # abilitiamo il nostro server ad accettare connessioni
38 # 5 è il numero di connessioni in attesa che il sistema
39 # processerà prima, le altre saranno rifiutate
40
41 s.listen(5)
42 print(f"[*] Listening as {SERVER_HOST}:{SERVER_PORT}")
43 # accetta la connessione se presente
```



Trasferimento di file crittografati tramite socket in Python - SERVER

```
44     client_socket, address = s.accept()
45     # se il codice che segue viene eseguito significa che il mittente è connesso
46     print(f"[+] {address} is connected.")
47     # riceve le informazioni sul file
48     # riceve usandi il socket del client, non il socket del server
49
50     received = client_socket.recv(BUFFER_SIZE).decode("utf-8", "ignore")
51     pippo = [filename, filesize] = received.split(SEPARATOR)
52     # rimuove il percorso assoluto se presente
53
54     filename = os.path.basename(pippo[0])
55     # converte in un intero
56     filesize = int(pippo[1])
57     # comincia a ricevere il file dal socket
58     # e lo scrive sul file stream
59
60     progress = tqdm.tqdm(range(filesize), f"Receiving {filename}", unit="B", unit_scale=True, unit_divisor=1024)
61     with open(filename1, "wb") as f:
62         decryptor = cipher.decryptor()
63         while True:
64             # legge 1024 bytes dal socket (receive)
65             bytes_read = client_socket.recv(BUFFER_SIZE)
66             if not bytes_read:
67                 # non riceve nulla
68                 # la trasmissione del file è completa
69                 break
70
71             # decifra i bytes ricevuti
72             # decrypted_bytes = decryptor.update(bytes_read)
73             # scrive nel file i bytes che abbiamo ricevuto
74             f.write(bytes_read)
75             # aggiorna la barra di avanzamento
76             progress.update(len(bytes_read))
77             # progress.update(len(decrypted_bytes))
78             if bytes_read[-5:] == b"<END>":
79                 break
80
81     # chiude il client socket
82     client_socket.close()
83     # chiude il server socket
84     s.close()
```



Trasferimento di file crittografati tramite socket in Python - CLIENT

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  Created on Tue May  7 12:21:51 2024
5
6  @author: apirodd
7
8
9  import socket
10 import tqdm
11 import os
12 from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
13 from cryptography.hazmat.backends import default_backend
14
15 key = b"TheBestRandomKey"
16 nonce = b"TheBestRandomNce"
17
18 # crea un oggetto AES cipher
19 backend = default_backend()
20 cipher = Cipher(algorithms.AES(key), modes.GCM(nonce), backend=backend)
21
22 SEPARATOR = "<SEPARATOR>"
23 BUFFER_SIZE = 1024 * 4 # 4KB
24
25 def send_file(filename, host, port):
26     # recuperiamo la dimensione del file
27     filesize = os.path.getsize(filename)
28     # creiamo il socket del client
29     s = socket.socket()
30     print(f"[+] Connecting to {host}:{port}")
31     s.connect((host, port))
32     print("[+] Connected.")
33
34     # inviamo il nome del file e la dimensione del file
35     s.send(f"{filename}{SEPARATOR}{filesize}".encode())
```



Trasferimento di file crittografati tramite socket in Python - CLIENT

```
34     # inviamo il nome del file e la dimensione del file
35     s.send(f"{filename}{SEPARATOR}{filesize}".encode())
36
37     # cominciamo ad inviare il file
38     progress = tqdm.tqdm(range(filesize), f"Sending {filename}", unit="B", unit_scale=True, unit_divisor=1024)
39     with open(filename, "rb") as f:
40         encryptor = cipher.encryptor()
41         while True:
42             # leggiamo i bytes dal file
43             bytes_read = f.read(BUFFER_SIZE)
44             if not bytes_read:
45                 # la trasmissione del file è completata
46                 break
47             # criptiamo i dati
48             encrypted = encryptor.update(bytes_read)
49             s.sendall(encrypted)
50             # aggiorniamo la barra di avanzamento
51             progress.update(len(bytes_read))
52             s.send(b"<END>")
53     # chiudiamo il socket
54     s.close()
55
56 if __name__ == "__main__":
57     import argparse
58     parser = argparse.ArgumentParser(description="Simple File Sender")
59     parser.add_argument("file", help="File name to send")
60     parser.add_argument("host", help="The host/IP address of the receiver")
61     parser.add_argument("-p", "--port", help="Port to use, default is 5001", default=5001)
62     args = parser.parse_args()
63     filename = args.file
64     host = args.host
65     port = int(args.port)
66     send_file(filename, host, port)
```

