

TRANSMISIA DATELOR - APLICAȚII

---

## Sisteme de comunicații cu fir

---

# 2 SISTEME DE COMUNICAȚII CU FIR

## 2.1 OBIECTIVE

- Familiarizarea cu mediile de transmisie cu fir;
- Dezvoltarea unei aplicații client / server utilizând protocoalele http și websocket.

## 2.2 INTRODUCERE

### 2.2.1 MOTIVAȚIE

### 2.2.2 MEDII DE TRANSMISIE CU FIR

Medii ghidate Mediile de transmisie a informației se clasifică în funcție de tipul de semnale transportate. Avantajele transmisiei prin fir sunt:

- viteza de transmisie - raportul dintre cantitatea de informație transmisă și timpul necesar transmisiei;
- lățimea de bandă - cantitatea de informație pe care o poate transmite.
- distanța pe care se face transmisia - distanța dintre emițător și receptor pe care se face transmisia de date fără atenuări sau erori.

#### 2.2.2.1 CABLUL COAXIAL

Banda de transmisie: 0 - 600 MHz

Este realizat dintr-un fir de cupru (miez) prin care informația este transmisă și o plasă de sârmă cu rol izolator la interferențe, separate printrul manșon de plastic. Informația transmisă este analogică sau digitală. [2.1](#)

Cele mai uzuale cabluri coaxiale, [2.2](#), [\[1\]](#) (RG - radio guide; /U - utilizare generală):

- RG-58 /U - utilizat pentru semnale low power si conexiuni RF, are rezistența de 50 Ohm și impedanța de 25 pF/ft (82 pF/m);
- RG-59 /U - utilizat pentru semnale video low power si conexiuni RF, are rezistența de 75 Ohm și impedanța de 20pF/ft (60pF/m); Cel mai des utilizat pentru distribuție CC TV.
- RG-62 /U - utilizat în aplicații industriale de interior pentru a transmite semnale de frecvențe înalte și de putere, are rezistența de 93 Ohm;

## CAPITOLUL 2. SISTEME DE COMUNICAȚII CU FIR



Figura 2.1: Cablul coaxial

- RG-6/U - utilizat în aplicații rezidențiale, are rezistența de 75 Ohm; Cel mai des utilizat pentru distribuție TV.
- RG-11/U - utilizat în aplicații industriale de interior pentru a transmite semnale de frecvențe înalte și de putere, are rezistența de 75 Ohm. Cel mai des utilizat pentru distribuție HD TV (High Definition TV).

Conectori, [2.3](#), [2.4](#)

Avantaje și dezavantaje:

- + ieftin și durabil;
- + ușor de folosit;
- + rezistență bună la EMI;
- + viteze de transfer până la 10 Mbps;
- mentenanța - un cablu defect, întreaga rețea este inactivă.

## CAPITOLUL 2. SISTEME DE COMUNICAȚII CU FIR



Figura 2.2: Cablul coaxial



Figura 2.3: Conectori, (Sursa: [3], [4])

### 2.2.2.2 CABLUL TORSADAT

Cablul torsadat este format din mai multe perechi de fire de cupru izolate și se utilizează în comunicații. Torsadarea este o protecție electromagnetică, pentru reducerea sau anularea semnalelor parazite produse de alte circuite electrice învecinate și se obține prin răsucirea



Figura 2.4: Conectori, (Sursa: [3], [4])

a două cabluri 2.5.

Un curent electric de intensitate variabilă produce la trecerea printr-un conductor, un câmp magnetic variabil. De asemenea, un câmp magnetic variabil induce într-un conductor o tensiune electrică variabilă. [5]

Dacă prin două conductoare electrice (ce închid un circuit) se transmite un semnal electric variabil, atunci curenții care le parcurg vor avea sensuri opuse. Aceștia vor genera două câmpuri electromagnetice în antifază care se vor anula reciproc.

Tipuri de cabluri torsadate (cele mai răspândite):

- FTP - Cablu cu perechi rasucite în folie - Foiled Twisted Pair;
- UTP - Cablu bifilar torsadat neecranat - Unshielded Twisted Pair, 2.6;
- STP - Cablu bifilar torsadat ecranat - Shielded Twisted Pair;
- S/UTP - Screened Unshielded Twisted Pair;
- S/FTP - Screened Foiled Twisted Pair;

## CAPITOLUL 2. SISTEME DE COMUNICAȚII CU FIR

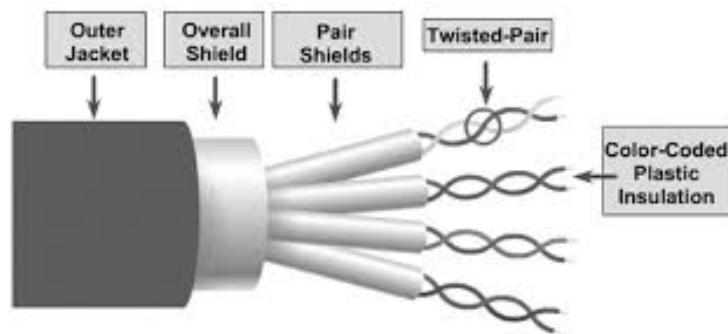


Figura 2.5: Cablul torsadat, 4 perechi

- S/STP: Screened Shielded Twisted Pair, [2.7](#).

