Xarxes: Pràctica 1, Programació d'aplicacions de xarxa
Versió v250303-01

César Fernàndez, Paula Galucci, Enric Guitart, Carles Mateu

Març 2025

Finalitat de la pràctica:

- Aprendre a programar aplicacions de xarxes
- Entendre el model client-servidor
- Aprendre a programar i dissenyar protocols de comunicacions

Transmissió de vídeo amb RTSP i RTP

En aquesta pràctica heu d'implementar un servidor i un client de vídeo en temps real que es comuniquen mitjançant el Real-Time Streaming Protocol (RTSP) i empren Real-time Streaming Protocol (RTP).

Heu d'implementar tant el client com el servidor. Disposareu per fer-ho, a banda d'aquesta documentació:

- d'un esquelet en Python de com heu de començar a implementar el servidor
- un altre esquelet en Python de com heu de fer el client. En aquest cas a inclourà la major part de la interfície gràfica ja programada.
- Un binari (executable) d'un servidor ja implementat.
- Un binari (executable) d'un client ja implementat.
- Alguns fitxers de vídeo d'exemple.

RSTP i RTP

RTSP i RTP son dos protocols de comunicacions emprats habitualment per streaming de vídeo. Un en gestiona el "control" (RTSP) mentre que l'altre fa la transmissió efectiva del contingut (RTP). Són habituals, per exemple, per accedir a càmeres de vídeo per xarxa, etc.

RTSP

RTSP (Real-Time Streaming Protocol) és un protocol de comunicacions que empra tant TCP com UDP, encara que per aquesta pràctica només haureu de fer servir TCP. És un protocol amb estat, per tant, jo us donaré un diagrama d'estats del protocol (és un protocol molt simple, i només en fareu un subset), i la millor forma d'implementar-lo, si és possible, és com una màquina d'estats. I la seva operativa és mitjançant l'enviament d'ordres (en text) que provoquen les transicions o canvis d'estat.

El diagrama d'estats és:

En el nostre cas només caldrà implementar-ne 3 dels possibles estats del protocol:

Informació

RTSP està definit a Schulzrinne, Rao et al. 2016, que actualitza la versió antiga del protocol a Rao, Lanphier i Schulzrinne 1998. Implementem una versió molt simple de RTSP i amb el RFC més antic en tindríeu prou.

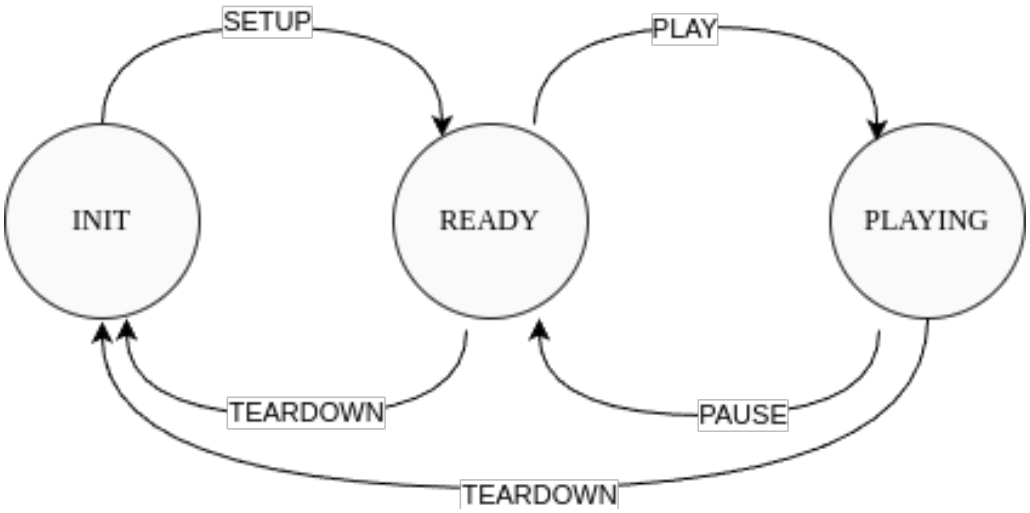


Figura 1: Diagrama d'estats simplificat de RTSP

- INIT
- READY
- PLAYING

Per simplificar, farem servir els mateixos estats amb el mateix nom a client i servidor.

Les ordres que haureu d'implementar són les següents:

- SETUP
- PLAY
- PAUSE
- TEARDOWN

Aquestes ordres s'envien des del client cap al servidor, el servidor no envia ordres al client, sinó que respon amb missatges d'estat.

Ordres

Implementar les ordres al client va lligat a la interfície gràfica, en teniu l'esquelet. Cadascun dels botons que teniu a la part superior correspon a una de les possibles ordres de RTSP que heu d'implementar (RTSP té moltes més ordres en realitat).

SETUP (C->S) Prepara la connexió. Passant alguns paràmetres al servidor, com el protocol de transport que fareu anar, el port, etc.