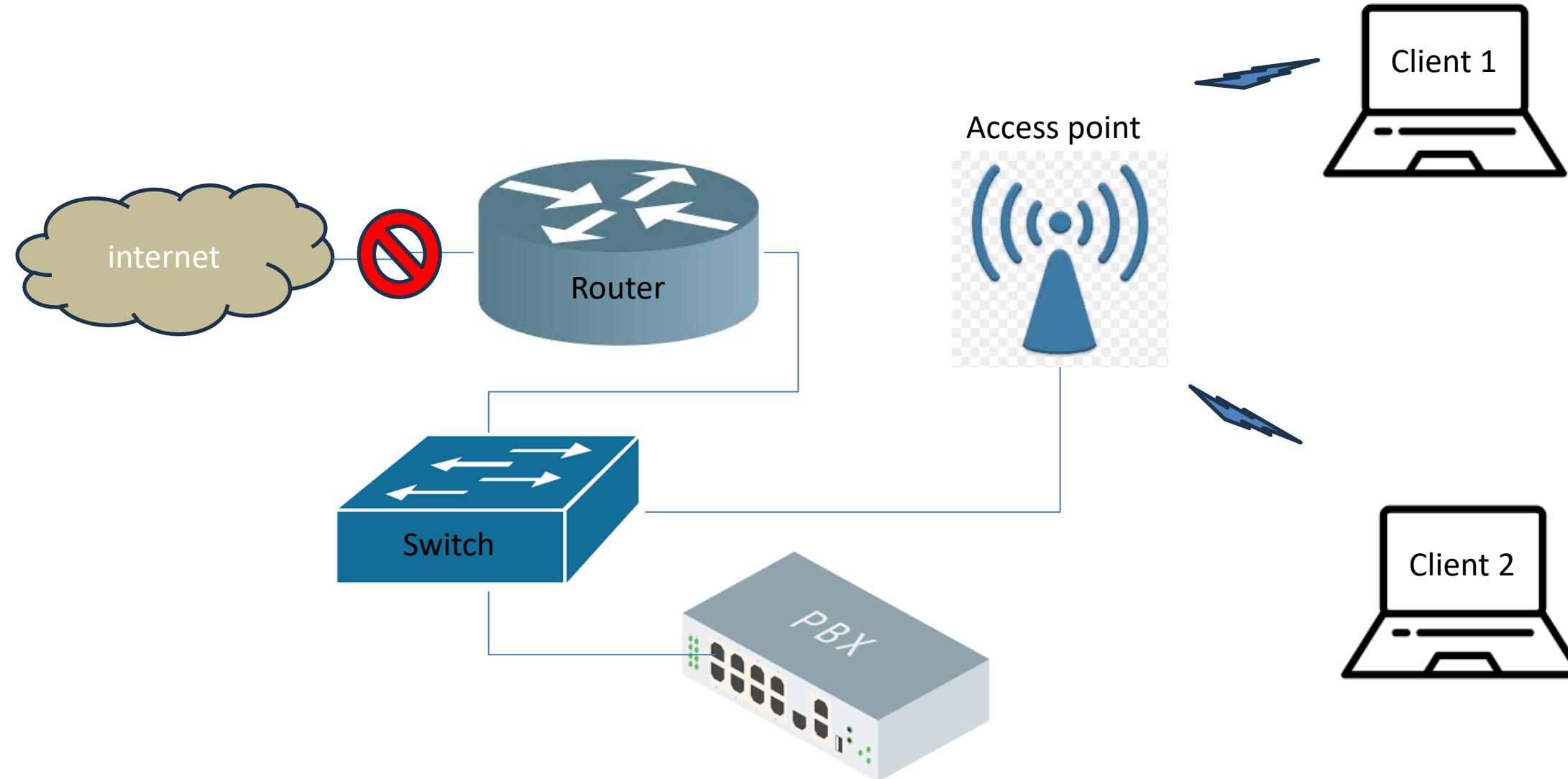


VoIP

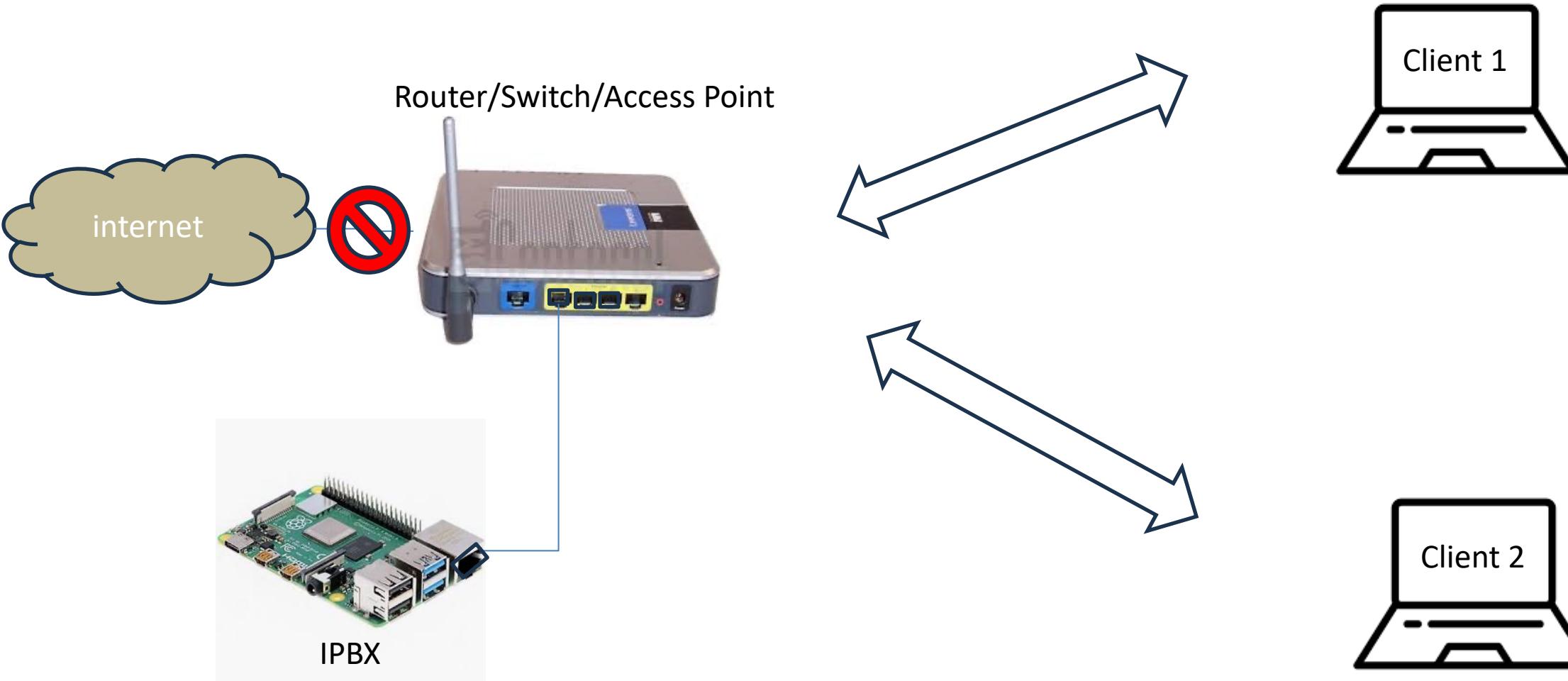


ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

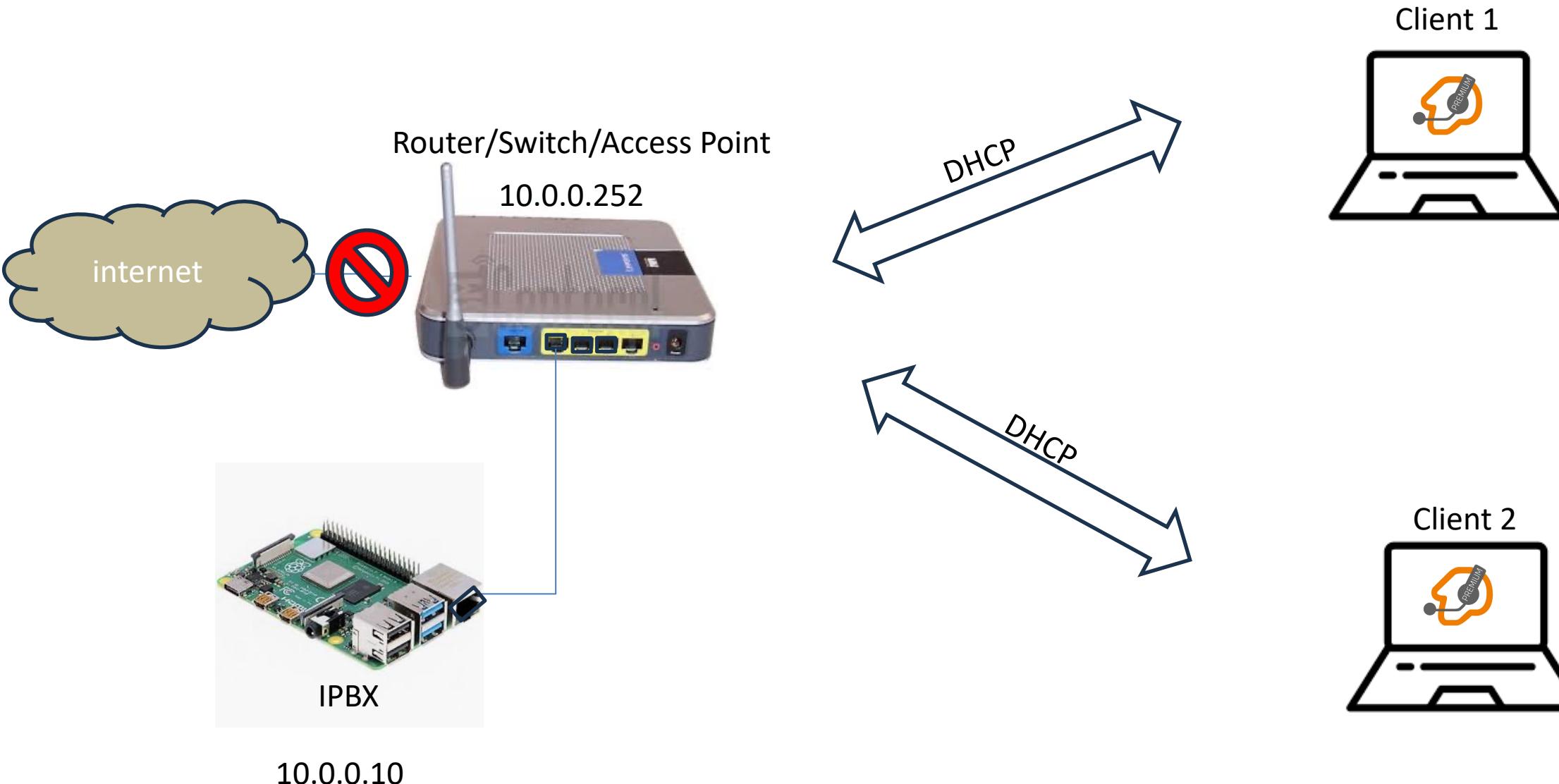
Test Bed VoIP



Test Bed VoIP



Test Bed VoIP

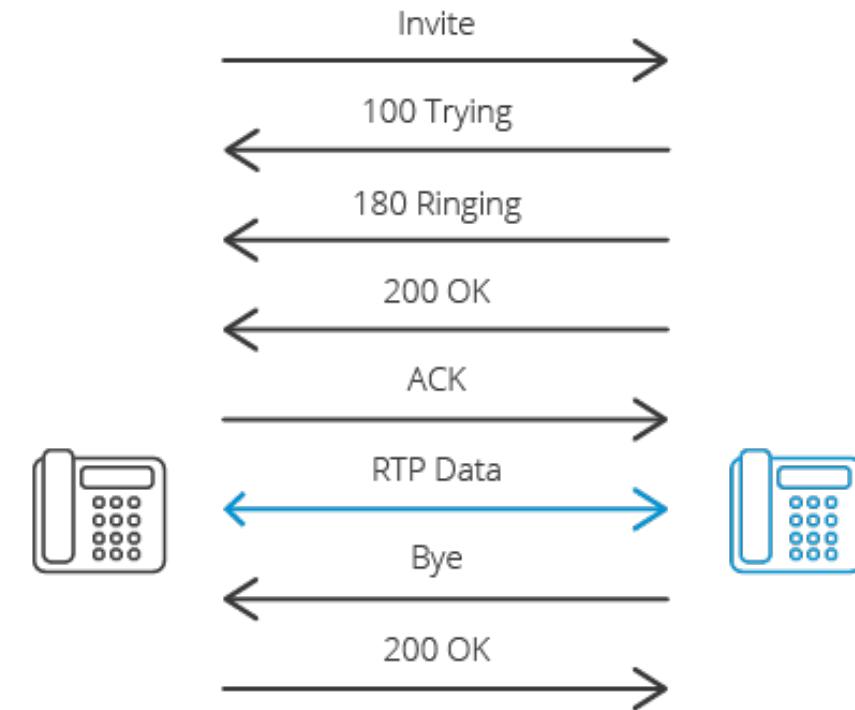


VoIP - Esempio di Sessione di Chiamata SIP tra 2 Telefoni

Le chiamate SIP sono il processo di trasmissione delle chiamate attraverso un Trunk SIP o un canale SIP.

Un Trunk SIP è l'equivalente ad oggi di un Trunk T1. Dove in passato si acquistava un Trunk T1 da un provider Telco e lo si collegava al proprio PBX legacy, oggi è possibile acquistare un Trunk SIP da un ITSP (provider di servizi di telefonia Internet) e collegarlo al proprio centralino VoIP / IP. I Trunk SIP utilizzano lo standard SIP. Il nome "Trunk" deriva dal mondo delle telecomunicazioni e significa un gruppo di linee telefoniche.

Le chiamate SIP utilizzano un percorso specifico per collegare le parti. Una sessione di chiamata SIP tra due telefoni viene stabilita come segue:

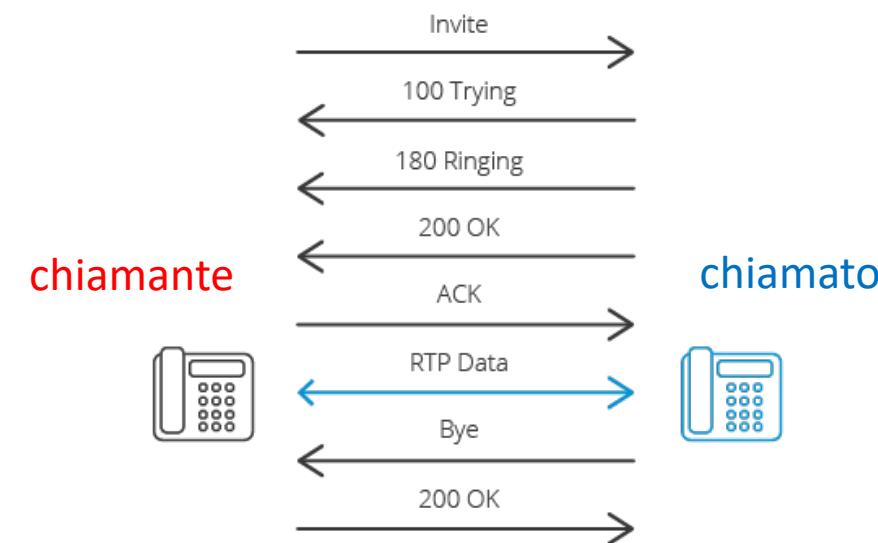
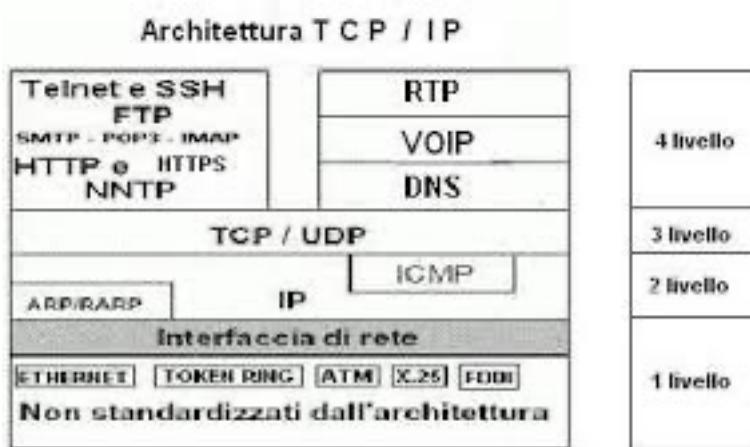


NB: alternativo al SIP è il protocollo H323



VoIP - Esempio di Sessione di Chiamata SIP tra 2 Telefoni

- Il telefono chiamante invia un **INVITO**.
- Il telefono chiamato invia la risposta informativa **100 -Trying – back**.
- Quando il telefono chiamato comincia a squillare, il telefono chiamante riceve la risposta **180 Ringing – sent back**.
- Quando il destinatario alza la cornetta, il telefono chiamato invia la risposta **200 – OK**.
- Il telefono chiamante risponde con **ACK – conferma**.
- L'effettiva conversazione viene quindi trasmessa sottoforma di dati mediante **RTP**.
- Quando il soggetto chiamato riattacca, viene inviata una richiesta **BYE** al telefono chiamante.
- Il telefono chiamante restituisce la risposta **200 – OK**.



VoIP - Che cos'è l' RTP – Real-time Transport Protocol?

RTP è l'acronimo di Real-time Transport Protocol (Protocollo di trasporto in tempo reale) e definisce il formato di un pacchetto standard per la consegna audio e video su internet. Viene descritto nella norma [RFC 1889](#). È stato sviluppato e pubblicato per la prima volta nel 1996.

Come suggerisce il nome, l'obiettivo di progettazione di RTP è lo streaming end-to-end in tempo reale dei dati relativi ai media.

RTP include meccanismi per la **compensazione del jitter**, il **rilevamento della perdita di pacchetti**, etc... particolarmente comuni nelle trasmissioni UDP ([User Datagram Protocol](#)) su IP.



VoIP - Che cos'è l' RTP – Real-time Transport Protocol?

Applicazioni come il VoIP che necessitano di utilizzare lo streaming in tempo reale di dati multimediali, in genere richiedono la consegna tempestiva dei dati, con una tolleranza variabile nella perdita di pacchetti. Ad esempio, la perdita di pacchetti audio in un'app VoIP può causare la perdita di alcuni millisecondi di dati audio. Questa perdita può essere adeguatamente gestita da algoritmi di compensazione degli errori per renderla insignificante e impercettibile per chi chiama.



VoIP - Che cos'è l' RTP – Real-time Transport Protocol?

Anche il TCP (Transmission Control Protocol) è standardizzato per l'uso di RTP, anche se non è tipicamente impiegato nelle app a causa dei suoi meccanismi di controllo degli errori che possono causare ritardi e influenzare la consegna puntuale dei pacchetti. Per questo motivo, la maggior parte delle applicazioni RTP basano le loro implementazioni su UDP.



ASTERIX o ...ASTERISK



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

ASTERISK

Asterisk è un framework open source per la creazione di applicazioni di comunicazione. Trasforma un normale computer in un server di comunicazione.

Alimenta sistemi IP PBX (Private Branch Exchange), gateway VoIP, server per conferenze...



Asterisk

Asterisk è nato come progetto Open Source per la realizzazione di un centralino VoIP e TDM in grado di gestire le moderne comunicazioni VoIP e le interfacce su linee PSTN e ISDN.

Il progetto iniziato nel 2001, è stato sviluppato sotto la direzione di Mark Spencer della Digium.

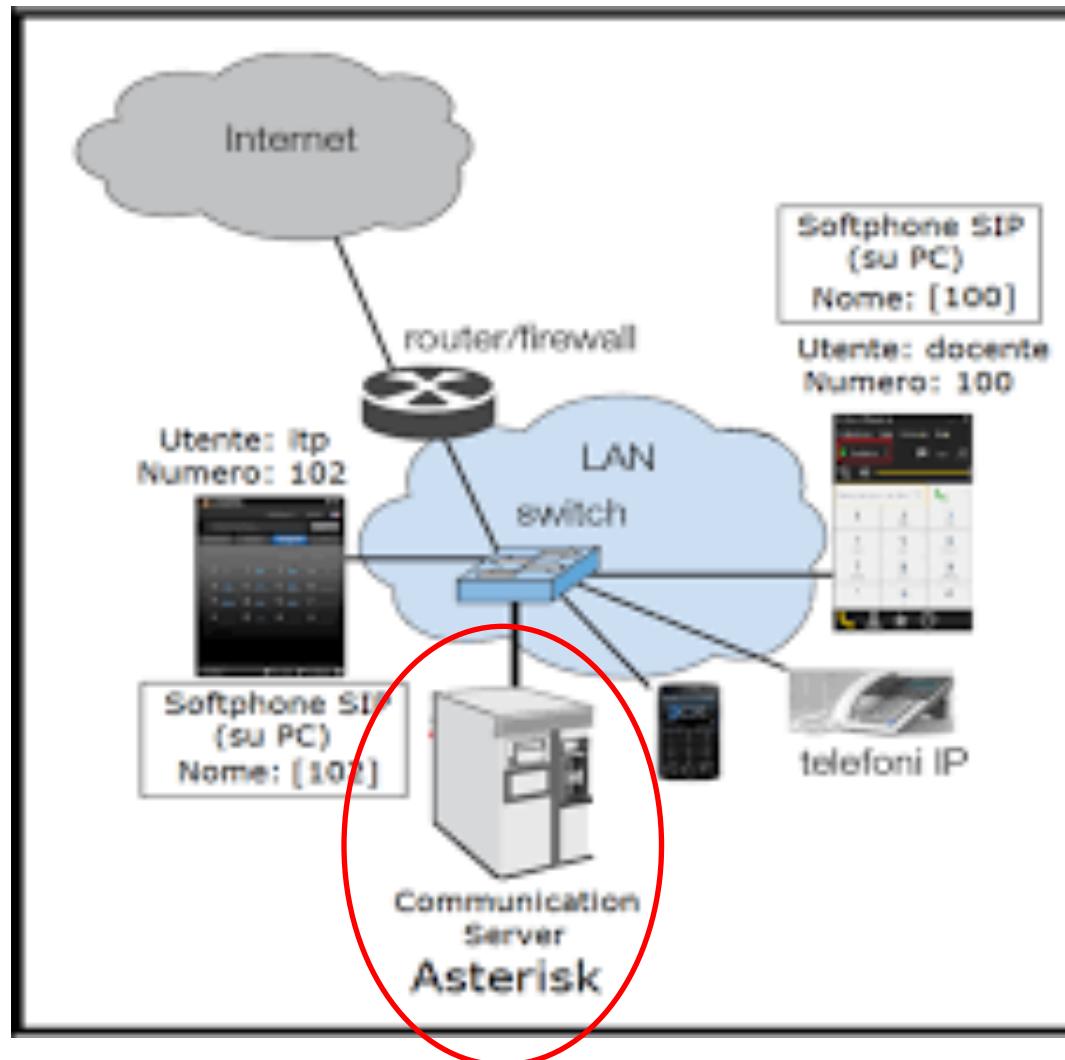
Il suo nome, Asterisk, proviene dal mondo Unix e Dos e rappresenta il cosiddetto “carattere jolly” (*) cioè la possibilità di rappresentare ogni file.