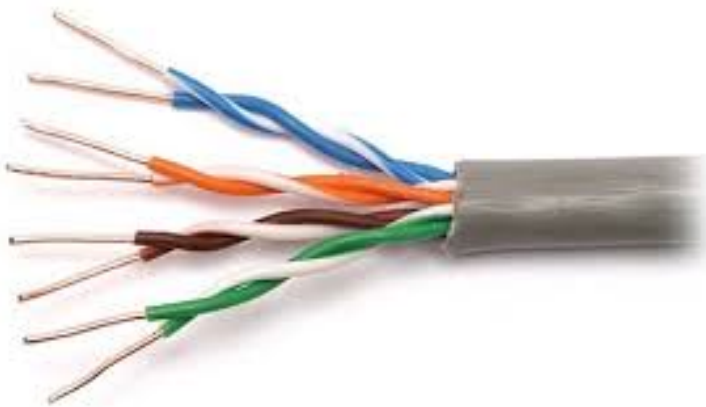


Figura 2.6: Cablul UTP, 4 perechi

Categorii de cabluri torsadate (Electronic Industries Association (EIA) a stabilit standardele UTP în 1995, [\[6\]](#):

- CAT 1 - Categoria 1, se utilizează în rețele de telefonie. Se pot transmite doar semnale codificate analogic, cu frecvențe de până la 0,4 MHz și viteze de transmisie mai mici de 100 Kbps; Nu a fost proiectat pentru transmisia datelor.
- CAT 2 - Categoria 2, cablu cu 4 perechi torsadate. Se utilizează pentru semnale analogice și digitale cu viteze de transmisie de până la 4 Mbps; Utilizat în rețele LAN vechi (Token-passing ring LANs, IEEE 802.5). Lățime de bandă de 1 MHz.

## CAPITOLUL 2. SISTEME DE COMUNICAȚII CU FIR



Figura 2.7: Cablul S/STP, 4 perechi

- CAT 3 - Categoria 3, cablu cu 4 perechi torsadate, având 7-8 răsuciri pe metru; Se utilizează pentru viteze de transmisie Ethernet de până la 10 Mbps și semnale cu frecvențe de până la 16 MHz. Utilizat în vechile rețele Ethernet 10base-T LANs (IEEE 802.3).
- CAT 4 - Categoria 4, cablu cu 4 perechi torsadate. Se utilizează în rețele cu viteze de transmisie de până la 16 Mbps și frecvențe de până la 20 Mhz; Utilizat în rețele token-based sau 10Base-T.
- CAT 5 - Categoria 5, cea mai utilizată categorie de cablu în rețele Ethernet, având 4 perechi torsadate la un număr de 3 -4 răsuciri pe inch. Poate fi utilizat pentru viteze de transmisie de până la 155 Mbps sau frecvențe de până la 100 MHz pe distanțe de maxim 100 m. Transportă date cu viteze de până la 100 Mbps (Fast Ethernet, IEEE 802.3u) și este utilizat pentru rețele 100 base-T și 10base-T.
- Categoria 5e – cablu cu 4 perechi torsadate. Poate fi utilizat pentru viteze de transmisie de până la 1 Gbps sau frecvențe de până la 100 MHz pe distanțe de maxim 100 m. Pentru 100BASE-TX și 1000BASE-T Ethernet.

## CAPITOLUL 2. SISTEME DE COMUNICAȚII CU FIR

- CAT 6 - Categoria 6, cablu cu 4 perechi torsadate. Poate fi utilizat pentru viteze de transmisie de până la 10 Gbps sau frecvențe de până la 250 MHz pe distanțe de maxim 100 m. Pentru 10GBASE-T Ethernet.
- CAT 6a - Categoria 6a, cablu cu 4 perechi torsadate. Poate fi utilizat pentru viteze de transmisie de până la 10 Gbps sau frecvențe de până la 500 MHz pe distanțe de maxim 100 m. Pentru 10GBASE-T Ethernet.
- CAT 7 - Categoria 7, suportă până la 10 Gbps și frecvențe de până la 600 MHz. Pentru telefonie, CCTV, 1000BASE-TX în același cablu, 10GBASE-T Ethernet. Folosește F/FTP și S/FTP.
- CAT 7a - Categoria 7a, suportă până la 10 Gbps și frecvențe de până la 1000 MHz. Pentru telefonie, CATV, 1000BASE-TX în același cablu, 10GBASE-T Ethernet. Folosește F/UTP.
- CAT 8.1 - Categoria 8.1, suportă până la 40 Gbps și frecvențe de până la 1600 - 2000 MHz. Pentru telefonie, CATV, 1000BASE-TX în același cablu, 40GBASE-T Ethernet. Folosește F/UTP.
- CAT \*.2 - Categoria 8.2, suportă până la 40 Gbps și frecvențe de până la 1600 - 2000 MHz. Pentru telefonie, CATV, 1000BASE-TX în același cablu, 40GBASE-T Ethernet. Folosește F/FTP și S/FTP.

