

# **Aplicații Multimedia**

## **Laborator 8**

### **VoIP. Conferință**

#### Cuprins

<b>VoIP – Voice over Internet Protocol .....</b>	<b>1</b>
Mod de funcționare VoIP .....	1
Avantajele VoIP .....	1
<b>WebRTC .....</b>	<b>2</b>

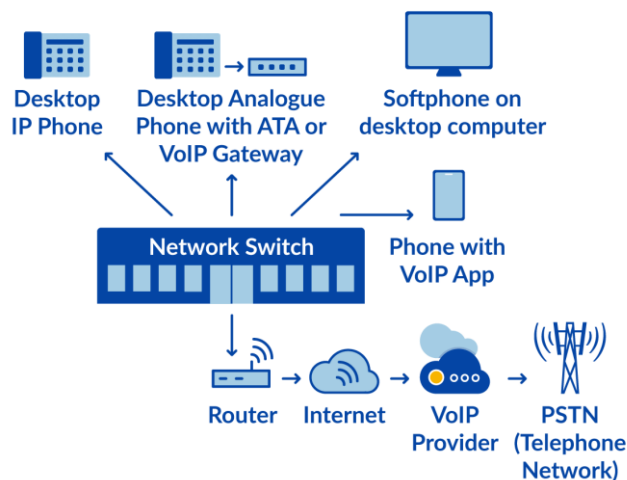
## VoIP – Voice over Internet Protocol

VoIP, după cum se poate deduce și din nume, este utilizat pentru a asigura transmiterea de conținut multimedia sau audio în cadrul rețelelor ce utilizează datagrame IP. Practic, VoIP oferă utilizatorilor posibilitatea de a iniția apeluri folosind o conexiune la Internet. Popularitatea se datorează calității informației transmise și necesității transmiterii pe același canal a tipurilor de date diverse precum audio, video, text. Din acest motiv este utilizate de către aplicații foarte cunoscute ca Skype, WhatsApp, Zoom, etc. Astfel, VoIP reprezintă un element primordial în evoluția conceptului clasic de funcționare a unui apel telefonic ce folosea PSTN (Public Switch Telephone Network).

### Mod de funcționare VoIP

Serviciul VoIP convertește semnalul analog vocal într-un semnal digital. După ce acesta trece printr-o procedură de compresie, este trimis în rețea sub formă de pachete ce vor străbate calea generată de către algoritmul de rutare până la destinație. O configurație clasică de tip VoIP este alcătuită de regulă dintr-un host de rețea (sau un telefon clasic conectat la un adaptor) și un server SIP (*Session Initiation Protocol*), ce va juca rolul provieder-ului de serviciu VoIP. O astfel de infrastructură poate fi observată în imaginea de mai jos.

### Hosted VoIP Network Infrastructure



Figură 1. Infrastructura rețea pentru VoIP

### Avantajele VoIP

Câteva din avantajele VoIP sunt:

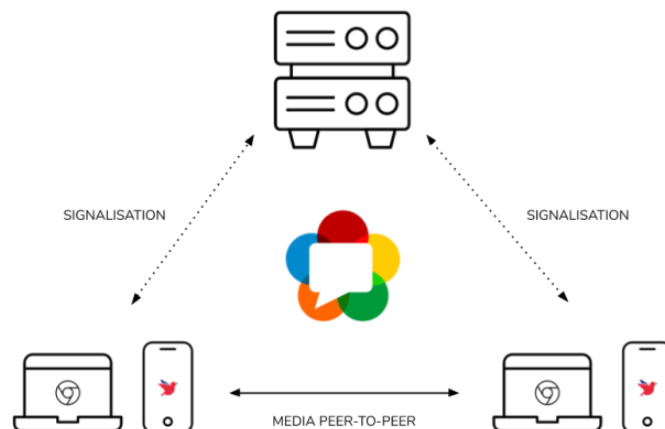
- Schimbarea de conținut media se poate realiza într-un mediu sigur și privat
- Costurile generate de companiile clasice de telefonie sunt reduse
- Capacitatea de inițiere a apelurilor video

## WebRTC

WebRTC este un framework ce utilizează serviciul VoIP și oferă posibilitatea de a dezvolta soluții software destinate comunicației vocale și video în timp real. Această tehnologie este implementată ca un standard web opensource disponibil sub formă de API în limbajul JavaScript, dar și pentru diferite sisteme de operare. Proiectul este compatibil cu majoritatea browserelor existente și pune la dispoziția utilizatorului capacitatea de a realiza o comunicație audio și video în cadrul unei pagini web utilizând o conexiune de tip peer-to-peer.