- Enviar una ordre SETUP al servidor. L'ordre SETUP indica al servidor, també, el fitxer (stream) que volem rebre. En el nostre cas serà un nom de fitxer de vídeo, però l'estandard permet tota mena d'URLs. Dins de l'ordre heu de passar el número de seqüència del missatge, aquest serà inicialment 1, i l'incrementarem en cada missatge que enviem del client al servidor. També heu de posar-hi, a la part de transport el protocol i el port. Pel nostre cas, sempre serà: "Transport: RTP/UDP; client_port= *numport*". On *numport* serà el nombre de port UDP a emprar per aquell client.

- Rebre la resposta del servidor:
  - Si la resposta és positiva:
    * Crear un socket per datagrames RTP.
    * [RECOMANACIÓ] Fer un thread per rebre els datagrames, descodificar-los, i refrescar la pantalla.
  - Si es negativa:
    * No fer res (mostrar l'error)

PLAY (C->S) Reprodueix el flux de vídeo.

- Enviar el missatge PLAY al servidor. Dins de l'ordre s'ha d'emprar l'identificador de sessió que ens ha donat el servidor.
- Rebre la resposta del servidor:
  - Si la resposta és positiva
    * Iniciar un Thread per reproduir el vídeo.
  - Si és negativa:
    * No fer res (mostrar l'error)

PAUSE (C->S) Atura temporalment el flux de vídeo.

- Enviar el missatge PAUSE al servidor. Dins de l'ordre s'ha d'emprar l'identificador de sessió que ens ha donat el servidor.
- Rebre la resposta del servidor:
  - Si la resposta és positiva
    * Aturar el Thread de reproduir el vídeo.
  - Si és negativa:
    * No fer res (mostrar l'error)

TEARDOWN (C->S) Destrueix la sessió.

- Enviar el missatge TEARDOW al servidor. Dins de l'ordre s'ha d'emprar l'identificador de sessió que ens ha donat el servidor.
- Rebre la resposta del servidor:
  - Si la resposta és positiva
    * Aturar el Thread de reproduir el vídeo.
    * Tancar el socket UDP.
    * Passar a estat inicial.
  - Si és negativa:
    * No fer res (mostrar l'error)

Les respostes del servidor són de la forma:

```
RTSP/1.0 <status> <TEXT>
CSeq: <num_sequencia>
Session: <id_sessio>
```

On, **status** pot ser:

- 200: OK
- 400: Bad Request
- 404: File Not Found
- 500: Internal Server Error
- 501: Not Implemented

El número de seqüència, **CSeq**, serà el mateix que l'ordre que ha generat aquesta resposta. I **id_sessio** serà el de la sessió en curs. En el missatge de resposta a **SETUP**, on no hi ha nombre de seqüència, el servidor inicialitza aquesta, **id_sessio** serà una cadena de text de mínim 10 caràctes de longitud, aleatoria. Podeu emprar un nombre aleatori de 10 digits (amb zeros a l'esquerra) per fer-vos fàcil el seguiment.

Exemple de diàleg

Un exemple de comunicació entre client i servidor:

Client envia:

```
SETUP rick.webm RTSP/1.0
CSeq: 1
Transport: RTP/UDP; client_port= 25000
```

Això vol dir, volem rebre el fitxer rick.webm, el número de seqüència és 1, i el rebrem al port 25000/UDP. Si tots està correcte la resposta serà:

```
RTSP/1.0 200 OK
CSeq: 1
Session: XARXES_00005017
```

Un cop l'usuari al client prem el botó de **PLAY**, envia:

```
PLAY rick.webm RTSP/1.0
CSeq: 2
Session: XARXES_00005017
```

Fixeu-vos que el nom de fitxer i l'identificador de sessió és el mateix. El nombre de seqüència s'incrementa en 1.

La resposta del servidor, si tot va bé:

```
RTSP/1.0 200 OK
CSeq: 2
Session: XARXES_00005017
```

El nombre de seqüència correspon al del missatge de **PLAY**, i la sessió és manté tota l'estona.

Si l'usuari al client prem el botó de **PAUSE**, envia:

```
PAUSE rick.webm RTSP/1.0
CSeq: 3
Session: XARXES_00005017
```

I la resposta del servidor, si tot va bé:

```
RTSP/1.0 200 OK
CSeq: 3
Session: XARXES_00005017
```

Finalment, si l'usuari al client prem el botó de **TEARDOWN**, envia:

```
TEARDOWN rick.webm RTSP/1.0
CSeq: 4
Session: XARXES_00005017
```

I la resposta del servidor, si tot va bé:

```
RTSP/1.0 200 OK
CSeq: 4
Session: XARXES_00005017
```

I la sessió es tanca.

NOTA: No cal implementar més ordres que aquestes. No cal implementar més estats que els que us he indicat.

NOTA 2: Els noms de fitxers que us he donat són els que he emprat per fer les proves i corresponen a fitxers que us proporciono.

RTP

RTP (Real-time Transport Protocol) és un protocol de nivell 4 que s'encarrega de la transmissió de dades en temps real, com ara l'àudio, la veu i el vídeo. Empra UDP i és un protocol molt simple, que consta d'una capçalera i una càrrega útil. La capçalera conté informació com el temps, el tipus de dades, etc. La càrrega útil conté les dades a transmetre.