Conectori UTP, FTP (standardele A și B de mufare), [2.8](#)

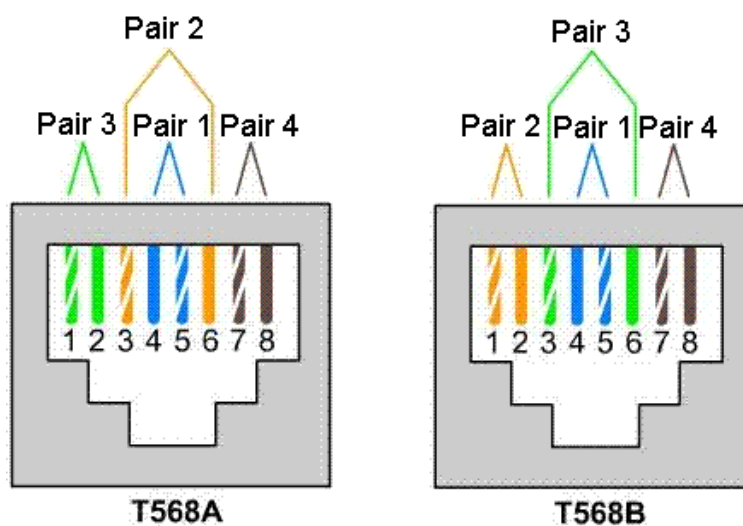


Figura 2.8: Conectori RJ-45, standardele 568A și 568B de mufare, (Sursa: [7], [8])



## CAPITOLUL 2. SISTEME DE COMUNICAȚII CU FIR

Avantaje și dezavantaje:

- + ieftin și ușor de instalat;
- + mai puțin sensibil la interferență cauzate de echipamentele din apropiere și la rîndul lor nu provoacă interferențe;
- + STP poate transporta date la viteze mai mari;
- STP e mai voluminos și mai scump decât UTP;
- STP e mai dificil de mufat.

### 2.2.2.3 FIBRA OPTICĂ

Un cablu de fibră optică este format din fire de sticlă sau plastic (fibră optică); un singur cablu poate avea până la câteva sute de fire, 2.9. Miezul fibrei optice are dimensiuni extrem de reduse, 2 - 125  $\mu\text{m}$ . Un singur fir poate transmite mai multe mii de apeluri telefonice. Capacitate de transmisie de ordinul gigabitilor pe mai multi kilometri avînd atenuări foarte mici (0.2 până la 1 dB/km) și o imunitate la zgomot foarte ridicată.

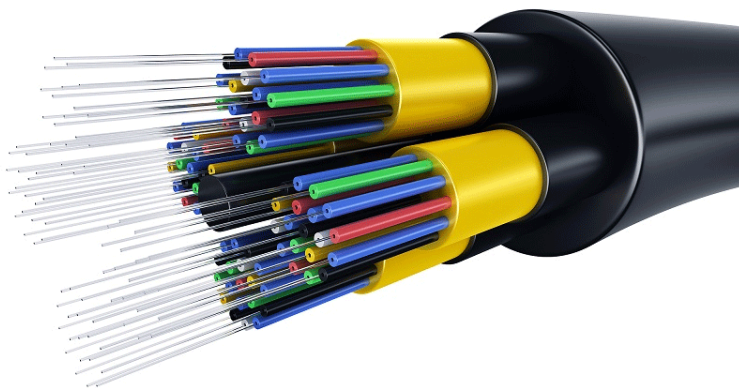


Figura 2.9: Fibră optică, (Sursa: [9])

Tipuri de fibră optică:

- fibră optică multi-mod (LED); Are un miez de 62.5 $\mu\text{m}$  sau 50 $\mu\text{m}$ , permițînd moduri multiple de propagare a lumini.

Cablurile multi-mode pot trimite informații doar pe distanțe relativ scurte și sunt folosite pentru interconectarea a două rețele.

## CAPITOLUL 2. SISTEME DE COMUNICAȚII CU FIR

- fibră optică mono-mod (laser); Are un miez de diametru în jurul valorii de  $9\mu\text{m}$  și transmite doar un singur fascicul de lumină cu o lungime de undă specifică.

Televiziunea prin cablu, Internetul și semnalele telefonice sunt în general realizate prin fibre single-mode. Pot trimite informații la distanță de peste 100 km (60 mile).

Unele fibre optice premium au o degradare mult mai mică a semnalului - mai puțin de 10% / km la 1.550 nm.

Semnalele digitale sunt codificate în impulsuri analogice de lumină, NRZ – “Non-Return to Zero”.

Cele mai multe fibre funcționează în duplex:

- una pentru transmisie, și
- una pentru recepție.

Conectori fibră optică, 2.10.