Asterisk è stato progettato per interfacciare qualsiasi tipo di apparato telefonico standard (sia hardware che software) con qualsiasi applicazione telefonica standard, in modo semplice e consistente.

Trattandosi di un progetto completamente Open Source, Asterisk è il favorito nella scelta per lo sviluppo di moltissime soluzioni e viene continuamente migliorato grazie a centinaia di sviluppatori in tutto il mondo.



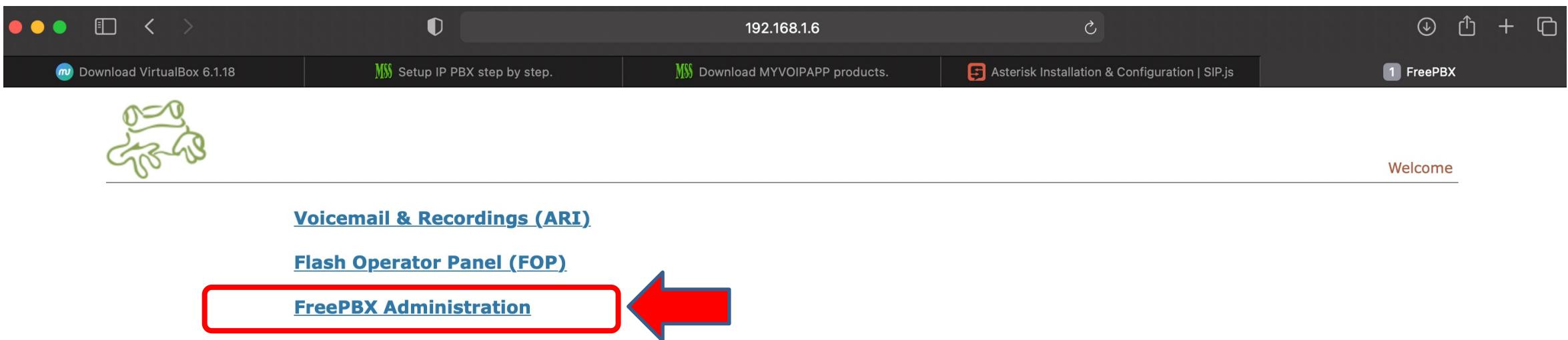
Asterisk - Scenario IP-PBX



Asterisk

Una volta installato e impostato l'indirizzo IP del centralino esso stesso diventa raggiungibile sia da riga di comando che da interfaccia web.

Proviamo a raggiungerlo tramite interfaccia web:



The screenshot shows a Mac OS X desktop environment with a browser window open to the URL `192.168.1.6`. The browser's address bar also displays `192.168.1.6`. The page content is the FreePBX Administration interface. At the top, there are several links: [Download VirtualBox 6.1.18](#), [Setup IP PBX step by step.](#), [Download MYVOIPAPP products.](#), [Asterisk Installation & Configuration | SIP.js](#), and [1 FreePBX](#). On the left side, there is a green frog icon. On the right side, there is a "Welcome" message. Below the links, there are three main navigation buttons: [Voicemail & Recordings \(ARI\)](#), [Flash Operator Panel \(FOP\)](#), and [**FreePBX Administration**](#). The "FreePBX Administration" button is highlighted with a red border and a large red arrow points to it from the bottom left.



Asterisk

The screenshot shows the FreePBX 2.7.0.0 interface on a local IP address. The left sidebar has a red box around the 'Extensions' link. The main content area displays the 'FreePBX System Status' page with several sections:

- FreePBX Notices:**
 - There are 5 modules available for online upgrades
 - Default Asterisk Manager Password Used
- FreePBX Statistics:**

Total active calls	0
Internal calls	0
External calls	0
Total active channels	0
- FreePBX Connections:**

IP Phones Online	5
IP Trunks Online	2
IP Trunk Registrations	2
- Uptime:**

System Uptime: 2 weeks, 3 days, 22 hours, 15 minutes
Asterisk Uptime: 2 weeks, 3 days, 22 hours, 13 minutes
Last Reload: 2 weeks, 3 days, 22 hours, 13 minutes
- System Statistics:**

Processor	
Load Average	0.00
CPU	0%
Memory	
App Memory	20%
Swap	0%
Disks	
/	3%
/boot	10%
/dev/shm	0%
Networks	
eth0 receive	0.32 KB/s
eth0 transmit	1.35 KB/s
- Server Status:**

Asterisk	OK
Op Panel	OK
MySQL	OK
Web Server	OK
SSH Server	OK

Configurazione Interno SIP

Per entrare nel vivo della configurazione la prima cosa da fare è configurare un interno con cui chiamare.

Nel nostro caso prenderemo in esame solo interni, e quindi softphones/telefoni IP, che utilizzino il protocollo SIP e non configureremo le caselle vocali (o segreteria telefonica via email). Le informazioni di cui avremo bisogno sono il numero d'interno e una password.

Per consentire al Telefono di potersi registrare sul Centralino VoIP sono necessari alcuni passaggi.



Asterisk – Configurazione di un interno

Il primo passo da seguire è quello di configurare l'interno desiderato

Pertanto la prima operazione è quella di entrare nella web interface di asterisk
cliccare sul Tab (a sinistra) “Extensions”
a questo punto comparirà la schermata seguente:

Extension: 203

 Delete Extension 203

Edit Extension

Display Name

203

CID Num Alias

SIP Alias

203

Extension Options

Outbound CID

Ring Time

Default

Call Waiting

Enable

Call Screening

Disable

Pinless Dialing

Disable

Emergency CID

Device Options

This device uses sip technology.

secret	abc345
dtmfmode	rfc2833
canreinvite	no
context	from-internal
host	dynamic
type	friend
nat	never
port	5060
qualify	no
callgroup	
pickupgroup	
disallow	
allow	
dial	SIP/203
accountcode	
mailbox	203@device
deny	0.0.0.0/0.0.0.0
permit	0.0.0.0/0.0.0.0

Recording Options



Asterisk – Configurazione di un interno

dove possiamo inserire lo “User Extension” (es: 203) e la Password di autenticazione ossia la “Secret” (es: abc345).

per il momento lasciamo inalterate il resto delle configurazioni.

Guardando la figura vediamo che in

- **Extension Number** va inserito il numero d'interno, mentre in
- **Display Name** possiamo inserire il nome dell'utilizzatore del softphone oppure il nome del dispositivo o anche semplicemente il numero d'interno stesso.
- In **Secret**, invece, inseriamo la password scelta che poi dovremo utilizzare per autenticare il softphone/telefono IP.
- I campi **Outbond CID** ed **Emergency CID** sono da compilare solo se si vuole sovrascrivere l'ID chiamante quando si chiama attraverso un trunk. (Se si lasciano vuoti l'ID verrà settato dal trunk oppure rimarrà quello dell'interno).
- Una volta premuto il tasto **Submit** la configurazione dell'interno è terminata; se si vuole modificare qualche parametro è possibile farlo cliccando sul numero d'interno della colonna di destra.
Aggiungendo due o più interni e configurando i dispositivi client sarà già possibile effettuare chiamate tra gli interni.



Asterisk – Configurazione di un interno

DTMF mode (Dual Tone Multi Frequency)

Le scelte possibili sono inband, rfc2833, info o auto

- **inband:** invia i toni come audio in banda all'interno del flusso vocale. Il dispositivo su cui si preme il tasto genera i toni DTMF. - Se il codec non è ulaw o alaw, i toni DTMF verranno distorti dalla compressione audio e non verranno riconosciuti.
- **rfc2833:** Questo è un altro metodo in banda, che invia toni DTMF separatamente come pacchetti RTP codificati in modo speciale, distinti dai pacchetti audio, ma all'interno della stessa connessione di rete.
- **info:** Questo è un metodo fuori banda che invia i segnali DTMF all'interno del protocollo SIP su una connessione di rete separata dai flussi.
- **auto:** Asterisk userà rfc2833 per il relè DTMF per impostazione predefinita, ma passerà ai toni audio DTMF se il lato remoto non indica il supporto di rfc2833 in SDP.



Asterisk – Configurazione di un interno

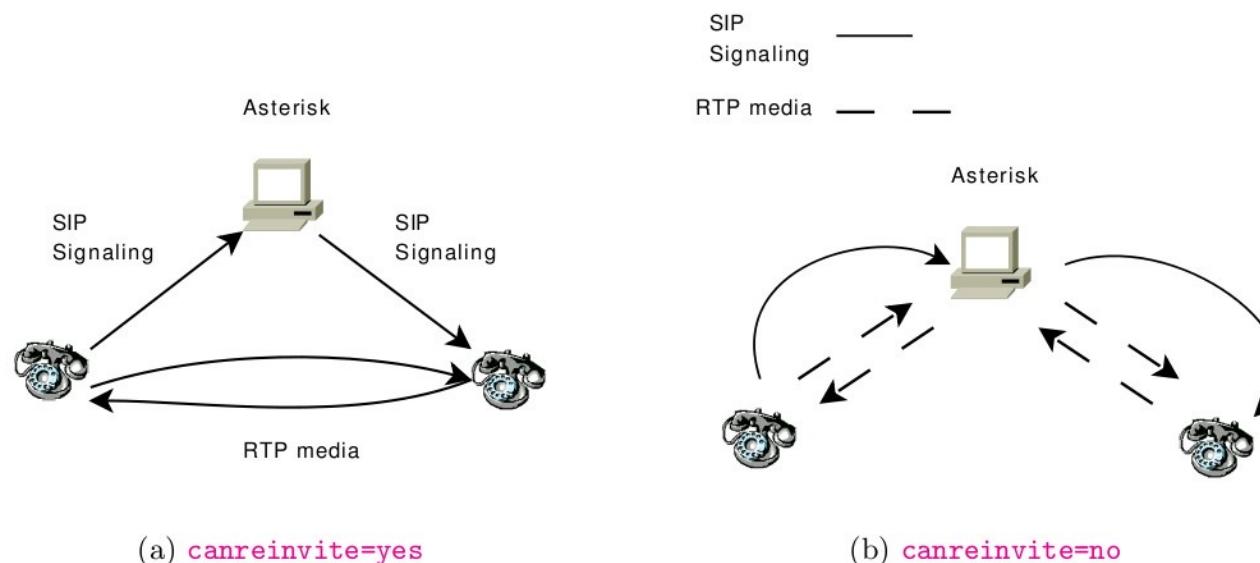
- **canreinvite**: yes|no: Questa opzione di default è settata a *no*: normalmente asterisk non utilizza i messaggi di reinvito.

I messaggi di INVITE all'interno del protocollo sip vengono usati per stabilire comunicazioni. All'interno di un messaggio di INVITE si trovano le informazioni per rendere noto all'altro endpoint dove inviare il media stream della chiamata.



Asterisk – Configurazione di un interno

Asterisk normalmente si pone lui stesso come endpoint della chiamata anche per il flusso RTP ma quando si vuole forzare il passaggio diretto dei pacchetti RTP tra i due endpoint allora Asterisk manda un messaggio di REINVITE ai due endpoint dando loro le informazioni necessarie per la comunicazione diretta.



Configuriamo un Soft phone



Configuriamo un Soft phone

<https://www.zoiper.com/en/voip-softphone/download/current>

Zoiper 5

Free VoIP softphone For non-commercial use

Desktop

 Windows	Download
 Mac	Download
 Linux	Download

Mobile

 Android	Download
iOS 	Download



Configuriamo un Soft phone

Inserite:
CID@ipaddress_centralino:porta



The screenshot shows the Zoiper5 application window. At the top, it says "Zoiper5". On the left, there's a Zoiper logo and the text "Inserite:" followed by a placeholder "CID@ipaddress_centralino:porta". Below this is a large blue arrow pointing to the right. To the right of the placeholder is a text input field containing "203@192.168.1.6:5060". Above the input field, a note says "This will usually look like 23d42a3542 or user@sip.example.com or user@sip.example.com:5060". Below the input field is another text input field with five dots and an eye icon. To the right of this is another blue arrow pointing right. At the bottom are two buttons: a green "Login" button and an orange "Create account" button.

Zoiper5

Zoiper

Inserite:
CID@ipaddress_centralino:porta

203@192.168.1.6:5060

This will usually look like
23d42a3542 or
user@sip.example.com or
user@sip.example.com:5060

.....

Login

Create account

Username and password

If you already have a VoIP account, please enter your login credentials to the left.
Most often these are provided either by your VoIP provider, or by your system administrator for your office's IP PBX.

*we will try to automatically find the necessary configuration for your account. If such does not exist in our database, you'll be asked to enter specific details.