Cu ajutorul lui se pot dezvolta aplicații care folosesc ca intrare semnalul captat de cameră sau microfon, dar și funcționalități mai complexe precum partajarea ecranului. De astfel scopul său principal este acela de a contribui la realizarea de aplicații RTC de înaltă performanță, capabile să funcționeze din browser. O astfel de aplicație bazată pe WebRTC vom dezvolta și noi în cadrul laboratorului de astăzi.

Arhitectura pe care se bazează WebRTC este una de natură triangulară fiind formată din 2 clienți și un server.



## Implementarea unei conferințe (video + audio) folosind WebRTC și Python

Primul exercițiu pe care îl vom face va fi să pornim o conferință între 2 browsere sau 2 pagini de browser de pe același calculator.

Pentru a putea implementa o sesiune de conferință bazate pe WebRTC în browser, vom folosi următoarele fișiere (preluate de pe acest [repo](#)):

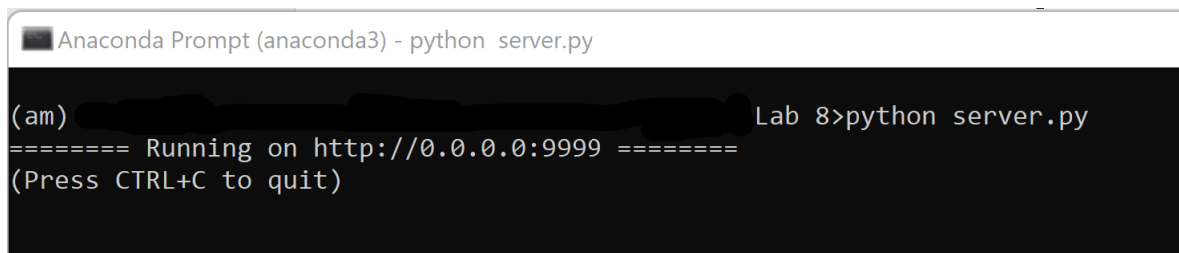
- server.py
- web (director)
  - index.html
  - main.js
  - socket.io.js

Pentru rularea serverului de signaling în Python (al cărui cod este în fișierul `server.py`), este necesară instalarea pachetelor `aiohttp` și `python-socketio`:

***pip install aiohttp python-socketio***

După instalare, putem rula serverul folosind ***python server.py*** în consola de Anaconda. Aceasta va porni serverul de signalling pentru conexiunea de WebRTC pe portul 9999.

Următorul pas este să pornim un server în directorul **web** care să hosteze pagina principală `index.html`. Aceasta este o pagină goală, care va conține feed-ul video de la camera celui alt participant, după conectare. Putem face acest lucru folosind comanda ***python -m http.server 7000*** în directorul **web**. Aceasta va face posibilă accesarea paginii `index.html` accesând în browser `localhost:7000`.



```
Anaconda Prompt (anaconda3) - python server.py

(am) Lab 8>python server.py
===== Running on http://0.0.0.0:9999 =====
(Press CTRL+C to quit)
```