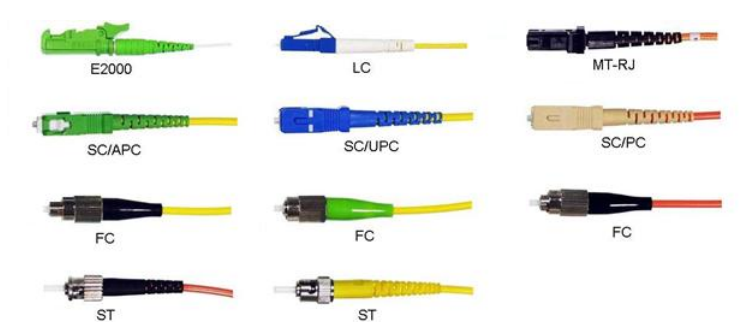


Figura 2.10: Conectori fibră optică, (Sursa: [10])

Avantaje și dezavantaje:

- + volum redus în comparație cu celelalte cabluri;
- + viteza mai mare;
- + lățimea de bandă mult mai mare;
- + securitatea datelor;
- + imună la EMI;
- costuri ridicate;
- dificultate la cablarea unei noi rețele;
- fibra optică trebuie slefuită cu precizie pentru a transmite lumina cu pierderi mici.

### 2.2.3 TOPOLOGII DE REȚEA

Există mai multe topologii de rețea, [2.11](#):


- Magistrală - Într-o rețea de tip magistrală toate nodurile sunt conectate în linie.
- Stea - Toate nodurile sunt conectate individual cu un nod central.
- Inel - Fiecare nod este conectat cu două noduri diferite.
- Arbore - E o combinație între topologiile magistrală și stea.
- Fiecare cu fiecare - Fiecare nod este conectat cu fiecare nod în parte.
- Punct la punct - Două noduri cu drepturi egale sunt conectate între ele.
- Client / Server - Serverul este un dispozitiv cu putere de calcul mai mare și nodurile client au putere de calcul redusă în comparație cu serverul.

### 2.2.4 APLICAȚII WEB

În prezent există mai multe tipuri de aplicații web, de la pagini web clasice, care utilizează o tehnologie de backend (php, java) pentru a gestiona și genera resurse web (html, css, js) până la aplicații complexe (Single page applications) care sunt executate izolat în browser și care sunt complet decuplate de modulul api care gestionează cererile acestor clienți și le expune o interfață de comunicare prin care se transferă datele. În continuare ne vom referi la al doilea tip de aplicații web pentru care există mai multe tehnologii actuale JavaScript care aduc modele arhitecturale software cât și mai multe funcționalități uzuale cum ar fi: arhitectura MVC (Model View Controller), legătură bidirecțională între modelul de date (variabile JavaScript) și interfața grafică, sintxă HTML extinsă care facilitează afișarea datelor pe interfață.

### 2.2.5 ARHITECTURA CLIENT-SERVER

Arhitectura client-server este o arhitectură distribuită compusă din module client (care gestionează resurse) și servere (care oferă o interfață de acces la resurse). Clienții și serverele sunt aplicații software executate de cele mai multe ori pe mașini diferite. De asemenea, serverele operează în strânsă legătură cu diverse mecanisme de stocare a datelor (ex: server SQL, MongoDB, MySQL).

 Câteva exemple de resurse sunt: utilizatori ai aplicației, produsele afișate într-un magazin online, posturile/tweeturile de pe twitter/facebook, imaginile de pe

## CAPITOLUL 2. SISTEME DE COMUNICAȚII CU FIR

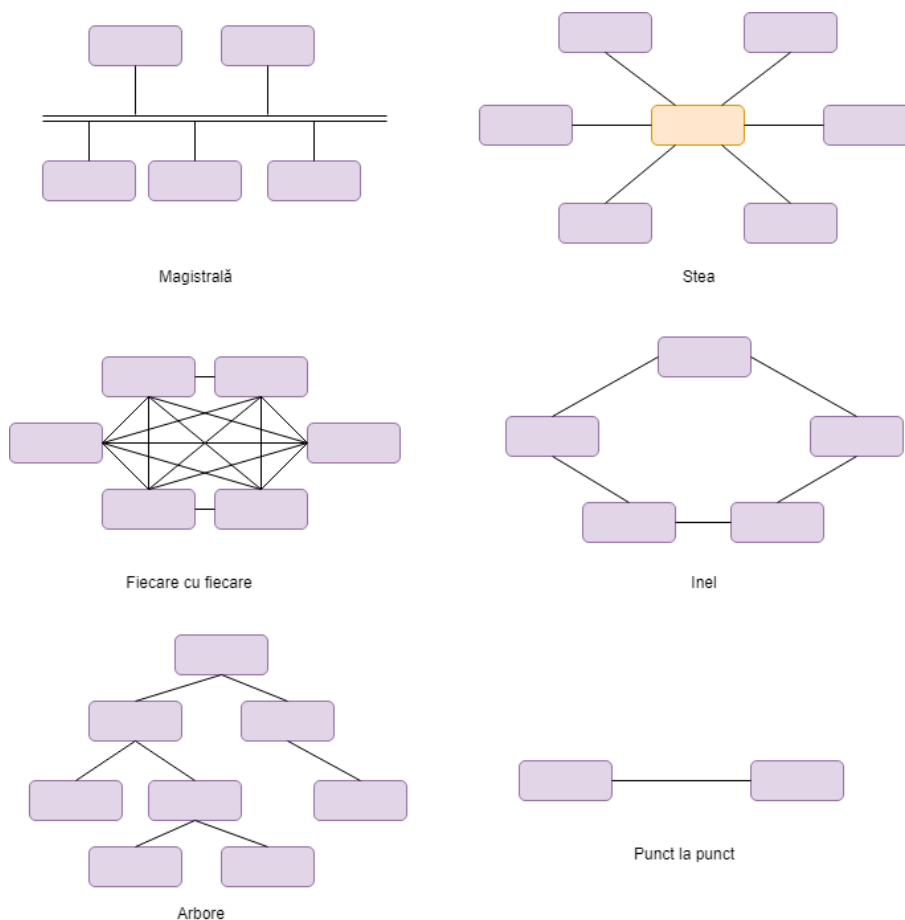


Figura 2.11: Topologii de rețea, (Sursa: [11])

instagram, emailurile.