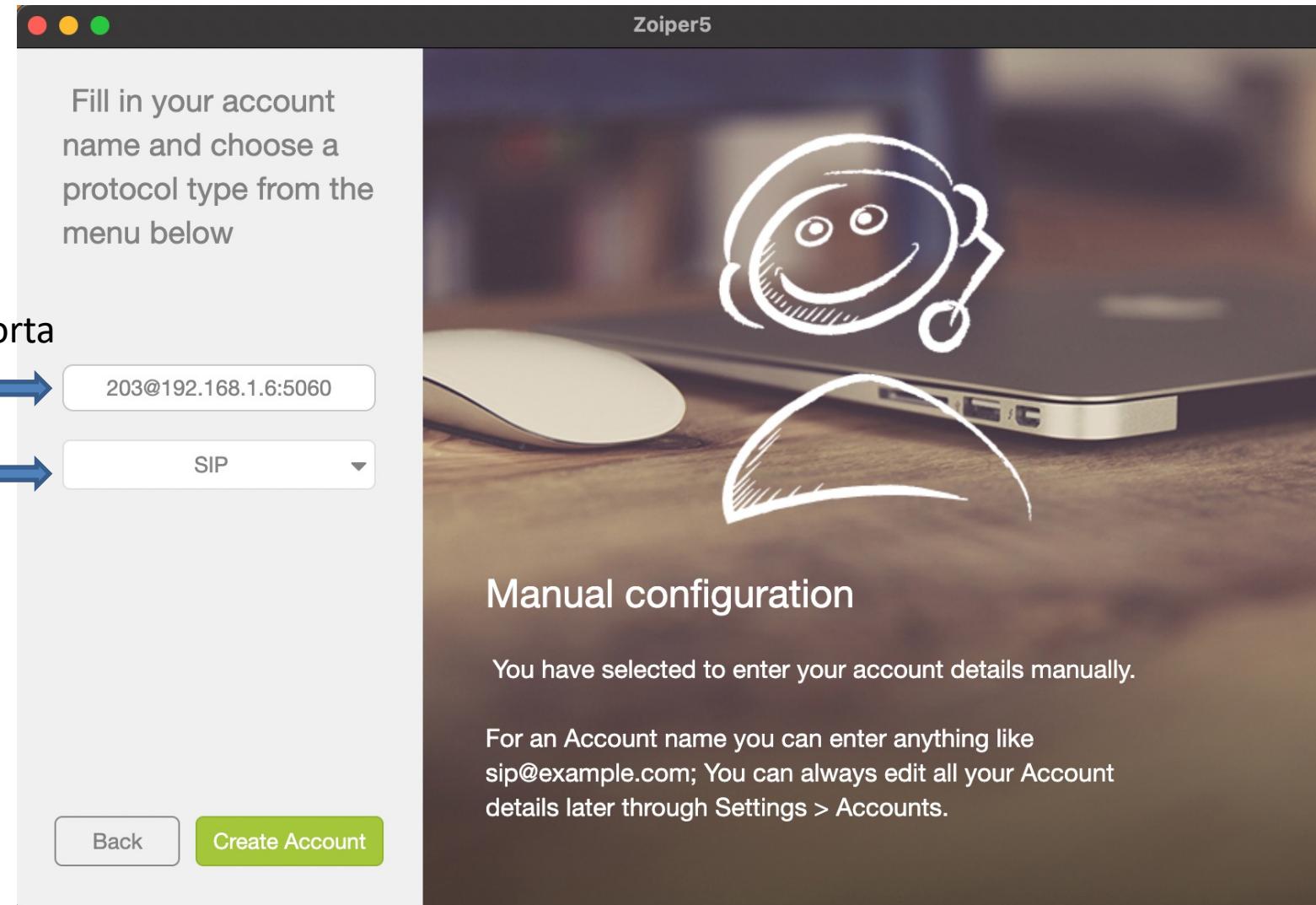
Configuriamo un Soft phone manualmente

Inserite:

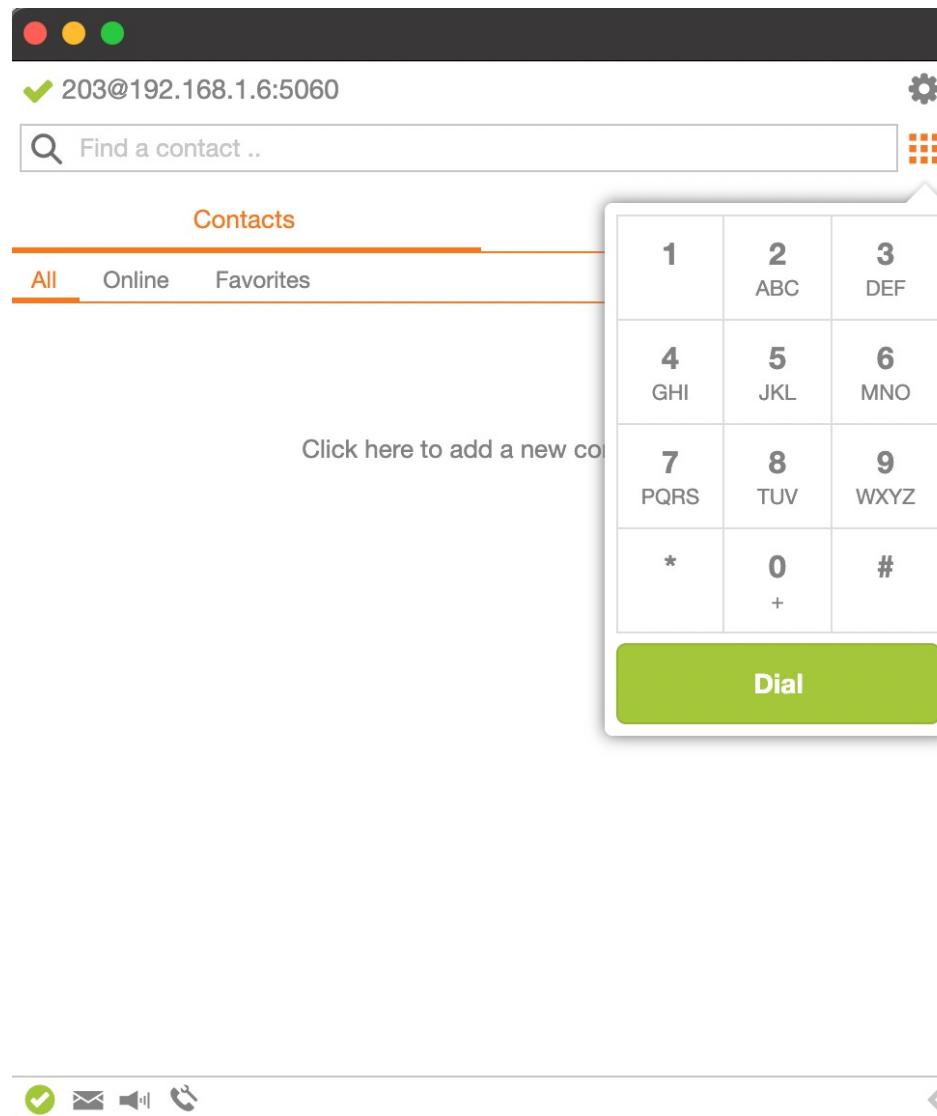
CID@ipaddress_centralino:porta

203@192.168.1.6:5060

SIP



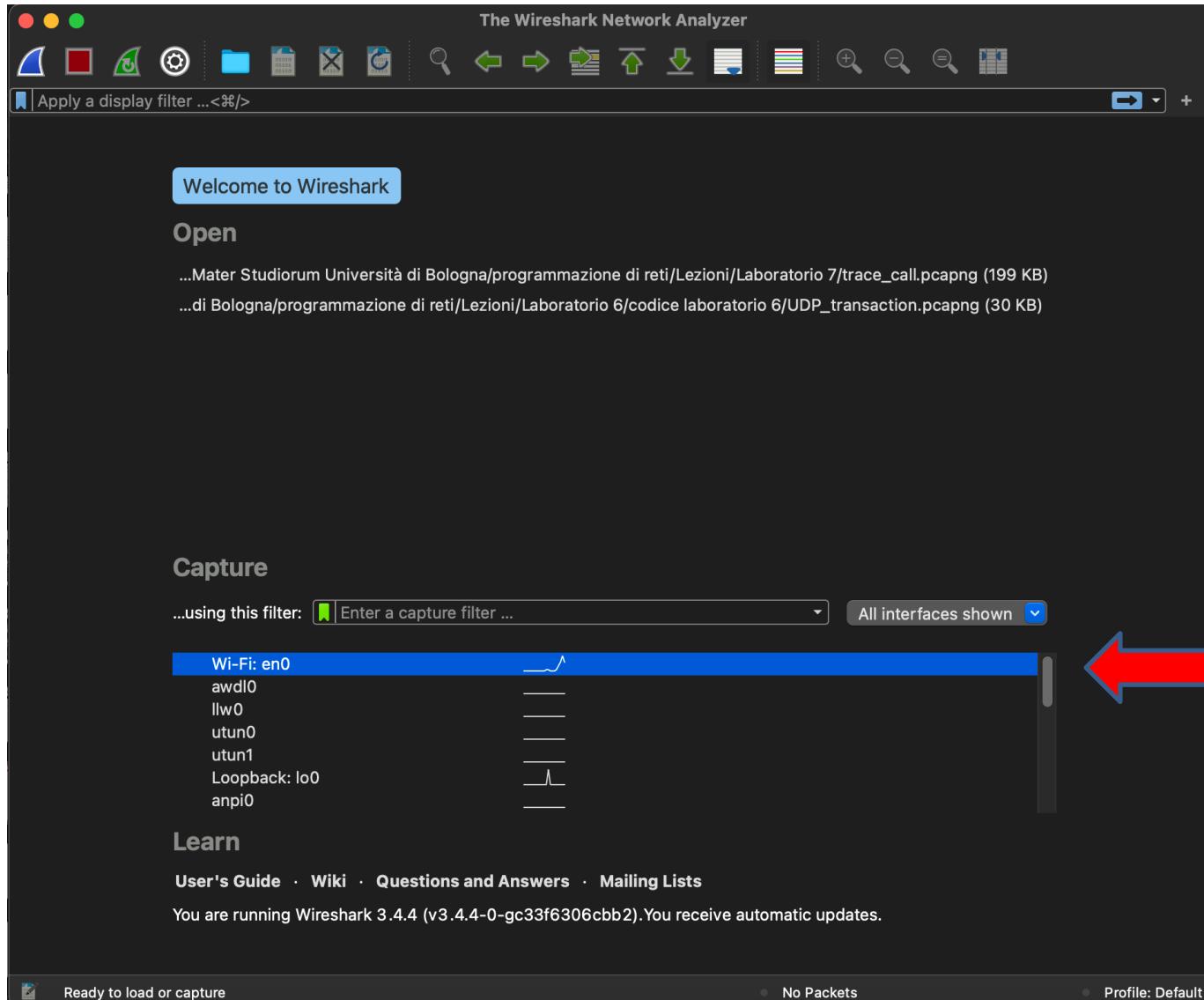
Configuriamo un Soft phone manualmente



Tracciamo ora una chiamata con Wireshark



Tracciamo ora una chiamata con Wireshark



Scegliete l'interfaccia su cui ascoltare



Tracciamo ora una chiamata con Wireshark

Wi-Fi: en0

ip.addr == 192.168.1.6

No.	Time	Source	Destination	Protocol	Length	Info
82	11.858059	192.168.1.124	192.168.1.6	SIP	799	Request: REGISTER sip:192.168.1.6:5060;transport=
83	11.872201	192.168.1.6	192.168.1.124	SIP	583	Status: 401 Unauthorized
84	11.873587	192.168.1.124	192.168.1.6	SIP	799	Request: REGISTER sip:192.168.1.6:5060;transport=
85	11.878238	192.168.1.6	192.168.1.124	SIP	639	Status: 200 OK (1 binding)
329	41.885382	192.168.1.124	192.168.1.6	UDP	46	54960 → 5060 Len=4

```
> Frame 82: 799 bytes on wire (6392 bits), 799 bytes captured (6392 bits) on interface en0, id 0
> Ethernet II, Src: Apple_c5:5c:b1 (74:8f:3c:c5:5c:b1), Dst: HewlettP_56:55:50 (00:12:79:56:55:50)
> Internet Protocol Version 4, Src: 192.168.1.124, Dst: 192.168.1.6
> User Datagram Protocol, Src Port: 54960, Dst Port: 5060
> Session Initiation Protocol (REGISTER)

0000  00 12 79 56 55 50 74 8f  3c c5 5c b1 08 00 45 00  .yVUPt. <\.E.
0010  03 11 d5 c5 00 00 40 11  1e 44 c0 a8 01 7c c0 a8  ..@. D..|..
0020  01 06 d6 b0 13 c4 02 fd  42 82 52 45 47 49 53 54  .....B.REGISTER
0030  45 52 20 73 69 70 3a 31  39 32 2e 31 36 38 2e 31  ER sip:1 92.168.1
0040  2e 36 3a 35 30 36 30 3b  74 72 61 6e 73 70 6f 72  .:5060; transport
0050  74 3d 55 44 50 20 53 49  50 2f 32 2e 30 0d 0a 56  t=UDP SI P/2.0..V
0060  69 61 3a 20 53 49 50 2f  32 2e 30 2f 55 44 50 20  ia: SIP/2.0/UDP
0070  31 39 32 2e 31 36 38 2e  31 2e 31 32 34 3a 35 34  192.168.1.124:54
0080  39 36 30 3b 62 72 61 6e  63 68 3d 7a 39 68 47 34  960;bran ch=z9hG4
0090  62 4b 2d 35 32 34 32 38  37 2d 31 2d 2d 2d 39 32  bK-52428 7-1---92
00a0  38 35 65 36 63 33 36 35  62 32 61 37 37 65 3b 72  85e6c365 b2a77e;r
00b0  70 6f 72 74 0d 0a 4d 61  78 2d 46 6f 72 77 61 72  port Ma x-Forwarding
00c0  64 73 3a 20 37 30 0d 0a  43 6f 6e 74 61 63 74 3a  ds: 70.. Contact:
00d0  20 3c 73 69 70 3a 32 30  33 40 31 39 32 2e 31 36  <sip:20 3@192.16
00e0  38 2e 31 32 34 3a 35 34  39 36 30 3b 72 69 8.1.124: 54960;ri
00f0  6e 73 74 61 6e 63 65 3d  30 61 31 33 38 31 39 36  nstance: 0a138196
0100  39 66 63 39 35 63 34 62  3b 74 72 61 6e 73 70 6f  9fc95c4b ;transpo
0110  72 74 3d 55 44 50 3e 0d  0a 54 6f 3a 20 3c 73 69  rt=UDP> To: <si
0120  70 3a 32 30 33 40 31 39  32 2e 31 36 38 2e 31 2e  p:203@19 2.168.1.
0130  36 3a 35 30 36 30 3b 74  72 61 6e 73 70 6f 72 74  6:5060;t ransport
0140  3d 55 44 50 3e 0d 0a 46  72 6f 6d 3a 20 3c 73 69  =UDP>;F rom: <si
0150  70 3a 32 30 33 40 31 39  32 2e 31 36 38 2e 31 2e  p:203@19 2.168.1.
0160  36 3a 35 30 36 30 3b 74  72 61 6e 73 70 6f 72 74  6:5060;t ransport
0170  3d 55 44 50 3e 3b 74 61  67 3d 61 38 63 63 66 37  =UDP>;ta g=a8ccf7
```

wireshark_Wi-FiP3GA10.pcapng

Packets: 365 · Displayed: 5 (1.4%)