Els paquets que rebeu tindran el format següent:

```
 0               1               2               3
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          RTP packet                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                         RTP header                            |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                         RTP payload                           |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

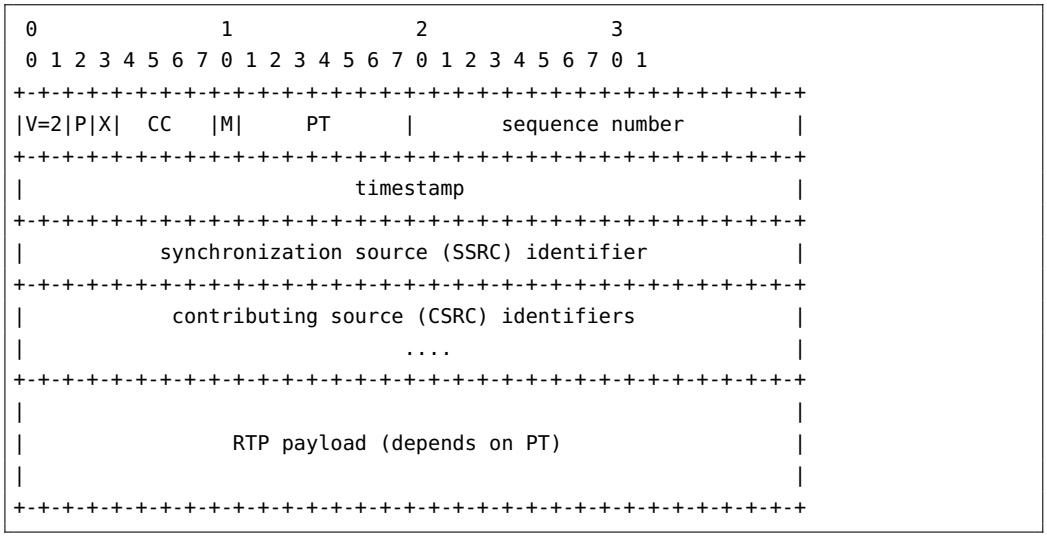
Informació

RTP està definit a Frederick et al. 1996, ja obsolet, actualitzat a Schulzrinne, Casner et al. 2003. Per la part que implementem, amb la versió antiga en teniu prou per llegir, si teniu curiositat.

On la part de **RTP header** és la capçalera de RTP i la part de **RTP payload** és la càrrega útil.

RTP Header

El format de la capcelera de RTP (**RTP header**) és el següent:



On:

- **V** és la versió del protocol. En el nostre cas sempre serà 2.
- **P** és un bit que indica si hi ha padding.
- **X** és un bit que indica si hi ha extensions.
- **CC** és el número de CSRCs.
- **M** és un bit que indica si és l'últim paquet del missatge.
- **PT** és el tipus de dades. En el nostre cas, el **PT** serà sempre 26, que indica que el tipus de dades a transmitir és el vídeo MJPEG. Transmetrem trames en format JPEG.
- **sequence number** és el número de seqüència.
- **timestamp** és el timestamp.
- **SSRC** és l'identificador de la font de sincronització. Zeros en el nostre cas.
- **CSRC** és l'identificador de la font de contribució. Zeros o un nombre aleatori en el nostre cas.

El **sequence number** és un nombre de 16 bits que incrementa en cada paquet enviat. Això permet identificar els paquets i poder-los ordenar.

El **timestamp** és un nombre de 32 bits que indica el temps en que s'ha capturat la imatge. Aquest temps és en mostres, i permet que el receptor pugui reproduir les imatges a la velocitat correcta.

Tant **SSRC** com **CSRC** s'utilitzen en aquells casos que tinguem múltiples fonts de dades en streams que les tinguin. En el nostres cas, per simplificar el protocol les deixarem com a 0.

INFORMACIÓ

Per la vostra pràctica, no és obligatori emprar el **sequence number** ni el **timestamp**, i els exemples que teniu (client i servidor) no els tenen en compte.

RTP Payload

La càrrega útil de RTP serà un número de frame i una imatge en format JPEG.

UDP a Mac OS

Al Mac OS, per disseny i decisió d'Apple, només es poden enviar datagrames UDP de 9216 bytes de mida (és la mida màxima del búffer pels sockets UDP). Tenim diverses solucions, per una banda, podem emprar (no persisteix d'arrancada en arrancada de l'ordinador):

```
sudo sysctl -w net.inet.udp.maxdgram=65535
```

a una terminal. Que ampliarà aquesta mida. O per l'altra podem emprar els paràmetres SO_SNDBUF i SO_RCVBUF al crear el socket, per ampliar aquest búffer.