Clienții (cunoscuți ca și aplicații frontend) gestionează datele la nivelul interfeței grafice și inițiază sesiuni de comunicație cu serverele (cunoscuți ca și aplicații backend) așteaptă cereri (incoming requests) de la aceștia.

### 2.2.6 TCP/IP

Orice dispozitiv conectat la o rețea TCP (Transport Control Protocol)/IP are o adresă unică de identificare denumită IP (Internet Protocol).

Pentru ca o mașină (server, calculator) să poată expune mai multe aplicații și servicii pe rețea este nevoie, pe lângă adresa IPO, de mai multe porturi prin care mașina să transfere date.

## CAPITOLUL 2. SISTEME DE COMUNICAȚII CU FIR

IP-ul identifică serverul iar portul identifică aplicația sau serviciul care este executat pe server.

Figura următoare prezintă conectarea dintre două calculatoare pe rețeaua Internet folosind IP-uri și porturi.

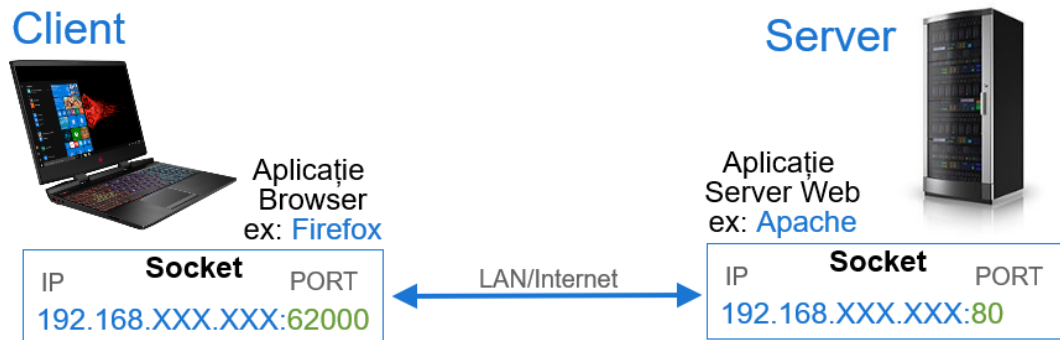


Figura 2.12: Conectare aplicație browser (client) - server web

Porturile iau valori pe 16 biți între 0 to 65535 și sunt grupate astfel:

- 0-1023: porturi alocate pentru servicii de sistem. Cele mai cunoscute porturi sunt: 80 (server web), 22 (SSH), 20 (FTP), 23 (SMTP).
- 1024-49151: porturi rezervate pentru aplicații client.
- 49152-65535: porturi dinamice/private.

O conexiune între două calculatoare folosește un socket (soclu) de comunicație care poate fi văzut ca o pereche de adresare IP:port. Așa cum reiese din figura 2.12 calculatoarele se conectează între ele folosind socketuri.

De exemplu atunci când accesăm site-ul Google se realizează o conexiune sursă-destinație între aplicația browser de pe calculatorul nostru local (care are atribuit un socket de felul 192.168.XXX.XXX:62000) și un server din infrastructura Google care servește aplicația web (216.58.XXX.XX:80). Porturile de pe calculatorul personal sunt atribuite dinamic aplicațiilor pentru fiecare sesiune de comunicație cu un socket extern și pot fi refolosite o dată ce această sesiune s-a încheiat.

TCP și UDP sunt protocoale aflate la nivelul transport de date pe când IP este implementat la nivelul rețea. Porturile sunt implementate la nivelul transport de date ca parte din antetul mesajelor TCP sau UDP:

Protocolul TCP/IP suportă două tipuri de porturi TCP și UDP. TCP este un protocol de transport orinatat pe conexiune care are implementat un mesaj de retransmisie a mesajelor