Profile: Default

Inserite un filtro su base
ip_address per esempio
quello del server-VoIP



Tracciamo ora una chiamata con Wireshark

The screenshot shows the Wireshark interface with the following details:

- Interface:** Wi-Fi: en0
- Filter:** ip.addr == 192.168.1.6
- Packets:** 741 · Displayed: 21 (2.8%)
- Profile:** Default

The packet list pane shows several SIP messages between two hosts:

No.	Time	Source	Destination	Protocol	Length	Info
540	65.895736	192.168.1.6	192.168.1.124	SIP	639	Status: 200 OK (1 binding)
670	71.890911	192.168.1.124	192.168.1.6	UDP	46	54960 → 5060 Len=4
671	72.202539	192.168.1.124	192.168.1.6	SIP	575	Request: CANCEL sip:201@192.168.1.6:5060;tr
672	72.217458	192.168.1.6	192.168.1.124	SIP	496	Status: 487 Request Terminated
673	72.217465	192.168.1.6	192.168.1.124	SIP	480	Status: 200 OK
674	72.218212	192.168.1.124	192.168.1.6	SIP	376	Request: ACK sip:201@192.168.1.6:5060;tran

The details pane displays the structure of a selected SIP REGISTER message:

```
> Frame 82: 799 bytes on wire (6392 bits), 799 bytes captured (6392 bits) on interface en0, id 0
> Ethernet II, Src: Apple_c5:c5:b1 (74:8f:3c:c5:c5:b1), Dst: HewlettP_56:55:50 (00:12:79:56:55:50)
> Internet Protocol Version 4, Src: 192.168.1.124, Dst: 192.168.1.6
> User Datagram Protocol, Src Port: 54960, Dst Port: 5060
> Session Initiation Protocol (REGISTER)
```

The bytes pane shows the raw hex and ASCII data of the selected SIP message. A blue arrow points from the text above to the Wireshark window.



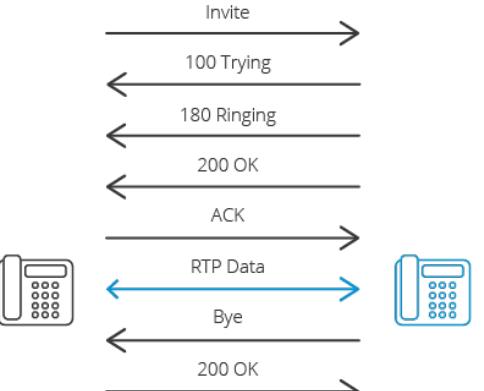
Analizziamo il trace salvato



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Analizziamo il trace salvato

No.	Time	Source	Destination	Protocol	Length	Info
84	11.873587	192.168.1.124	192.168.1.6	SIP	799	Request: REGISTER sip:192.168.1.6:5060
85	11.878238	192.168.1.6	192.168.1.124	SIP	639	Status: 200 OK (1 binding)
329	41.885382	192.168.1.124	192.168.1.6	UDP	46	54960 → 5060 Len=4
502	64.230810	192.168.1.124	192.168.1.6	SIP/S...	940	Request: INVITE sip:201@192.168.1.6:5060;method=INVITE;from=192.168.1.124;to=201@192.168.1.6:5060;rport
503	64.234909	192.168.1.6	192.168.1.124	SIP	566	Status: 401 Unauthorized
504	64.235286	192.168.1.124	192.168.1.6	SIP	376	Request: ACK sip:201@192.168.1.6:5060;rport
505	64.235454	192.168.1.124	192.168.1.6	SIP/S...	1115	Request: INVITE sip:201@192.168.1.6:5060;method=INVITE;from=192.168.1.124;to=201@192.168.1.6:5060;rport
506	64.241592	192.168.1.6	192.168.1.124	SIP	501	Status: 100 Trying
507	64.426752	192.168.1.6	192.168.1.124	SIP	517	Status: 180 Ringing
520	64.821565	192.168.1.6	192.168.1.124	SIP	517	Status: 180 Ringing



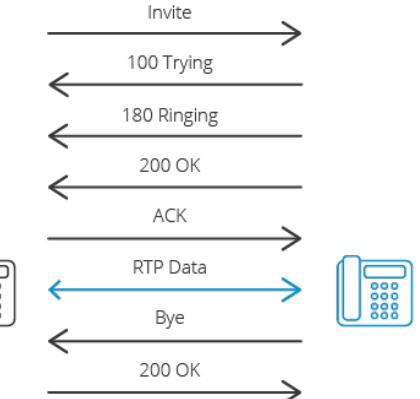
È abbastanza normale avere una risposta 401 iniziale al primo INVITO non autorizzato. Il 401 dovrebbe contenere un'intestazione di autenticazione.

La risposta 401 dovrebbe essere seguita da un secondo INVITE contenente un'intestazione di autorizzazione.



Analizziamo il trace salvato

ip.addr == 192.168.1.6						
No.	Time	Source	Destination	Protocol	Length	Info
121	10.161725	192.168.1.6	192.168.1.124	SIP	517	Status: 180 Ringing
196	10.603058	192.168.1.6	192.168.1.124	SIP	517	Status: 180 Ringing
269	17.298719	192.168.1.124	192.168.1.6	UDP	46	54960 → 5060 Len=4
298	19.870975	192.168.1.6	192.168.1.124	SIP/S...	826	Status: 200 OK
299	19.876520	192.168.1.124	192.168.1.6	RTP	55	PT=Unassigned, SSRC=0x82C4A958, Seq=15254, Time=1604299326
300	19.878113	192.168.1.124	192.168.1.6	SIP	444	Request: ACK sip:200@192.168.1.6
301	19.963657	192.168.1.6	192.168.1.124	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x527662ED, Seq=48016, Time=845808, Mark
302	19.983954	192.168.1.6	192.168.1.124	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x527662ED, Seq=48017, Time=845968
303	20.014362	192.168.1.6	192.168.1.124	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x527662ED, Seq=48018, Time=846128
304	20.023365	192.168.1.6	192.168.1.124	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x527662ED, Seq=48019, Time=846288
305	20.043711	192.168.1.6	192.168.1.124	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x527662ED, Seq=48020, Time=846448
306	20.063281	192.168.1.6	192.168.1.124	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x527662ED, Seq=48021, Time=846608
307	20.083347	192.168.1.6	192.168.1.124	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x527662ED, Seq=48022, Time=846768
308	20.103802	192.168.1.6	192.168.1.124	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x527662ED, Seq=48023, Time=846928
309	20.130562	192.168.1.6	192.168.1.124	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x527662ED, Seq=48024, Time=847088
310	20.143342	192.168.1.6	192.168.1.124	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x527662ED, Seq=48025, Time=847248



No.	Time	Source	Destination	Protocol	Length	Info
618	23.442890	192.168.1.6	192.168.1.124	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x527662ED, Seq=48190, Time=873648
619	23.447177	192.168.1.124	192.168.1.6	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x82C4A958, Seq=15393, Time=1604321406
620	23.463497	192.168.1.6	192.168.1.124	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x527662ED, Seq=48191, Time=873808
621	23.467158	192.168.1.124	192.168.1.6	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x82C4A958, Seq=15394, Time=1604321566
622	23.482914	192.168.1.6	192.168.1.124	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x527662ED, Seq=48192, Time=873968
623	23.487193	192.168.1.124	192.168.1.6	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x82C4A958, Seq=15395, Time=1604321726
624	23.502819	192.168.1.6	192.168.1.124	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x527662ED, Seq=48193, Time=874128
625	23.507194	192.168.1.124	192.168.1.6	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x82C4A958, Seq=15396, Time=1604321886
628	23.522785	192.168.1.6	192.168.1.124	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x527662ED, Seq=48194, Time=874288
629	23.527181	192.168.1.124	192.168.1.6	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x82C4A958, Seq=15397, Time=1604322046
630	23.547250	192.168.1.124	192.168.1.6	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x82C4A958, Seq=15398, Time=1604322206
632	23.567133	192.168.1.124	192.168.1.6	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x82C4A958, Seq=15399, Time=1604322366
637	23.587254	192.168.1.124	192.168.1.6	RTP	214	PT=ITU-T G.711 PCMU, SSRC=0x82C4A958, Seq=15400, Time=1604322526
638	23.595889	192.168.1.6	192.168.1.124	SIP	459	Request: BYE sip:203@192.168.1.124:54960;transport=UDP
639	23.611101	192.168.1.124	192.168.1.6	RTCP	102	Receiver Report Source description Goodbye
640	23.613358	192.168.1.124	192.168.1.6	SIP	382	Status: 200 OK



Esercizi

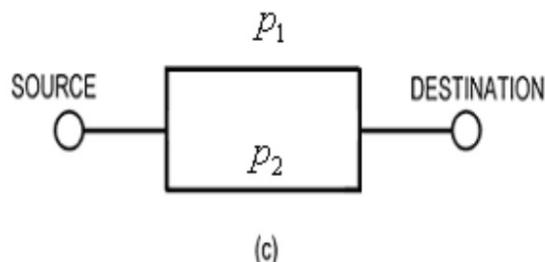
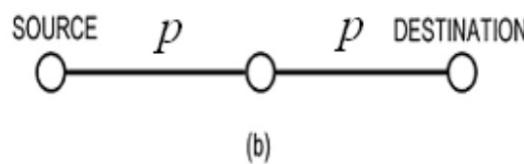
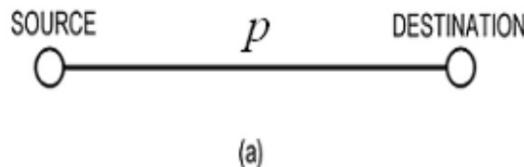


ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Esercizi

Consideriamo i modelli di rete mostrati in Figura.

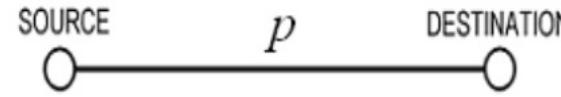
La probabilità che un pacchetto sia danneggiato su un collegamento di rete è p .



Basic network structures: (a) single link; (b) series link; (c) parallel link



Single-Link Network



Si consideri un messaggio di posta elettronica trasmesso su un unico collegamento di rete.

L'e-mail è suddivisa in K pacchetti, ognuno dei quali viene trasmesso individualmente attraverso il collegamento.

La probabilità che una trasmissione fallisca è p ; in caso di trasmissione fallita, il pacchetto viene ritrasmesso.

Introduciamo l'idea della distribuzione di Bernoulli di una variabile casuale osservando una singola trasmissione. Notiamo che il risultato di una singola trasmissione (la variabile casuale X) è un successo o un fallimento.



Quesito 1

Quale è la probabilità che il pacchetto sia trasmesso correttamente dopo n trasmissioni?

- $p^n \cdot (1 - p)$
- $p^{n-1} \cdot (1 - p)$
- p^n
- p^{n-1}



Quesito 1

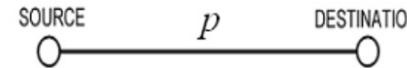
Quale è la probabilità che il pacchetto sia trasmesso correttamente dopo n trasmissioni?

- $p^n \cdot (1 - p)$
- $p^{n-1} \cdot (1 - p)$
- p^n
- p^{n-1}

Motivo: la trasmissione deve fallire $(n - 1)$ volte, ciascuna con probabilità p , e poi riuscire all'ennesimo tentativo.



Single-Link Network

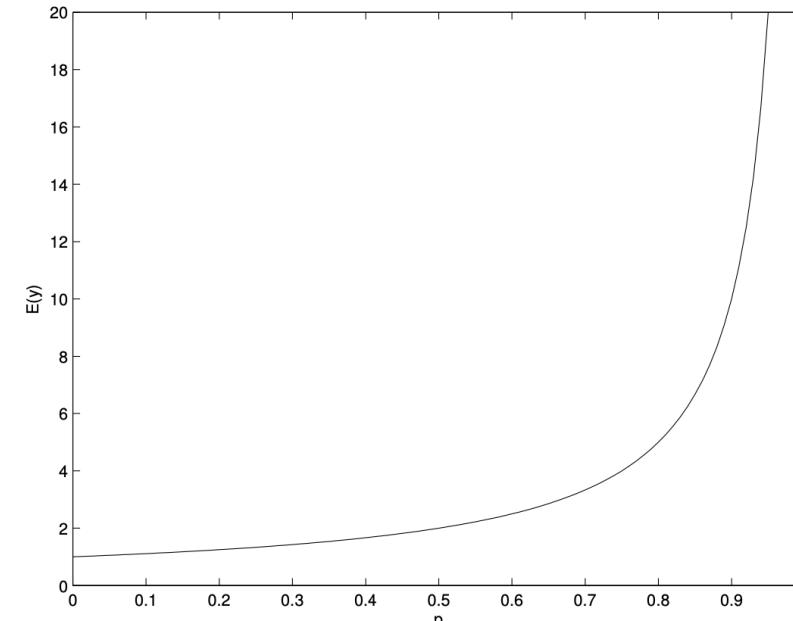


Definiamo la variabile casuale Y come il numero di trasmissioni per avere una trasmissione riuscita di un pacchetto.