```
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_RCVBUF, <i>)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_SNDBUF, <i>)
```

Encara que teniu en compte que Mac OS (i altres sistemes operatius) fan auto tune dels sockets UDP canviant-ne les mides dels búffers, per tant, el mecanisme més segur és el primer, el sysctl.

Implementació de RTP

Per simplificar el vostre treball, he implementat a l'esquelet de codi una part de com llegir els paquets RTP i com descodificar-los. Aquesta part de la pràctica ja està feta, i no cal que la modifiqueu **encara que us caldrà ampliar-la.**

Per fer la part de RTP heu de fer el següent:

1. Construir al servidor els paquets RTP a partir de les imatges del vídeo. Teniu codi que us ajudarà.
2. Implementar al servidor un socket UDP per enviar els paquets RTP al client.
3. Implementar un socket UDP al client per rebre els paquets RTP.
4. Llegir els paquets RTP i passar-los a la part de descodificació.
5. Refrescar la pantalla amb les imatges rebudes. Teniu codi que us ajudarà.

El vostre treball serà implementar tots aquests passos per tenir una operativa completa, d'enviament i recepció d'UDP.

Per accedir i treballar amb vídeo, farem servir la llibreria **OpenCV2** de Python.

Per descomprimir les imatges, podeu fer servir la llibreria **Pillow** de Python.

Per refrescar la pantalla, heu de fer servir la llibreria **Tkinter** de Python.

Fitxers de vídeo

Us proporciono un conjunt de fitxers de vídeo que podeu fer servir per fer les proves. Teniu més fitxers de domini públic que podeu trobar a la pàgina de l'Internet Archive.

Per fer les proves, podeu fer servir el fitxer `rick.webm` que us proporciono, així com els altres que penjarem al campus. Podeu fer servir altres fitxers, descarregats de la xarxa, però pot ser que tinguin encodings que facin que els datagrames UDP siguin massa grans per enviar (ho notareu pels errors a la traça del servidor en pantalla).

El format dels fitxers que us proporciono és WebM, que és un format de vídeo obert i lliure, que es basa en el codec VP8/VP9, encara que el codi pot enviar altres formats.

Requisits de funcionament

- El client ha de ser capaç de connectar-se al servidor, i enviar les ordres de RTSP. En rebre les respostes, ha de ser capaç de processar-les i fer el que correspongui.
- El servidor, per la seva banda, ha de ser capaç de rebre les ordres del client, processar-les, i enviar les respostes correctes.
- Un cop el client i el servidor han establert la connexió, el client ha de ser capaç de rebre els datagrames RTP, processar-los, i mostrar-los a la pantalla.
- El client ha de ser capaç de reproduir i aturar el vídeo.
- El servidor ha de ser capaç de servir més d'un client al mateix temps.
- El servidor ha de ser capaç de servir més d'un fitxer de vídeo al mateix temps a diferents clients.
- El servidor ha de ser capaç de servir més d'un vídeo del mateix fitxer al mateix temps.
- El servidor ha de ser capaç de servir més d'un vídeo de diferents fitxers al mateix temps.
- El servidor ha de rebre el número de port a atendre per línea d'ordres.

Implementació de prova

Us proporcionem una implementació de prova que podeu emprar per validar els vostres clients i servidors mentre els aneu desenvolupant. Aquesta implementació està en binari, disponible per:

- Fedora 41
- Rocky Linux 9 (CentOS 9, RedHat 9)
- Mac OS/X

Possiblement durant el semestre pujarem alguna versió addicional per algun altre sistema operatiu.

Nota per Mac OS

En macOS es factible que no us deixi executar el binari. Pot ser pels mecanismes de seguretat del vostre Mac. Podeu treure l'atribut de quarentena al binari, per permetre executar-lo. En una terminal feu:

```
$ sudo xattr -dr com.apple.quarantine ./practicaxarxes2025.macosx
```

Això treu l'atribut com.apple.quarantine de practicaxarxes2025.macosx i us el permet executar.

Funcionament

El binari de proves que teniu es diu:

practicaxarxes2025.<sistema>

Per exemple: practicaxarxes2025.macosx per Mac OS/X.

Un cop tingueu el binari del vostre sistema operatiu, si executeu el binari us sortirà l'ajuda:

```
(base) [~/tmp/xarxes]$ ./practicaxarxes2025.macosx
Usage: practicaxarxes2025.macosx [OPTIONS] COMMAND [ARGS]...

  Practica de Xarxes VideoStreaming 2025

Options:
  --version                       Show the version and exit.
  --debug / --no-debug            [default: no-debug]
  --debug-level [TRACE|DEBUG|INFO|WARNING|ERROR]
                                  [default: INFO]
  --debug-file PATH               [default: xarxes.log]
  --help                          Show this message and exit.

Commands:
  client  Client for video streaming with RTSP over TCP and RTP over UDP.
  server  Server xarxes 2025 video streaming
```

El programa té dues ordres, **client** i **server** (Client i Servidor). Cadascuna d'elles té una sèrie de paràmetres addicionals, a banda dels paràmetres generals del programa, els que us surten amb l'ajuda del programa principal o si no passeu cap ordre al programa.

Servidor

Per executar els servidor executem el programa amb el paràmetre server.