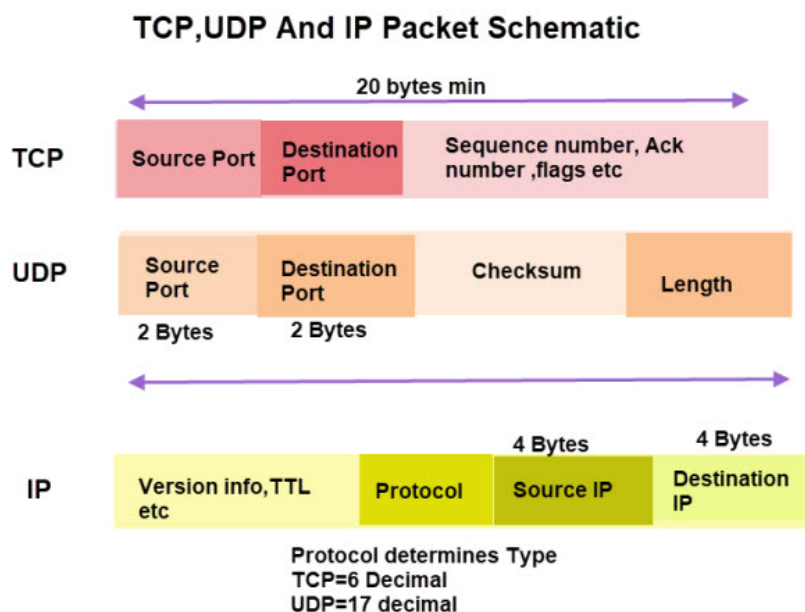


Figura 2.13: TCP, UDP, IP

în caz de eroare astfel încât toate packetele să ajungă la destinație. Este folosit pentru mesaje text, email, etc.

UDP este utilizat pentru transmisii multimedia (video, audio) și nu are implementat un mecanism care să asigure transferul cu succes al pachetelor de date.

### 2.2.7 HTTP vs. WEBSOCKET

Pentru transferul de date între diverse aplicații software conectate la rețea se folosesc în general socketuri (socluri de comunicație). Aceste socketuri pot să folosească protocoale de comunicație diverse din stiva TCP/IP cum ar fi: HTTP și WebSocket.

HTTP este un protocol unidirecțional care permite clienților să facă așa numitele cereri spre servere care gestionează resursele (eventual folosind o bază de date). Tipurile de cereri HTTP cele mai frecvent utilizate sunt: GET, PUT, POST și DELETE. Aceste 4 funcții permit citirea, salvarea, actualizarea și ștergerea resurselor.

WebSocket este o modalitate de a comunica bidirecțional între sisteme software distribuite. Cel mai simplu exemplu este un serviciu de chat în care clienții trimit mesaje spre server, care la rândul său le salvează și le trimite mai departe spre toți clienții interesați de aceste mesaje. Totuși prin websocketuri se pot trimite date audio și video.

JavaScript pune la dispoziție suport nativ pentru protocoalele HTTP și WebSockets pentru aplicații conectate la rețea.

### 2.3 DESFĂȘURAREA LUCRĂRII

Lucrarea constă în execuția unei aplicații compusă din două module: un modul client care expune către utilizator interfața grafică și un modul server care expune către modulul client puncte de interfațare (API). Modulele colaborează pentru a realiza împreună transmiterea de mesaje între utilizatori și gestiunea utilizatorilor care trimit mesaje prin intermediul modulului client. Comunicația între module se poate realiza în mai multe moduri și am ales să detaliem protocoalele HTTP și WebSocket.

Resurse folosite:

- biblioteca vue.js pentru modulul client
- biblioteca axios.js pentru modulul client
- mediul de execuție Node.js pentru modulul server
- npm - administratorul de pachete pentru Node.js

#### 2.3.1 APLICAȚIE CLIENT CU VUE.JS

Fișierul **index.html** conține:

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <meta content="text/html; charset=utf-8" http-equiv="Content
        -Type">
6      <meta content="utf-8" http-equiv="encoding">
7      <title>Vue first app</title>
8      <link rel="stylesheet" href="style.css">
9      <script type="text/javascript" src="libs/vue/vue.js"></
        script>
10 </head>
11
12 <body>
13     <div id="app">
14
```

## CAPITOLUL 2. SISTEME DE COMUNICAȚII CU FIR

```
15 <input class="message" v-model="message" placeholder="
    Name">
16
17 <div class="output-empty" v-if="message == ''">The
    message is empty</div>
18 <div class="output" v-if="message != ''"># {{ message }}
    #</div>
19
20
21 <button class="process-button" v-on:click="process()">
    Process</button>
22
23 </div>
24 <script src="app.js"></script>
25 </body>
26
27 </html>
```

Fișierul **app.js** conține:

```
1 var app = new Vue({
2   el: '#app',
3   data: {
4     message: ''
5   },
6   methods: {
7     process: function(){
8       console.log(this.message);
9     }
10  }
11 })
```

Fișierul **style.css** conține:

```
1 .message {
2   border: 1px solid #777;
3   padding: 7px;
4   margin: 5px;
5 }
6
7 .output-empty {
8   padding: 7px;
9   margin: 5px;
10  color: rgb(255, 104, 34);
11 }
12
```



```

13 .output {
14     padding: 7px;
15     margin: 5px;
16     color: rgb(27, 189, 27);
17 }
18
19 .process-button {
20     border: 1px solid #777;
21     background-color: #555;
22     color: white;
23     padding: 15px;
24 }
25
26 .process-button:hover {
27     background-color: #999;
28     cursor: pointer;
29 }

```

### 2.3.2 APLICAȚIE CLIENT-SERVER CU HTTP

Această aplicație realizează transmiterea de date folosind protocolul HTTP și, așa cum am zis, este împărțită în două module: modulul client și modulul server.

#### 2.3.2.1 MODULUL SERVER

Modulul server este alcătuit din 4 fișiere: **api.js**, **run.js**, **users.json** și **package.json**.

Fișierul **api.js** conține:

```

1  var express = require('express');
2  var cors = require('cors');
3  var app = express();
4  app.use(cors());
5
6
7  var bodyParser = require('body-parser');
8  app.use(bodyParser.urlencoded({
9      extended: true
10 }));
11 app.use(bodyParser.json());
12
13 exports.app = app;

```

Fișierul **run.js** conține:

## CAPITOLUL 2. SISTEME DE COMUNICAȚII CU FIR

```
1 var api = require('./api.js').app;
2 var users = require('./users.json');
3
4 api.get('/', function(request, response) {
5     response.json("node.js backend");
6 });
7
8 api.get('/users', function(request, response) {
9     response.json(users);
10 });
11
12 api.put('/users', function(request, response) {
13     users[users.length] = request.body;
14     response.json('User was saved succesfully');
15 });
16
17
18 api.delete('/users/:index', function(request, response) {
19     users.splice(request.params.index, 1);
20     response.json('User with index ' + request.params.index + '
21         was deleted');
22 });
23
24 api.listen(3000, function(){
25     console.log('CORS-enabled web server is listening on port
26         3000...');
```

Fișierul **users.json** conține:

```
1 [
2     {"name": "Cristina", "city": "Sebes"},
3     {"name": "Ion", "city": "Turda"},
4     {"name": "Sebastian", "city": "Bistrita-Nasaud"}
5 ]
```

Fișierul **package.json** conține:

```
1 {
2     "name": "it-prototype",
3     "description": "Node-Express Server",
4     "private": true,
5     "version": "0.0.1",
6     "dependencies": {
7         "body-parser": "^1.12.2",
8         "cors": "^2.8.5",
```

## CAPITOLUL 2. SISTEME DE COMUNICAȚII CU FIR

```
9     "express": "3.x"
10   },
11   "main": "Prototype",
12   "devDependencies": {}
13 }
```

Pentru a instala dependențele externe se execută comanda: **npm install** în linia de comandă.

Pentru a executa modulul server se execută comanda: **node run.js**.

Serverul răspunde requesturilor declarate în fișierul run.js (ex: <http://localhost:3000/users>)

### 2.3.2.2 MODULUL CLIENT

Modulul client este alcătuit din 4 fișiere: **index.html**, **app.js**, **users.js** și **style.css**. Modulul include librăriile externe vue.js și axios.js adaugate ca fișiere separate.

Fișierul **index.html** conține:

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5    <meta content="text/html; charset=utf-8" http-equiv="
      Content-Type">
6    <meta content="utf-8" http-equiv="encoding">
7    <title>Users manager</title>
8    <link rel="stylesheet" href="style.css">
9    <script type="text/javascript" src="libs/vue/vue.js"></
      script>
10   <script type="text/javascript" src="libs/axios/axios.js"></
      script>
11   <script type="text/javascript" src="users.js"></script>
12 </head>
13
14 <body>
15   <div id="app">
16
17     <ul class="user-list">
18       <li v-for="(user, index) in users">
19         <div class="user-row">
20           <div class="user-name"><b>{{ user.name }}</b></div>
21           <div class="user-city"><i>{{ user.city }}</i></div>
22         </div>
23       </li>
```

## CAPITOLUL 2. SISTEME DE COMUNICAȚII CU FIR

```
24     </ul>
25
26   </div>
27   <script src="app.js"></script>
28 </body>
29
30 </html>
```

Fișierul **app.js** conține:

```
1 var app = new Vue({
2   el: '#app',
3   data: {
4     users: [],
5     userService: null
6   },
7   created: function () {
8     userService = users();
9     userService.get().then(response => (this.users =
10      response.data));
11   },
12   methods: {
13   })
```

Fișierul **users.js** conține:

```
1 function users() {
2
3   get = function() {
4     return axios.get("http://localhost:3000/users");
5   }
6
7   return {
8     get: get
9   }
10 }
```

Fișierul **style.css** conține:

```
1 ul {
2   list-style-type: none;
3 }
4
5 .user-row {
6   margin: 20px auto;
7   background-color: rgb(255, 254, 234);
```

## CAPITOLUL 2. SISTEME DE COMUNICAȚII CU FIR

```
8 padding: 10px;
9 text-align: center;
10 width: 400px;
11 }
12
13 .user-row:hover {
14 background-color: rgb(255, 251, 175);
15 }
16
17 .user-name {
18 width: 150px;
19 display: inline-block;
20 }
21
22 .user-city {
23 width: 150px;
24 display: inline-block;
25 }
```

Pentru a executa modulul client este nevoie de un server care să servească resursele web. Un server simplu se poate instala pe platforma node.js executând următoarea comandă (din folderul clientului):

```
1 | npm install http-server -g
```

După ce s-a instalat server-ul, aplicația client se execută cu comanda:

```
1 | http-server
```

Aceasta va fi accesibilă în orice browser la adresele <http://localhost:8080> și <http://127.0.0.1:8080>.