Il numero medio di trasmissioni $E(Y)$ per una trasmissione riuscita di un pacchetto è:

$$E(Y) = \sum_{i=0}^{\infty} ip^{i-1}(1 - p) = \frac{1}{(1 - p)}$$

Un rapido sguardo alle condizioni al contorno mostra che se $p = 0$ (tutte le trasmissioni hanno esito positivo), allora il numero previsto di trasmissioni è 1, mentre se $p = 1$ (tutte le trasmissioni falliscono), sono necessarie infinite trasmissioni.



Quesito 2

Supponiamo di volere che il pacchetto attraversi il collegamento dopo non più di due tentativi. Quale è la probabilità che ciò accada?

- $q_{1,2} = (1 - 2p)$
- $q_{1,2} = (2p)$
- $q_{1,2} = (1 - p^2)$
- $q_{1,2} = p^2$



Quesito 2

Supponiamo di volere che il pacchetto attraversi il collegamento dopo non più di due tentativi. Quale è la probabilità che ciò accada?

- $q_{1,2} = (1 - 2p)$
 - $q_{1,2} = (2p)$
 - $q_{1,2} = (1 - p^2)$
 - $q_{1,2} = p^2$
- $(1 - p) + p(1 - p) = (1 - p)(1 + p) = 1 - p^2$

MOTIVO: è la somma delle probabilità che il pacchetto arrivi dopo esattamente uno o esattamente due tentativi o 1 meno la probabilità di due fallimenti in due tentativi). Ad esempio, se $p = 0$ ossia tutti i tentativi hanno successo, è necessaria solo una trasmissione. Tuttavia, se $p = 1$ ossia tutti i tentativi falliscono, la probabilità è zero.



Quesito 3

In generale quale è la probabilità che un pacchetto venga trasmesso con successo attraverso una rete **single-link** al massimo in N tentativi?

- $q_{1,N} = 1 - p^N$
- $q_{1,N} = 1 - p^{N-1}$
- $q_{1,N} = p^N$
- $q_{1,N} = p^{N+1}$

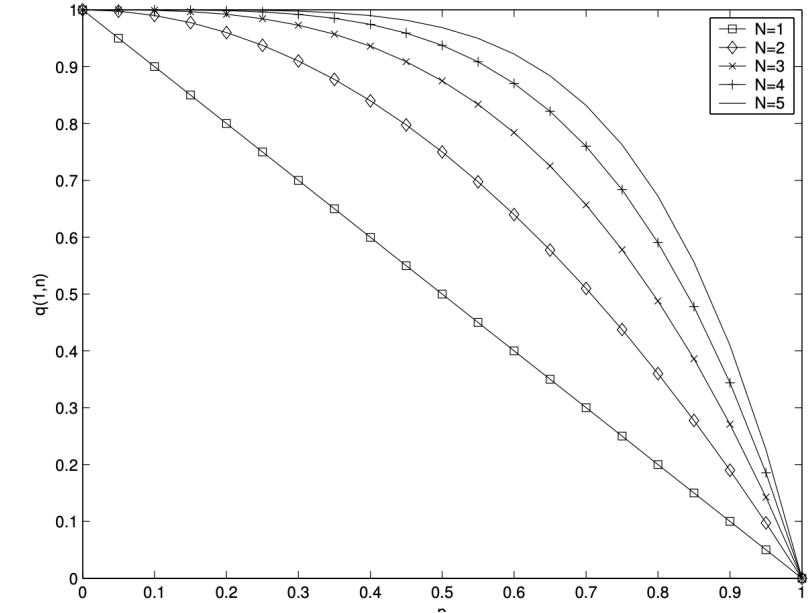


Quesito 3

In generale quale è la probabilità che un pacchetto venga trasmesso con successo attraverso una rete **single-link** al massimo in N tentativi?

- $q_{1,N} = 1 - p^N$
- $q_{1,N} = 1 - p^{N-1}$
- $q_{1,N} = p^N$
- $q_{1,N} = p^{N+1}$

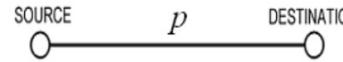
$$q_{1,N} = \sum_{i=1}^N p^{i-1}(1-p) = 1 - p^N$$



OSSERVAZIONE: per valori di p inferiori a uno (una probabilità diversa da zero di una singola trasmissione riuscita), la probabilità che il pacchetto venga eventualmente trasmesso con successo si avvicina a uno quando N aumenta (tende all'infinito).



Single Link: Multiple packets

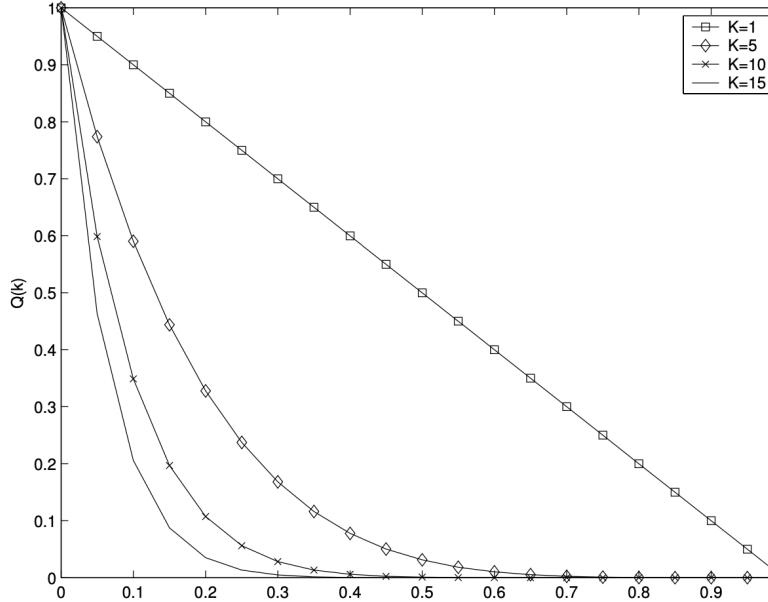


Un messaggio di posta elettronica viene suddiviso in più pacchetti prima di essere trasmesso in rete.

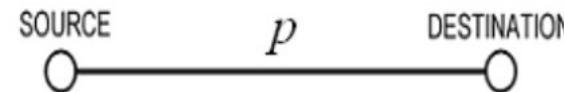
Supponiamo che ci siano K pacchetti, ognuno dei quali viene trasmesso separatamente.

A seconda dei valori di p e K , potrebbe essere molto improbabile che l'intero messaggio venga trasmesso correttamente al primo tentativo. Tutte le trasmissioni di pacchetti devono riuscire, rendendo questa probabilità pari a:

$$Q_K = (1 - p)^K$$



Single Link: Multiple packets



Supponiamo che i pacchetti vengano inviati sulla rete in successione e, se una trasmissione di pacchetti fallisce, il pacchetto fallito viene ritrasmesso fino a quando non si verifica il successo.

I pacchetti trasmessi correttamente prima che un pacchetto successivo fallisca non vengono reinviati.

Definiamo la variabile casuale Z come il numero medio di trasmissioni richieste per una corretta trasmissione dell'intera e-mail.



Quesito 4

Quale è il valore atteso di trasmissioni, $E(Z)$, rispetto a p e K ?

- $(1 - p)^K$
- $(1 - p)^{K-1}$
- $\frac{K}{(1 - p)}$
- $\frac{K}{(1 - p)^{K-1}}$

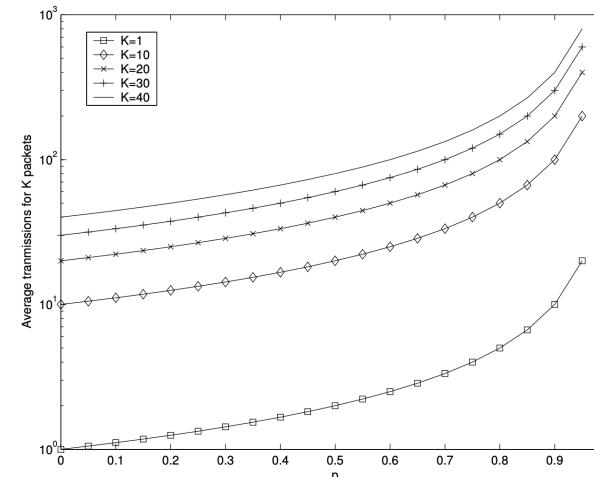


Quesito 4

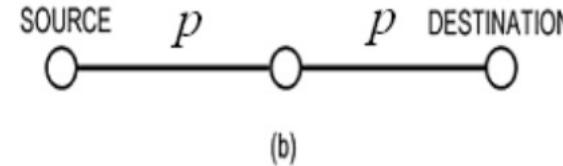
Quale è il valore atteso di trasmissioni, $E(Z)$, rispetto a p e K ?

- $(1 - p)^K$
- $(1 - p)^{K-1}$
- $\frac{K}{(1 - p)}$
- $\frac{K}{(1 - p)^{K-1}}$

MOTIVO: $E(Z)$ può essere calcolato facilmente poiché il successo o il fallimento di qualsiasi trasmissione di pacchetto è indipendente dalle altre $(K - 1)$ trasmissioni, e abbiamo visto nella slide 7 che un singolo pacchetto richiede (in media) $\frac{1}{(1-p)}$ trasmissioni per avere successo, quindi ci aspettiamo che l'insieme completo di K pacchetti venga trasmesso in $\frac{K}{(1-p)}$ tentativi.



Series Two-Link Network



Si supponga che ciascun pacchetto dell'e-mail suddivisa in K pacchetti debba essere trasmesso attraverso due collegamenti di rete collegati in serie.

La probabilità di perdere un pacchetto su qualsiasi collegamento è ancora p .

Durante il processo di trasmissione, se un pacchetto è danneggiato su uno qualsiasi dei due collegamenti, deve essere ritrasmesso nuovamente attraverso il primo collegamento.



Quesito 5

Quale è la probabilità che un pacchetto venga trasmesso correttamente su entrambi i collegamenti nella rete?

- $1 - p^2$
- $(1 - p)^2$
- $(1 - p)p$
- $(1 - p)$

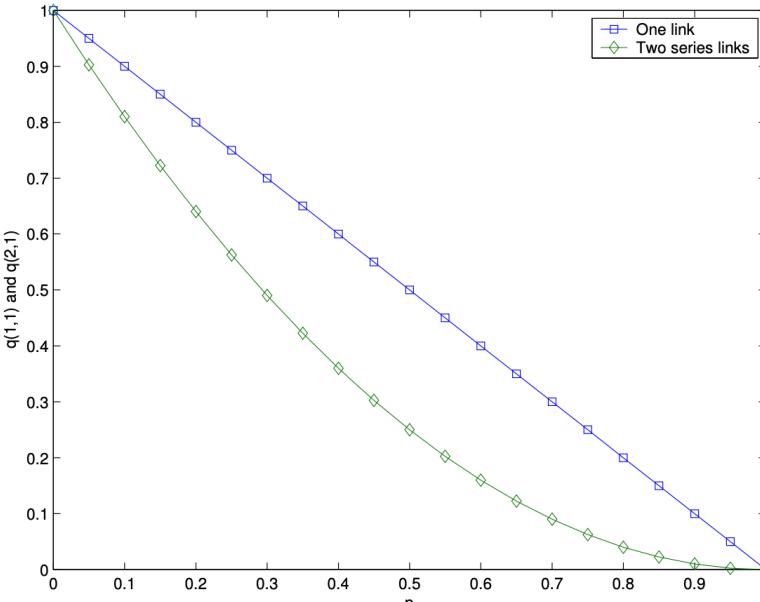


Quesito 5

Quale è la probabilità che un pacchetto venga trasmesso correttamente su entrambi i collegamenti nella rete?

- $1 - p^2$
- $(1 - p)^2$
- $(1 - p)p$
- $(1 - p)$

MOTIVO: La probabilità che un pacchetto venga trasmesso con successo attraverso entrambi i collegamenti nella rete è $(1 - p)^2$ perché una trasmissione riuscita con probabilità $(1-p)$ deve avere successo in due prove consecutive indipendenti.



Quesito 6

Quale è la probabilità che una mail composta da K pacchetti venga trasmessa correttamente su questa rete al primo tentativo?

- $(1 - p)^{K+2}$
- $(1 - p)^{K^2}$
- $(1 - p)^{2K}$
- $K \cdot (1 - p)^2$



Quesito 6

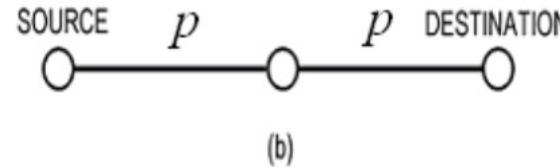
Quale è la probabilità che una mail composta da K pacchetti venga trasmessa correttamente su questa rete al primo tentativo?

- $(1 - p)^{K+2}$
- $(1 - p)^{K^2}$
- $(1 - p)^{2K} \xleftarrow{} (1 - p)^K \cdot (1 - p)^K$
- $K \cdot (1 - p)^2$

MOTIVO: tutti i K pacchetti devono essere trasmessi correttamente sui due collegamenti ossia un totale di $2K$ trasmissioni riuscite.



Series Two-Link Network



Sia Z la variabile casuale che rappresenta il numero di volte in cui un pacchetto viene trasmesso sul primo collegamento fino a quando non viene ricevuto correttamente.

Se la trasmissione di un pacchetto fallisce (con probabilità p), dobbiamo inviare nuovamente il pacchetto attraverso entrambi i collegamenti.

Sia $q_{2,k}$ la probabilità che il pacchetto venga trasmesso con successo esattamente in k tentativi.