```
(base) [~/tmp/xarxes]$ ./practicaxarxes2025.macosx server --help
Usage: practicaxarxes2025.macosx server [OPTIONS]

  Server xarxes 2025 video streaming

Options:
```

```
  -p, --port INTEGER    RTSP port (TCP)  [default: 4321]
  -h, --host TEXT       IP Address for RTSP (TCP)  [default: localhost]
  --max-frames INTEGER  Maximum number of frames to stream
  --frame-rate INTEGER  Frame rate to stream (FPS)  [default: 25]
  --loss-rate INTEGER   Loss rate of UDP/RTP stream (0-100)  [default: 0]
  --error INTEGER       Comportar-se com si hi hagues un error  [default: 0]
  --help                Show this message and exit.
```

Tenim aleshores diverses opcions:

-p, –port INTEGER El port TCP on esperarà el servidor les connexions dels clients. Per defecte el port 4321/TCP.

-h, –host L'adreça IP de les disponibles al servidor on esperarem les connexions, per defecte 127.0.0.1 (localhost)

–max-frames INTEGER El nombre màxim d'imatges (*frames*) del vídeo a transmetre. En el moment que se'n transmetin max-frames s'atura la transmissió.

–frame-rate INTEGER Taxa d'imatges per segon (*frames per second*, FPS), a transmetre. Habitualment 25 imatges per segon és la taxa que s'empra vídeo.

–loss-rate INTEGER Taxa de pèrdues de la transmissió UDP. Permet eliminar, aleatòriament, un cert nombre de trames UDP de la transmissió.

–error INTEGER Permet provocar errors en l'operativa del servidor. Els possibles valors són:

- 1: Respondre amb 400 a totes les peticions de SETUP.
- 2: Respondre amb 500 a totes les peticions de SETUP.
- 3: Respondre amb 400 a totes les peticions de PLAY o TEARDOWN si estem a estat READY.
- 4: Respondre amb 500 a totes les peticions de PLAY o TEARDOWN si estem a estat READY.
- 5: Respondre amb 400 a totes les peticions de PAUSE o TEARDOWN si estem a estat PLAYING.
- 6: Respondre amb 500 a totes les peticions de PAUSE o TEARDOWN si estem a estat PLAYING.
- 100: Respondre amb un id de sessió erròni (ERROR<numaleatori>).
- 101: Respondre amb un número de seqüència erroni, concretament Cseq -1.
- 102: Respondre amb un número de seqüència erroni, enviant una cadena de text ("ERROR") en comptes d'un número.

Client

```
(base) [~/tmp/xarxes]$ ./practicaxarxes2025.macosx client --help
Usage: practicaxarxes2025.macosx client [OPTIONS] [VIDEOFILE]

  Client for video streaming with RTSP over TCP and RTP over UDP.

  Connects to the server and requests VIDEOFILE for streaming.

  If no VIDEOFILE is given, the client will use "rick.webm" as default.