### 2.3.3 APLICAȚIE CLIENT-SERVER CU WEBSOCKETS

#### 2.3.3.1 SERVER

Fișierul **package.json** conține:

```
1 {
2   "name": "chat-server",
3   "version": "0.0.1",
4   "description": "Data Transmission websocket api",
5   "dependencies": {},
6   "devDependencies": {
7     "socket.io": "^2.2.0"
8   }
9 }
```

## CAPITOLUL 2. SISTEME DE COMUNICAȚII CU FIR

Fișierul **server.js** conține:

```
1  const io = require("socket.io");
2  const server = io.listen(8000);
3  let connectedClients = new Map();
4
5
6  server.on("connection", (socket) => {
7      console.info(`Client connected [id=${socket.id}]`);
8      connectedClients.set(socket);
9
10
11     socket.on("disconnect", () => {
12         connectedClients.delete(socket);
13         console.info(`Client [id=${socket.id}] disconnected`);
14     });
15
16     socket.on("message-from-client", (payload) => {
17         sendMessageToAllOtherClients(socket, payload);
18     });
19
20 });
21
22 function sendMessageToAllOtherClients(sender, message) {
23     for (const [client, sequenceNumber] of connectedClients.
24         entries()) {
25         if (sender.id !== client.id) client.emit("message-from-
26             -server", message);
27     }
28 }
```

Dependințele specificate în package.json se descarcă folosind comanda:

```
1  npm install
```

Serverul se execută executând următoarea comandă în folderul în care se află serverul:

```
1  node server.js
```

### 2.3.3.2 CLIENT

Fișierul **package.json** conține:

```
1  {
2      "name": "chat-client",
```

```

3   "version": "0.0.1",
4   "description": "Data Transmission socket application",
5   "dependencies": {},
6   "devDependencies": {
7     "live-server": "^1.2.1",
8     "socket.io-client": "^2.2.0"
9   }
10 }

```

Fișierul **client.js** conține:

```

1  var socket = io.connect('localhost:8000');
2
3  try {
4
5      socket.on('connect', function (data) {
6          socket.emit("message-from-client", "Hello to everyone
7              from " + checkBrowser());
8      });
9
10     socket.on('message-from-server', function (message) {
11         console.log(message);
12     });
13 }
14 catch (err) {
15     alert('ERROR: socket.io encountered a problem:\n\n' + err
16         );
17 }
18
19 function checkBrowser() {
20     var browser = 'Noname browser';
21     if (navigator.userAgent.search("Chrome") > -1) {
22         browser = "Chrome";
23     }
24     if (navigator.userAgent.search("Firefox") > -1) {
25         browser = "Firefox";
26     }
27     return browser;
28 }
29
30 document.getElementById("send").addEventListener("click",
31     sendMessage);
32
33 function sendMessage() {

```

```

31 |     var message = document.getElementById("message").value;
32 |     socket.emit("message-from-client", message);
33 | }

```

Dependțele specificate în package.json se descarcă folosind comanda:

```
1 | npm install
```

Clientul se pornește executând următoarea comandă în folderul clientului:

```
1 | live-server
```

Acesta va fi accesibilă în orice browser la adresele <http://localhost:8080> și <http://127.0.0.1:8080>.

## 2.4 APLICATII SI PROBLEME

### 2.4.1 PROBLEMA 1 (1 punct)

- Aplicație client VueJS (0,5 p).
- La apăsarea butonului 'Process, dacă valoarea din câmpul message este egală cu '123' atunci se afișează pe interfața grafică "Mesajul este egal cu 123' (0,5 p).

### 2.4.2 PROBLEMA 2 (3 puncte)

- Aplicație client-server HTTP (1 p).
- Să se implementeze ștergerea unui user (1 p).
- Să se implementeze adăugarea unui user nou (1 p).
- Să se implementeze modificarea unui user existent (pe baza indexului său în șirul de elemente) (1 p).

### 2.4.3 PROBLEMA 3 (3 puncte)

- Testarea aplicației client-server cu WebSocket folosind 2 clienți deschiși în browsere diferite (1 p).
- Să se implementeze cu VueJS o interfață grafică pentru un serviciu de mesagerie (ex: facebook messenger). Serverul rutează mesajul primit de la un client spre toți clienții conectați iar aceștia îl afișează în browser (1 p), nu doar la consolă. Se dorește păstrarea pe interfața grafică a mesajelor primite/transmise.



# BIBLIOGRAFIE

- [1] <https://uk.rs-online.com>
- [2] <https://tri-vcables.com/custom-assemblies/coax/>
- [3] <http://www.l-com.com/content/Article.aspx?Type=L&ID=10057>
- [4] <http://www.coax-connectors.com/>
- [5] <https://cdn.kramerav.com>
- [6] <http://www.berkteklevitontechnologies.com>
- [7] <http://www.groundcontrol.com/galileo/ch5-ethernet.htm>
- [8] <https://www.fiberoptics4sale.com>
- [9] <http://www.netcom-activ.ro/fibra-optica.php>
- [10] [http://www.scritub.com/stiinta/informatica/retele/  
TEHNOLOGIA-RETELELOR-DE-COMUNI45325.php](http://www.scritub.com/stiinta/informatica/retele/TEHNOLOGIA-RETELELOR-DE-COMUNI45325.php)
- [11] <https://www.orosk.com/>