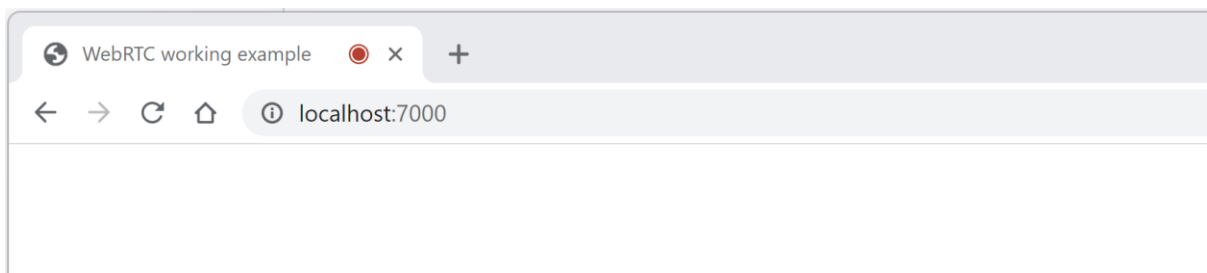
*Pas 1*



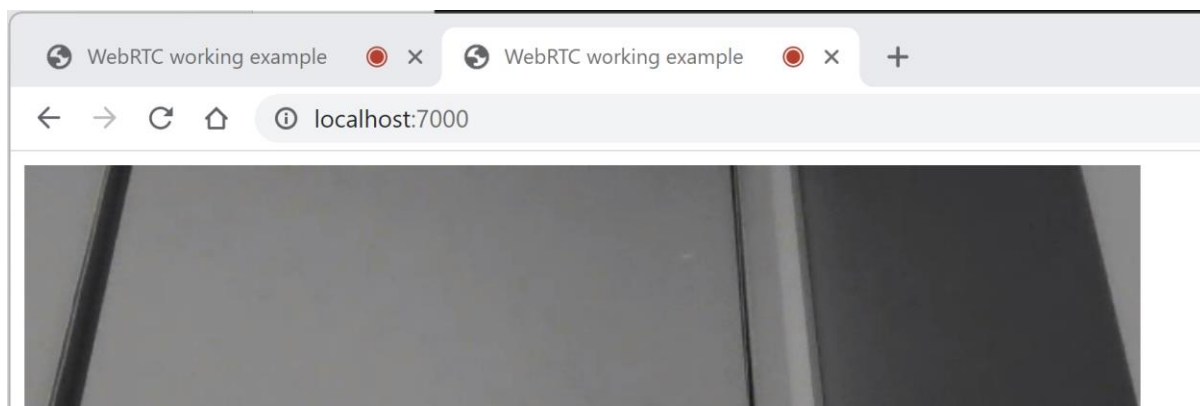
```
Anaconda Prompt (anaconda3) - python -m http.server 7000

(am) Lab 8\web>python -m http.server 7000
Serving HTTP on :: port 7000 (http://[::]:7000/) ...
```