Options:
  -p, --port INTEGER      RTSP port (TCP) Server
```

```
  -h, --host TEXT         IP Address for RTSP server to connect (TCP)
  -u, --udp-port INTEGER  RTP port (UDP) Client
  --help                  Show this message and exit.
```

Tenim aleshores diverses opcions i un paràmetre:

<*videofile*> El fitxer que demanarem al servidor, per defecte *rick.webm*.
-p, –port INTEGER El port TCP on connectar al servidor.
-h, –host TEXT L'adreça IP del servidor.
-u, –udp-port INTEGER El port udp que emprarem, per defecte 25000/UDP

L'aspecte del client és el següent:

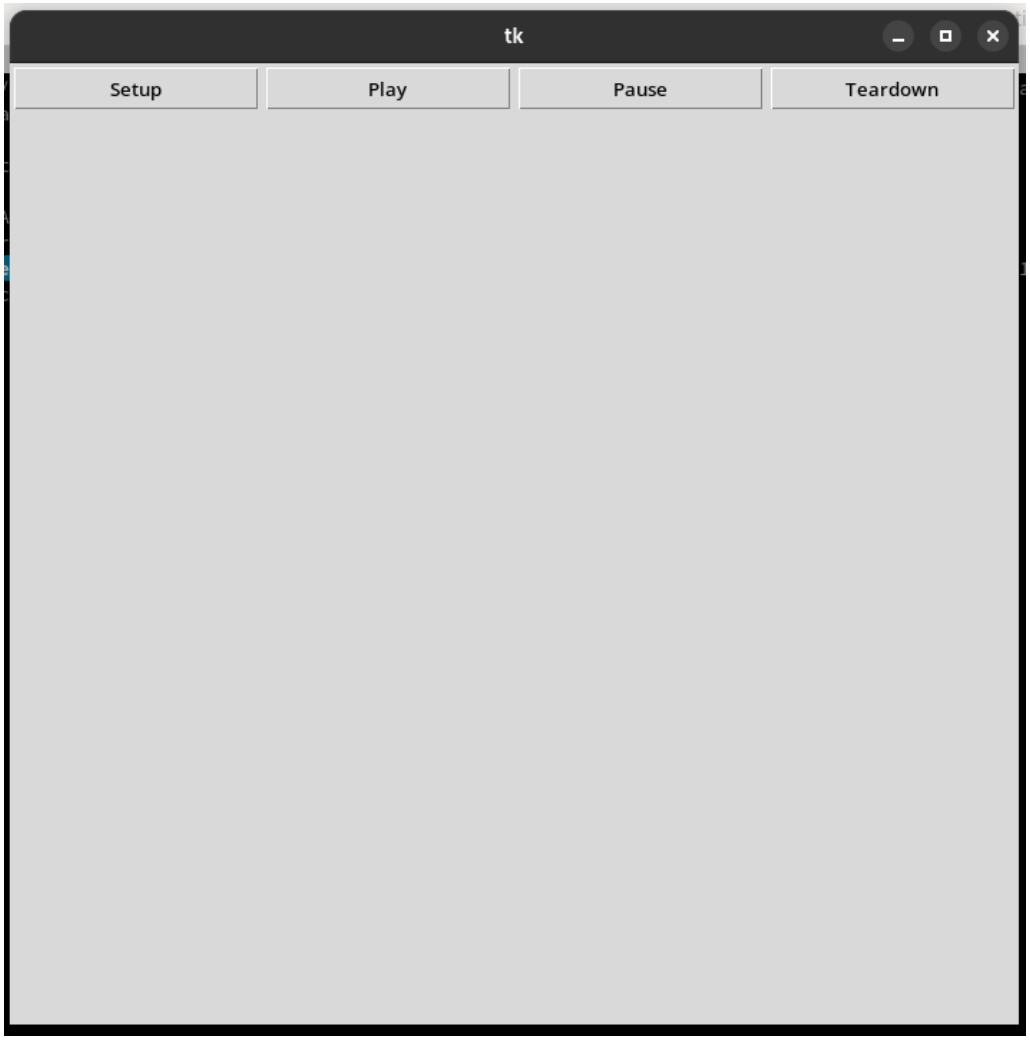


Figura 2: Aspecte inicial del client

Llibreries permeses

Durant el desenvolupament podeu fer servir algunes llibreries i eines que us poden facilitar la vida. Algunes les teniu ja incloses al requirements.txt i/o al pyproject.toml.

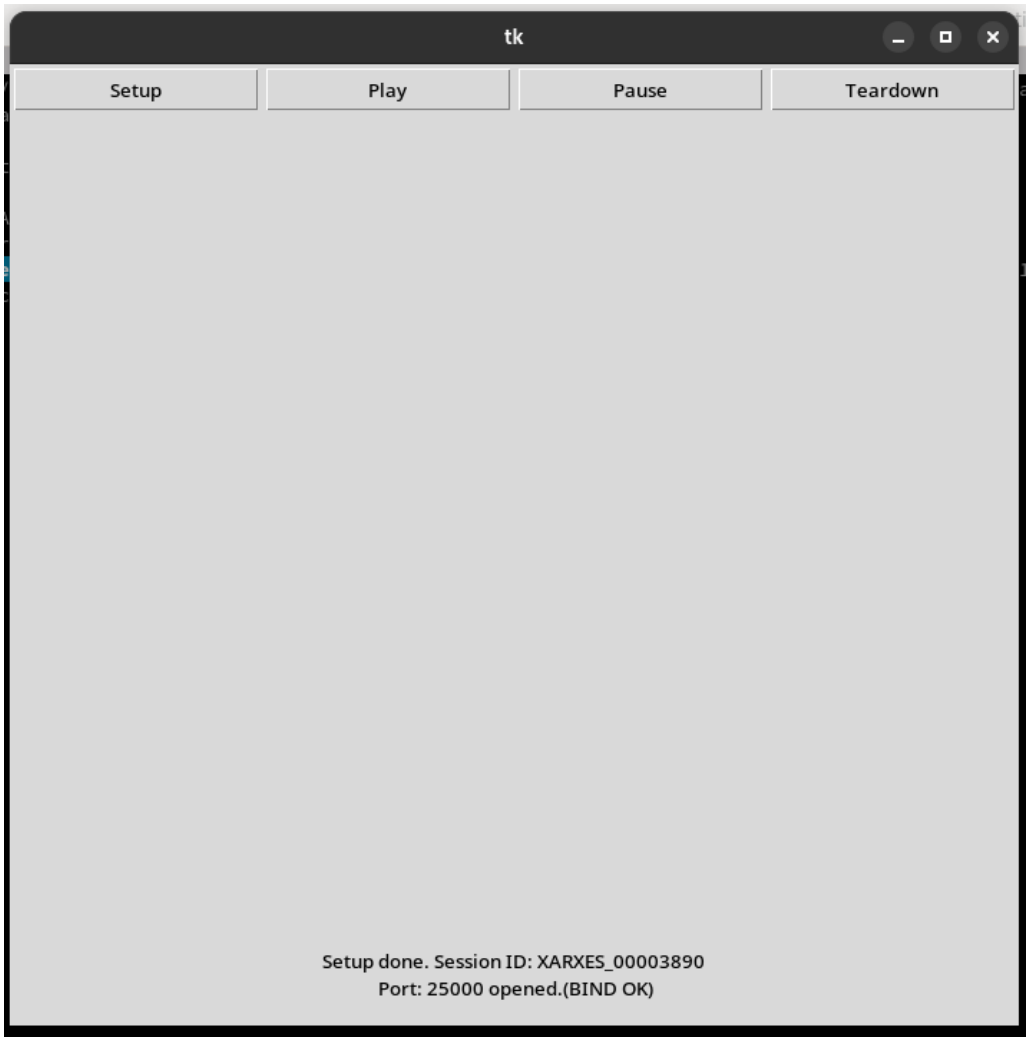


Figura 3: Un cop fet el setup

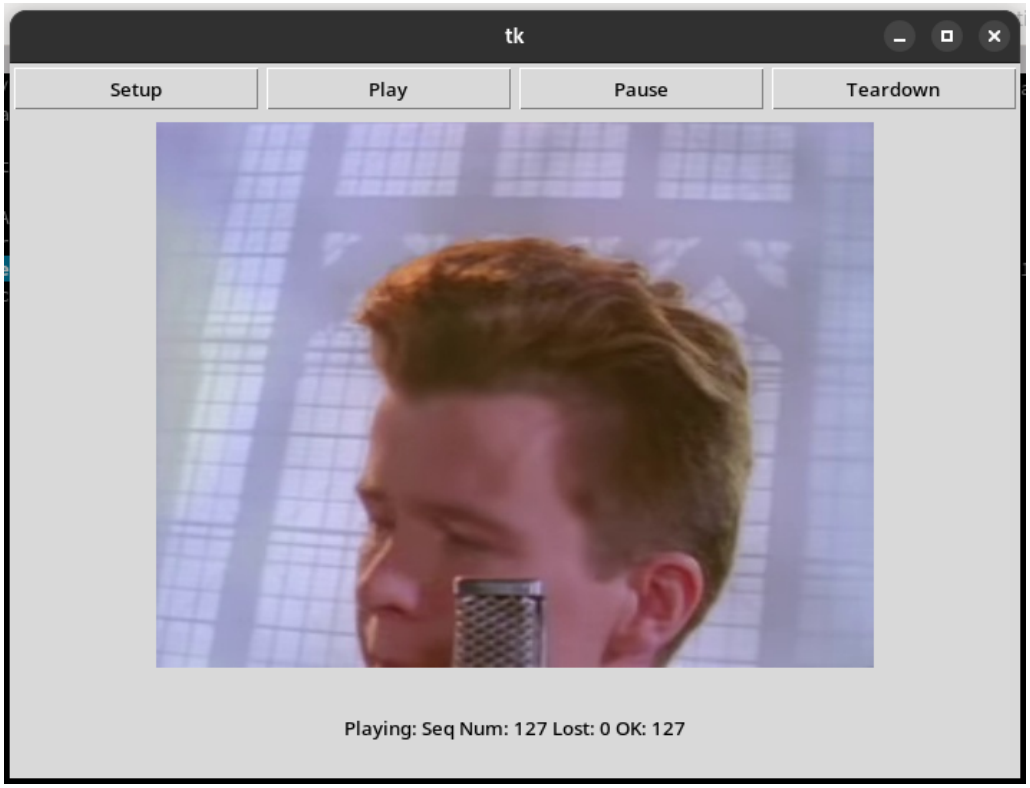


Figura 4: Reproducció/Pausa

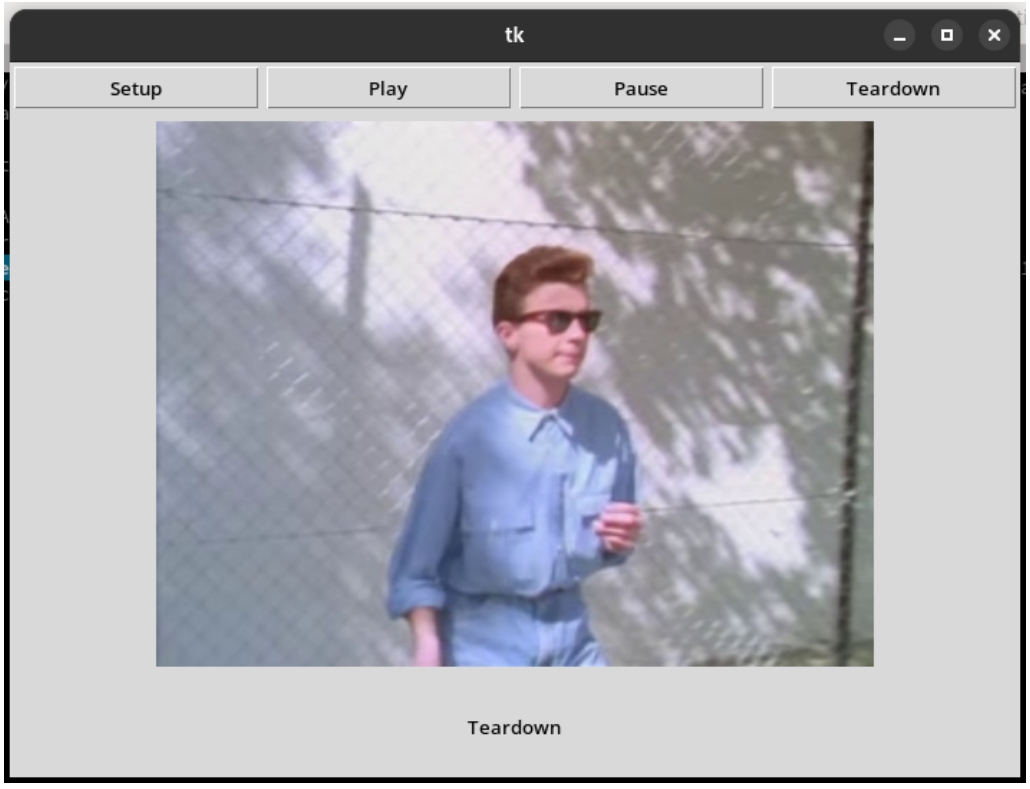


Figura 5: Fase de Teardown

- **Click**: Llibreria de processament de paràmetres de línia d'ordres. L'empro al codi que us he donat d'exemple.
- **Icecream**: Llibreria que us pot ajudar a depurar el programa si feu *print-debugging*
- **Loguru**: Llibreria que substitueix la logging de Python. L'he emprat al programa que us he passat.

NO està permés fer servir cap llibreria de comunicacions que no sigui la socket que forma part de Python.

Heu d'emprar, per mantenir la vostra salut mental quan desenvolupeu, no només en Python, sinó en la major part de llenguatges, alguna eina de gestió de dependències, entorns virtuals, etc. o bé recòrrer a desenvolupar en DevContainers, distrobox, etc. En el meu cas, en aquest projecte, he emprat, indistintament (segons l'ordinador on estava):