Allora $q_{2,k}$ può essere espresso con la seguente ricorsione:

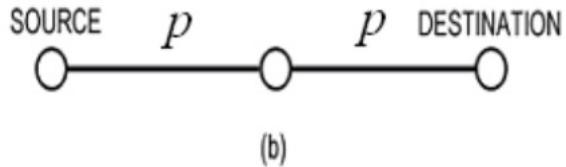
$$q_{2,1} = (1 - p)^2$$

$$q_{2,k} = (1 - \sum_{j=1}^{k-1} q_{2,j})(1 - p)^2$$

Notiamo che affinché la trasmissione abbia successo al k -esimo tentativo, tutti i tentativi precedenti devono essere falliti. La probabilità che il tentativo sia riuscito per qualche $j < k$ è $\sum_{j=1}^{k-1} q_{2,j}$ quindi la probabilità che esso fallisca $1 - \sum_{j=1}^{k-1} q_{2,j}$. Infine, il k -esimo tentativo deve riuscire, quindi abbiamo il termine $(1 - p)^2$ alla fine.



Series Two-Link Network



$$q_{2,k} = (1 - \sum_{j=1}^{k-1} q_{2,j})(1 - p)^2$$

$$q_{2,k} = (1 - p)^2 - (1 - p)^2 \sum_{j=1}^{k-1} q_{2,j}$$

$$q_{2,k} = (1 - p)^2 - (1 - p)^2 \sum_{j=1}^{k-2} q_{2,j} - (1 - p)^2 q_{2,k-1}$$

$$q_{2,k} = q_{2,k-1} - (1 - p)^2 q_{2,k-1}$$

$$q_{2,k} = q_{2,k-1}(1 - (1 - p)^2)$$

$$q_{2,k} = q_{2,k-1}(1 - (1 - p)^2)$$

$$q_{2,k} = q_{2,2}(1 - (1 - p)^2)^{k-1}$$

$$q_{2,k} = (1 - p)^2(1 - (1 - p)^2)^{k-1}$$



Quesito 7

Qual è il numero medio di trasmissioni sul primo collegamento, $E(Z)$, necessario per inviare un singolo pacchetto attraverso questa rete?

- $\frac{1}{(1 - p^2)}$
- $\frac{1}{(1 - p)^k}$
- $\frac{1}{(1 - p)^2}$
- $(1 - p)^k$



Quesito 7

Qual è il numero medio di trasmissioni sul primo collegamento, $E(Z)$, necessario per inviare un singolo pacchetto attraverso questa rete?

- $\frac{1}{(1-p^2)}$
- $\frac{1}{(1-p)^k}$
- $\frac{1}{(1-p)^2}$
- $(1-p)^k$

$$E(Z) = \sum_{k=1}^{\infty} k (1-p)^2 (1 - (1-p)^2)^{k-1}$$
$$E(Z) = \sum_{k=1}^{\infty} k \delta (1-\delta)^{k-1} = \delta \sum_{k=1}^{\infty} k (1-\delta)^{k-1} = \delta \frac{1}{\delta^2} = \frac{1}{\delta} = \frac{1}{(1-p)^2}$$

Vedi slide 7



Series Two-Link Network

Tuttavia, la nostra e-mail ha K pacchetti separati, ognuno dei quali richiede una media di $\frac{1}{(1-p)^2}$ trasmissioni (supponiamo ancora che un pacchetto non venga trasmesso fino a quando il pacchetto precedente non ha attraversato la rete con successo).



Quesito 8

Quale è il numero totale atteso di trasmissioni di pacchetti previste per l'e-mail? .

- $\frac{K}{(1 - p^2)}$
- $\frac{1}{(1 - p)^K}$
- $\frac{K}{(1 - p)^2}$
- $(1 - p)^K$



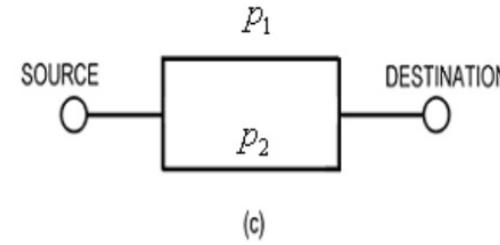
Quesito 8

Quale è il numero totale atteso di trasmissioni di pacchetti previste per l'e-mail? .

- $\frac{K}{(1 - p^2)}$
- $\frac{1}{(1 - p)^K}$
- $\frac{K}{(1 - p)^2}$
- $(1 - p)^K$



Parallel Two-Link Network



Supponiamo che la nostra rete sia costituita da due collegamenti che collegano una coppia di nodi comuni (una rete parallela).

La probabilità che un pacchetto venga danneggiato durante una trasmissione attraverso i due collegamenti è rispettivamente p_a e p_b .

Il pacchetto viaggia attraverso entrambi i collegamenti per arrivare a destinazione.



Quesito 9

Si consideri il caso in cui la trasmissione deve avere successo su entrambi i collegamenti contemporaneamente

Quale è la probabilità con cui il pacchetto raggiungerà la sua destinazione attraverso entrambi i collegamenti ?

- $(1 - p_a)p_b$
- $(1 - p_b)p_a$
- $(1 - p_a)(1 - p_b)$
- $p_a p_b$



Quesito 9

Quale è la probabilità con cui il pacchetto raggiungerà la sua destinazione attraverso entrambi i collegamenti (poiché la trasmissione deve avere successo su entrambi i collegamenti contemporaneamente)?

- $(1 - p_a)p_b$
- $(1 - p_b)p_a$
- $(1 - p_a)(1 - p_b)$
- $p_a p_b$



Quesito 10

Si consideri il caso in cui la trasmissione del pacchetto è considerata riuscita se il pacchetto viaggia su uno dei due collegamenti paralleli senza danneggiarsi.

Quale è la probabilità con cui il pacchetto raggiungerà la sua destinazione?

$(1 - p_a p_b)$

$(1 - p_b)p_a$

$(1 - p_a)(1 - p_b)$

$p_a p_b$



Quesito 10

Si consideri il caso in cui la trasmissione del pacchetto è considerata riuscita se il pacchetto viaggia su uno dei due collegamenti paralleli senza danneggiarsi.

Quale è la probabilità con cui il pacchetto raggiungerà la sua destinazione?

$(1 - p_a p_b)$

$(1 - p_b)p_a$

$(1 - p_a)(1 - p_b)$

$p_a p_b$

MOTIVO: Ciò si verifica con probabilità $(1 - p_a p_b)$ perché l'unica possibilità che la trasmissione non riesca è se si verifica un errore in entrambi i collegamenti (che si verifica con probabilità $p_a p_b$).



DNS AND HTTP DELAYS

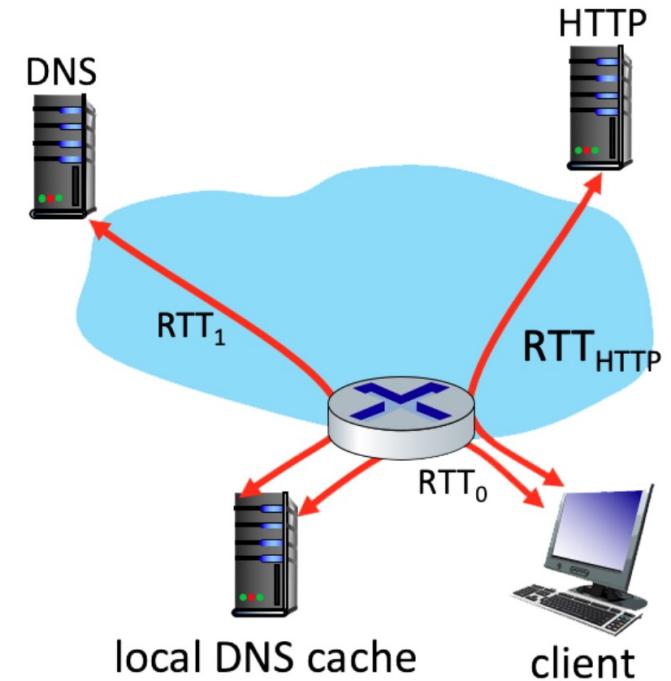
Si supponga di fare clic su un collegamento all'interno del browser Web per ottenere una pagina Web.

L'indirizzo IP per l'URL associato non è memorizzato nella cache dell'host locale, quindi è necessaria una ricerca DNS per ottenere l'indirizzo IP.

Supponiamo che due server DNS vengano visitati prima che il vostro host riceva l'indirizzo IP dal DNS.

Il primo server DNS visitato è la cache DNS locale, con un ritardo RTT di $RTT_0 = 2 \text{ ms}$. Il secondo server DNS contattato ha un RTT di 2 ms .

Supponiamo inizialmente che la pagina Web associata al collegamento contenga esattamente un oggetto, costituito da una piccola quantità di testo HTML. Si supponga che l'RTT tra l'host locale e il server Web contenente l'oggetto sia $RTT_{HTTP} = 95 \text{ ms}$.



Quesito 11

Assumendo un tempo di trasmissione zero per l'oggetto HTML, quanto tempo (in msec) trascorre da quando il client fa clic sul collegamento fino a quando il client riceve l'oggetto?

- 97 ms
- 26 ms
- 194 ms
- 99 ms



Quesito 11

Assumendo un tempo di trasmissione zero per l'oggetto HTML, quanto tempo (in msec) trascorre da quando il client fa clic sul collegamento fino a quando il client riceve l'oggetto?

97 ms

Il tempo da quando la richiesta Web viene effettuata nel browser fino alla visualizzazione della pagina nel browser è:
 $RTT_0 + RTT_1 + 2 \cdot RTT_{HTTP} = 2 + 2 + 2 \cdot 95 = 194 \text{ ms.}$

26 ms

Si noti che sono necessari 2 RTT_{HTTP} per recuperare l'oggetto HTML: un RTT_{HTTP} per stabilire la connessione TCP e quindi un RTT_{HTTP} per eseguire HTTP GET/risposta su quella connessione TCP.

194 ms

99 ms



Quesito 12

Supponiamo ora che l'oggetto HTML faccia riferimento a 5 oggetti molto piccoli sullo stesso server.

Trascurando i tempi di trasmissione, quanto tempo (in msec) trascorre da quando il client fa clic sul collegamento fino a quando l'oggetto di base e tutti i 5 oggetti aggiuntivi vengono ricevuti dal server Web sul client, supponendo HTTP non persistente e nessuna connessione TCP parallela?

- 95 ms
- 752 ms
- 1144 ms
- 212 ms



Quesito 12

Supponiamo ora che l'oggetto HTML faccia riferimento a 5 oggetti molto piccoli sullo stesso server.

Trascurando i tempi di trasmissione, quanto tempo (in msec) trascorre da quando il client fa clic sul collegamento fino a quando l'oggetto di base e tutti i 5 oggetti aggiuntivi vengono ricevuti dal server Web sul client, supponendo HTTP non persistente e nessuna connessione TCP parallela?

- 95 ms
Il tempo da quando la richiesta Web viene effettuata nel browser fino alla visualizzazione della pagina nel browser è:
- 752 ms
$$RTT_0 + RTT_1 + 2 \cdot RTT_{HTTP} + 2 \cdot 5 \cdot RTT_{HTTP} = 2 + 2 + 2 * 95 + 2 * 5 * 95 = 1144 \text{ ms.}$$
Si noti che sono necessari due ritardi RTT_{HTTP} per recuperare l'oggetto HTML di base: un RTT_{HTTP} per stabilire la connessione TCP e un RTT_{HTTP} per inviare la richiesta HTTP e ricevere la risposta HTTP. Quindi, in serie, per ciascuno dei 5 oggetti incorporati, è necessario un ritardo di $2 * RTT_{HTTP}$: un RTT_{HTTP} per stabilire la connessione TCP e quindi un RTT_{HTTP} per eseguire HTTP GET/risposta su quella connessione TCP.
- 1144 ms
- 212 ms



Quesito 13

Si supponga che l'oggetto HTML faccia riferimento a 5 oggetti molto piccoli sullo stesso server, ma si assuma che il client sia configurato per supportare un massimo di 5 connessioni TCP parallele, con HTTP non persistente.

- 380 ms
- 384 ms
- 190 ms
- 212 ms



Quesito 13

Si supponga che l'oggetto HTML faccia riferimento a 5 oggetti molto piccoli sullo stesso server, ma si assuma che il client sia configurato per supportare un massimo di 5 connessioni TCP parallele, con HTTP non persistente.

- 380 ms
- 384 ms
- 190 ms
- 212 ms

Poiché ci sono solo 5 oggetti, c'è un ritardo di 4 msec per la query DNS, due RTT_{HTTP} per la pagina di base e $2 \cdot RTT_{HTTP}$ per gli oggetti poiché le richieste per questi possono essere eseguite in parallelo.

$$RTT_0 + RTT_1 + 2 \cdot RTT_{HTTP} + 2 \cdot RTT_{HTTP}$$

Il totale è $2 + 2 + 190 + 190 = 384\text{ ms}$. Come sopra, sono necessari 2 RTT_{HTTP} per recuperare l'oggetto HTML di base: un RTT_{HTTP} per stabilire la connessione TCP e un RTT_{HTTP} per inviare la richiesta HTTP e ricevere la risposta HTTP contenente l'oggetto HTML di base.

Una volta ricevuto l'oggetto di base sul client, i 5 HTTP Gets per gli oggetti incorporati possono procedere in parallelo. Ciascuno (in parallelo) richiede due ritardi RTT_{HTTP} : uno RTT_{HTTP} per configurare la connessione TCP e un RTT_{HTTP} per eseguire HTTP GET/risposta per un oggetto incorporato.



Quesito 14

Si supponga che l'oggetto HTML faccia riferimento a 5 oggetti molto piccoli sullo stesso server, ma si assuma che il client sia configurato per supportare un massimo di 5 connessioni TCP parallele, con HTTP persistente.

- 280 ms
- 289 ms
- 412 ms
- 212 ms



Quesito 14

Si supponga che l'oggetto HTML faccia riferimento a 5 oggetti molto piccoli sullo stesso server, ma si assuma che il client sia configurato per supportare un massimo di 5 connessioni TCP parallele, con HTTP persistente.