*Pas 2*



*Pas 3*



Pas 4

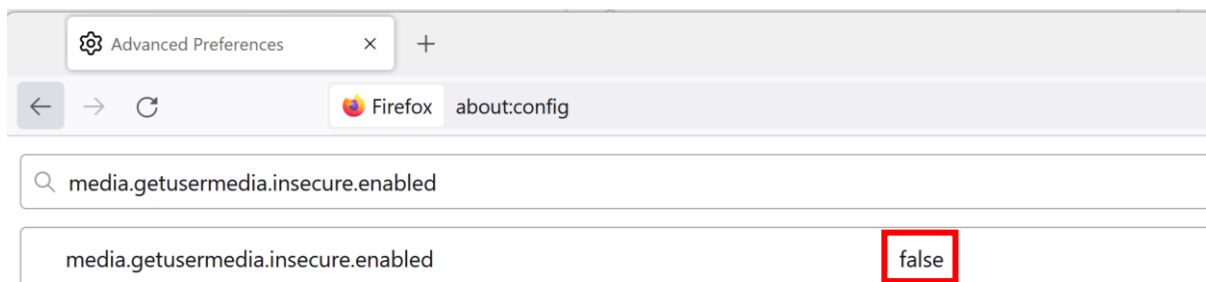
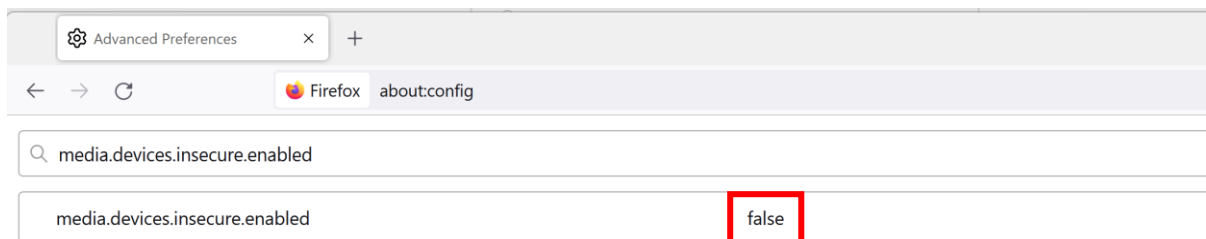
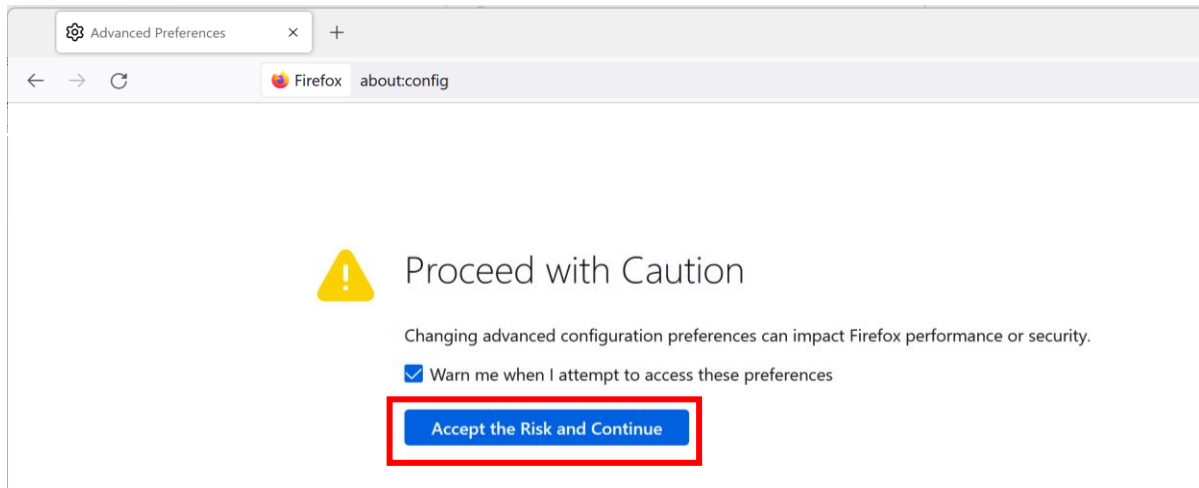
**Atenție!** La acest punct este recomandată utilizarea căștilor sau păstrarea volumului pe 0, pentru a evita microfonia care se poate produce, având în vedere că ambele microfonul și difuzoarele sunt resurse comune pentru ambii clienți.

Putem analiza traficul schimbat atât cu serverul de signaling din Python, cât și cu serverul pentru pagina principală, pornind o captură în Wireshark și urmărind traficul pe porturile 9999 și 7000. Ce protocol este utilizat?

Pentru cel de-al doilea exercițiu, vom încerca să pornim o sesiune între 2 calculatoare diferite, însă conectate la aceeași rețea, prin intermediul IP-ului.

Primul pas necesar va fi instalarea browser-ului Mozilla Firefox, dacă nu există pe sistem: <https://www.mozilla.org/en-US/firefox/new/>. Acest lucru este necesar deoarece procedeul pe care în vom folosi are doar scop didactic, neavând toate componentele necesare pentru a fi o aplicație finală de conferință online, iar unele browsere (cum ar fi Google Chrome sau Microsoft Edge), ca urmare a procedurilor de securitate, vor face imposibilă rularea.

Următorul pas este configurarea în browser-ul Firefox a două opțiuni pentru a permite preluarea de date de la dispozitive media conectate la calculator și transmiterea acestora peste o conexiune nesecurizată. Puteți face acest lucru accesând în URL **about:config** și în lista care apare să căutați **media.devices.insecure.enabled** și **media.getusermedia.insecure.enabled** și să le setați pe ambele pe **true**. Dacă plănuți să utilizați în continuare browserul, este recomandată resetarea lor înapoi pe **false**.



După ce ați format perechi de câte 2, următorul pas este modificarea în fișierul main.js a adresei serverului de signaling, astfel încât acesta să nu mai fie asociat adresei de loopback (localhost), ci adresei IP a calculatorului pe care va rula serverul. Pe același calculator veți porni serverul de signaling și serverul pentru pagina web.

În browserul de pe calculatorul pe care ați pornit serverul, puteți accesa pagina în același mod (localhost:7000), însă pentru cel de-al doilea calculator este necesară menționarea adresei IP în URL (192.168.0.15:7000).