- **virtualenv/virtualenvwrapper**: gestor d'entorns virtuals de Python, per mantenir isolades les llibreries del sistema.
- **pyenv**: Gestor de versions de Python, per tenir diverses versions de Python a la mateixa màquina.
- **poetry**: Gestor d'entorns virtuals i projectes Python.

Possiblement, si no n'heu fet servir mai, poetry sigui el més adient per vosaltres.

Lliurament

Documentació i codi

La pràctica és individual, cada alumne haurà de lliurar un únic arxiu amb les següents característiques:

- *Nom*: **XARXES-P1-***nom* (On *nom* son els dos cognoms de l'autor separats per guionet (-))
- *Format*: ZIP, TGZ o TBZ.
- *Contingut*:
  - Codi font degudament comentat pel seu seguiment.
  - Arxius necessaris pel correcte funcionament del codi (configuracions clients, servidor, etc).
  - Un informe en format PDF que inclogui:
    1. L'estructura, a nivell esquemàtic (diagrama de blocs), del client i del servidor.
    2. Totes aquelles consideracions que cregueu oportunes.

Aquest document ha de complir les següents condicions:

* Una bona estructuració.
* Un format dels elements de text correctes.
* Un contingut clar i una redacció correcta (no s'acceptaran errades ortogràfiques ni abreviatures o símbols per substituir paraules).

- S'ha de lliurar al campus virtual (apartat Activitats) i no s'acceptarà per cap altre mitjà.

> **IMPORTANT**
>
> **La documentació que no compleixi tots aquests requisits no serà avaluada.**

Termini

El termini per rebre la documentació relacionada amb aquesta pràctica finalitza el dia **21 d'abril de 2025**(provisional).

Avaluació

Per a que una pràctica pugui ser avaluada haurà de complir els següents requisits mínims:

- El codi del servidor i del client no ha de donar cap tipus d'error de l'interpret de Python.
- El codi ha de ser mínimament llegible.
- S'executarà el servidor i dos clients en la mateixa màquina i, com a mínim, han d'arribar a la fase de PLAYING, enviant un datagrama RTP i refrescant un frame com a mínim.