- 280 ms
- 289 ms
- 412 ms
- 212 ms

Poiché ci sono solo 5 oggetti, c'è un ritardo di 4 msec per la query DNS, due RTT_{HTTP} per la pagina di base e 1 RTT_{HTTP} per gli oggetti.

$$RTT_0 + RTT_1 + 2 \cdot RTT_{HTTP} + RTT_{HTTP}$$

Il totale è

$$2 + 2 + 190 + 95 = 289 \text{ ms}$$

Come sopra, sono necessari due ritardi RTT_{HTTP} per recuperare l'oggetto HTML di base: un RTT_{HTTP} per stabilire la connessione TCP e un RTT_{HTTP} per inviare la richiesta HTTP e ricevere la risposta HTTP contenente l'oggetto HTML di base. Tuttavia, con HTTP persistente, questa connessione TCP rimarrà aperta per future richieste HTTP, che quindi non subiranno un ritardo nella creazione del TCP. Una volta ricevuto l'oggetto base presso il client, è possibile procedere in parallelo al massimo di cinque richieste, ognuna delle quali recupera uno dei 5 oggetti incorporati. Ciascuno (in parallelo) richiede solo un ritardo RTT_{HTTP} per eseguire HTTP GET/risposta per un oggetto incorporato. Una volta recuperati questi primi cinque oggetti, (se necessario) è possibile recuperare (in parallelo) gli oggetti incorporati rimanenti. Questo secondo round di HTTP GET/risposta per recuperare gli oggetti incorporati rimanenti richiede solo un altro RTT_{HTTP} , poiché la connessione TCP è rimasta aperta.



Quesito 15

Qual è il metodo più veloce che abbiamo esplorato: seriale non persistente, parallelo non persistente o parallelo persistente?

- connessioni parallele non persistenti
- connessioni parallele persistenti
- connessioni seriali non persistenti



Quesito 15

Qual è il metodo più veloce che abbiamo esplorato: seriale non persistente, parallelo non persistente o parallelo persistente?

- connessioni parallele non persistenti
- connessioni parallele persistenti
- connessioni seriali non persistenti

Il ritardo quando si utilizzano connessioni parallele persistenti è inferiore rispetto all'utilizzo di connessioni parallele non persistenti, che è comunque più veloce rispetto all'utilizzo di connessioni seriali non persistenti.



Esercizi – Livello di Applicazione e Livello di Trasporto



Esercizi - Livello di Trasporto



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Esercizi - Livello di Trasporto

R1: Supponete che il livello di rete fornisca il seguente servizio. Il livello di rete nell'host sorgente accetta un segmento di dimensione massima di 1200 byte e un indirizzo dell'host di destinazione dal livello di trasporto. Il livello di rete poi garantisce di consegnare il segmento al livello di trasporto nell'host di destinazione. Supponete che molti processi applicativi di rete possano essere in esecuzione sull'host di destinazione.

- a. Progettate un protocollo a livello di trasporto più semplice possibile che porti i dati applicativi al processo desiderato dell'host di destinazione. Assumete che il sistema operativo nell'host di destinazione abbia assegnato un numero di porta di 4 byte a ciascun processo applicativo in esecuzione.
- b. Modificate questo protocollo in modo che fornisca un «indirizzo di ritorno» al processo di destinazione.
- c. Nei vostri protocolli il livello di trasporto deve fare qualcosa nel nucleo della rete di calcolatori?



Esercizi - Livello di Trasporto

R8: Supponete che un web server sia in esecuzione sull'host C sulla porta 80.

Supponete che questo web server usi le connessioni persistenti e stia al momento ricevendo richieste da due diversi host, A e B.

Tutte le richieste vengono inviate attraverso la stessa socket sull'host C?

Se vengono fatte passare attraverso socket diverse, entrambe hanno la porta 80?
Discutete e spiegate questa situazione.



Soluzione Esercizio R8

Per ogni connessione persistente, il server Web crea un "socket di connessione" separato.

Ogni socket di connessione è identificato da una quadrupla: (indirizzo IP di origine, numero di porta di origine, indirizzo IP di destinazione, numero di porta di destinazione).

Quando l'host C riceve un datagramma IP, esamina questi quattro campi nel datagramma/segmento per determinare a quale socket deve passare il payload del segmento TCP.

Pertanto, le richieste di A e B passano attraverso socket diversi.

L'identificatore per entrambi questi socket ha 80 per la porta di destinazione; tuttavia, gli identificatori per questi socket hanno valori diversi per gli indirizzi IP di origine.

A differenza di UDP, quando il livello di trasporto passa il carico utile di un segmento TCP al processo dell'applicazione, non specifica l'indirizzo IP di origine, poiché questo è implicitamente specificato dall'identificatore del socket.



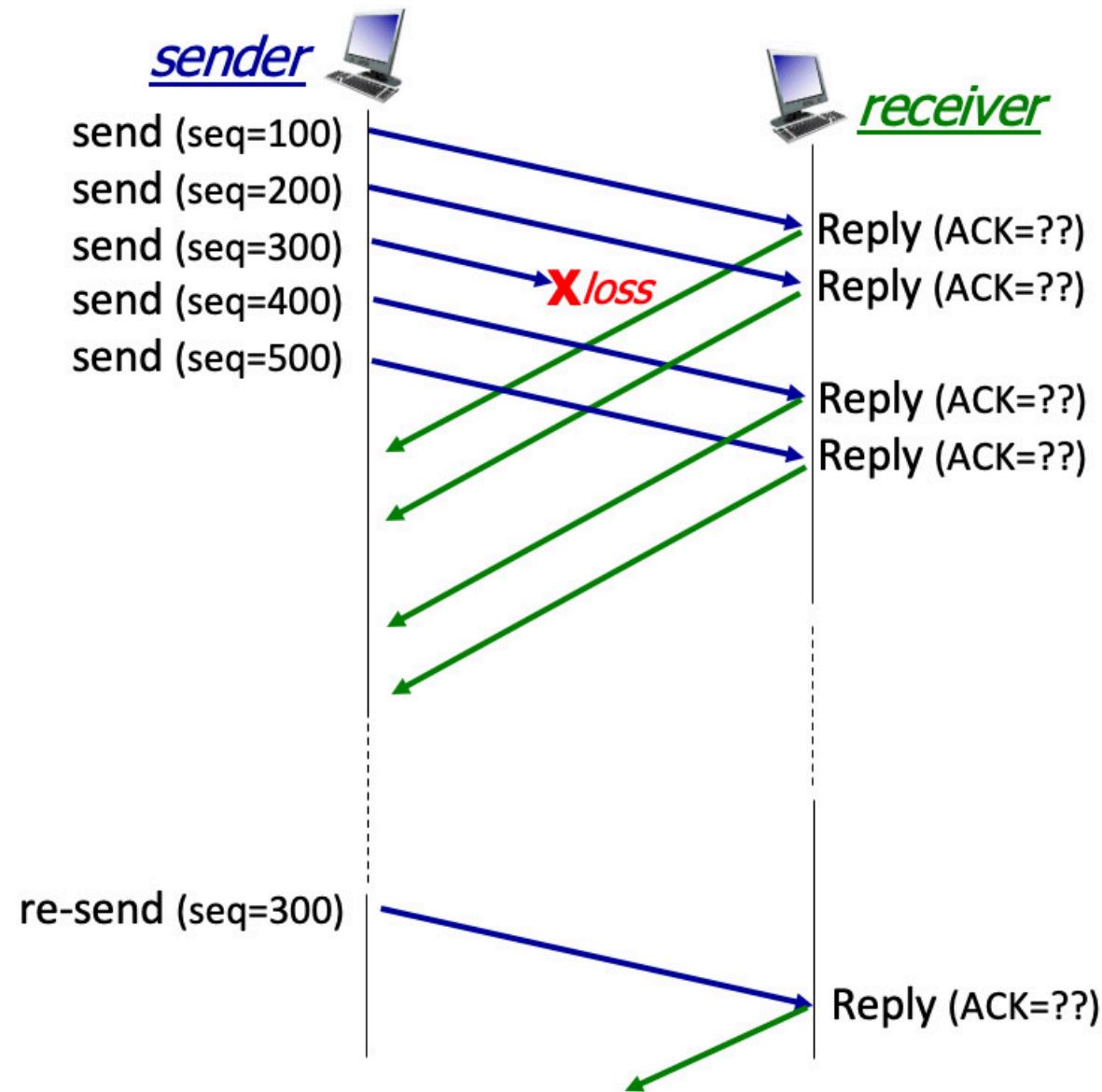
Esercizi - Livello di Trasporto

R9: Nei nostri protocolli *rdt*, perché abbiamo bisogno di introdurre i numeri di sequenza?



Soluzione Esercizio R9

I numeri di sequenza sono necessari a un destinatario per scoprire se un pacchetto in arrivo contiene nuovi dati o è una ritrasmissione.



Esercizi - Livello di Trasporto

P3: UDP e TCP utilizzano il complemento a 1 per calcolare il checksum. Supponiamo di avere i seguenti tre byte: 01010011, 01100110 e 01110100.

Qual è il complemento a 1 della loro somma?

Notiamo che, sebbene UDP e TCP usino checksum da 16 bit, in questo problema consideriamo addendi a 8 bit.

Con lo schema del complemento a 1, come vengono rilevati gli errori dal destinatario?

E' possibile che un errore su 1 bit non venga rilevato?

E che cosa avviene per gli errori su 2 bit?



Soluzione Esercizio P3

Avvolgere in caso di overflow

$$\begin{array}{r} 01010011 \\ + 01100110 \\ \hline 10111001 \end{array}$$

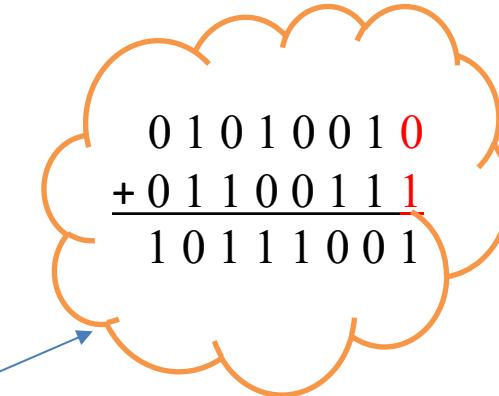
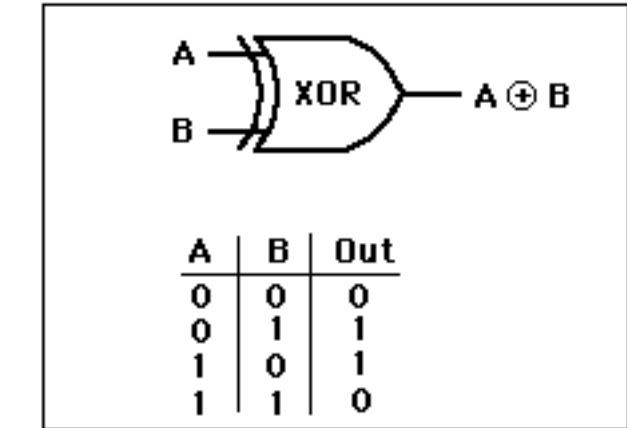
$$\begin{array}{r} 10111001 \\ + 01110100 \\ \hline 00101101 \end{array}$$

Riporto di 1

$$\begin{array}{r} + 00101101 \\ \hline 00101110 \end{array}$$

Il Complemento a uno = **11010001**

Per rilevare gli errori, il ricevitore aggiunge le quattro parole (le tre parole originali e il checksum). Se la somma contiene uno zero, il ricevente sa che c'è stato un errore. Verranno rilevati tutti gli errori a un bit, ma gli errori a due bit possono non essere rilevati (ad esempio, se l'ultima cifra della prima parola viene convertita in uno 0 e l'ultima cifra della seconda parola viene convertita in un 1).



Esercizi - Livello di Trasporto

P37: Paragonate Go-Back-N, Selective Repeat, e TCP (senza il delayed ACK). Assumete che i valori di timeout per tutti questi tre protocolli siano sufficientemente grandi per cui 5 segmenti di dati consecutivi e i corrispondenti ACK possano essere ricevuti (se non persi nel canale) dall'host ricevente (Host B) e dall'host mittente (Host A) rispettivamente. Supponete che l'Host A invii cinque segmenti di dati all'HOST B e che il secondo segmento (inviato da A) venga perso. Alla fine, tutti e 5 i segmenti di dati sono stati ricevuti correttamente dall'Host B.

- Quanti segmenti ha inviato l'Host A in tutto e quanti ACK ha mandato l'Host B in tutto? Quali sono i loro numeri di sequenza? Rispondete a queste domande per tutti e tre i protocolli.
- Se i valori di timeout per tutti e tre i protocolli sono molto più grandi di 5 RTT, quale protocollo consegnerà con successo tutti e cinque i segmenti di dati nell'intervallo di tempo più breve?



Soluzione Esercizio P37.a

- a. Quanti segmenti ha inviato l'Host A in tutto e quanti ACK ha mandato l'Host B in tutto? Quali sono i loro numeri di sequenza? Rispondete a queste domande per tutti e tre i protocolli.

Go-Back-N:

A invia 9 segmenti in totale. Vengono inizialmente inviati i segmenti 1, 2, 3, 4, 5 e successivamente i segmenti 2, 3, 4 e 5.

B invia 8 ACK. Sono 4 ACKS con numero di sequenza 1 ,e 4 ACKS con numeri di sequenza 2, 3, 4 e 5.

Selective Repeat:

A invia 6 segmenti in totale. Inizialmente vengono inviati i segmenti 1, 2, 3, 4, 5 e successivamente il segmento 2.
B invia 5 ACK. Sono 4 ACK con numero di sequenza 1, 3, 4, 5. E c'è un ACK con numero di sequenza 2.

TCP:

A invia 6 segmenti in totale. Inizialmente vengono inviati i segmenti 1, 2, 3, 4, 5 e successivamente il segmento 2.
B invia 5 ACK. Sono 4 ACK con numero di sequenza 2. C'è un ACK con numero di sequenza 6. Si noti che TCP invia sempre un ACK con numero di sequenza previsto.