- Complir tots els requeriments de l'apartat *Documentació* de la secció **Lliurament**

Es valorarà, entre altres coses:

- Claredat i estructura del codi. Això vol dir (no limitat a):
  - Poques repeticions i poca redundància
  - Estructures de dades adients
  - Noms i identificadors clars, indicant funció.
  - Funcions i mètodes de mida correcta.
  - Bona modularització.
  - El codi no ha de ser spaghetti-code.

- La solució no ha de ser excessivament complexa (no cal).
- La gestió de processos/threads no ha de ser complexa (no cal).
- Funcionament en els diversos casos:
  - Servidor alumne i 1 client alumne
  - Servidor alumne i n clients alumne
  - Servidor professors i 1 client alumne
  - Servidor professors i n clients alumne
  - Servidor alumne i 1 client professors
  - Servidor alumne i n clients professors

Opcional Mostrar estadístiques de paquets perduts

La avaluació de la pràctica es realitzara en un sistema Linux (RockyLinux 9.x). Podeu provar-la en aquest sistema o bé al laboratori o bé emprant màquines virtuals o contenidors.

Referències

Frederick, Ron et al. (gen. de 1996). *RTP: A Transport Protocol for Real-Time Applications*. RFC 1889. DOI: 10.17487/RFC1889. URL: https://www.rfc-editor.org/info/rfc1889.

Rao, Anup, Rob Lanphier i Henning Schulzrinne (abr. de 1998). *Real Time Streaming Protocol (RTSP)*. RFC 2326. DOI: 10.17487/RFC2326. URL: https://www.rfc-editor.org/info/rfc2326.

Schulzrinne, Henning, Stephen L. Casner et al. (jul. de 2003). *RTP: A Transport Protocol for Real-Time Applications*. RFC 3550. DOI: 10.17487/RFC3550. URL: https://www.rfc-editor.org/info/rfc3550.

Schulzrinne, Henning, Anup Rao et al. (des. de 2016). *Real-Time Streaming Protocol Version 2.0*. RFC 7826. DOI: 10.17487/RFC7826. URL: https://www.rfc-editor.org/info/rfc7826.