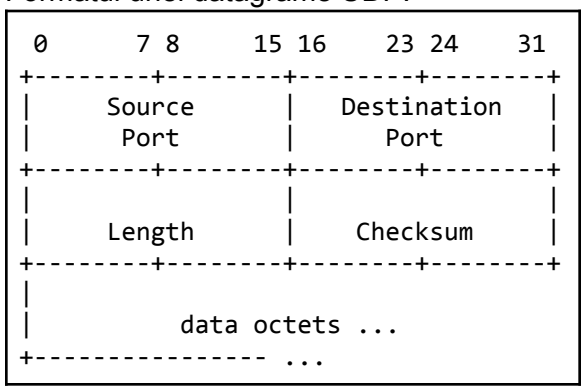
Lab 6 - Protocolul UDP

UDP

- Protocol de nivel 4 (transport) neorientat pe conexiune, care nu garanteaza ordinea primirii mesajelor sau prevenirea pierderii pachetelor
- Realizeaza **multiplexare la nivel de aplicatii** si **control al fluxului**
- Un segment UDP se numeste **datagrama**
- La nivelul transport, o aplicatie care poate comunica pe retea este definita de un **port** (numar pozitiv pe 2 octeti)
- Formatul unei datagrame UDP:



Socket

- Canal generalizat de comunicare inter-proces (peste retea) reprezentat in sisteme Unix printr-un **descriptor de fisiere**
- Poate fi vazut ca:
  - un capat al unei conexiuni intre doua aplicatii
  - o pereche **{adresa IP, port}**

Host si network byte order

- Endianness → ordonarea octetilor unui numar in memorie (ex. 256 = 0x0100)
  - big endian (MSB first) → 01 | 00 (procesoare IBM, Motorola, etc.)
  - little endian (LSB first) → 00 | 01 (procesoare Intel, etc.)
- In retea → **intotdeauna big endian**
- Cand **trimitem** date pe retea, convertim in big endian (**network order**):
  - htonl() → unsigned int pe 4 octeti
  - htons() → unsigned int pe 2 octeti
- Cand **primim** date de pe retea, convertim in ce avem pe masina gazda (**host order**)
  - ntohl() → unsigned int pe 4 octeti
  - ntohs() → unsigned int pe 2 octeti (toate functiile se afla in *arpa/inet.h*).

Comunicatie prin socketi UDP

- Structuri utilizate:

```
// adresa IPv4 (unde se pot folosi functiile inet_ntoa() si inet_aton()
// pentru a converti intr-un/dintr-un sir de caractere)
struct in_addr { uint32_t s_addr };

// adresa de socket
struct sockaddr {
    unsigned char sa_family; // AF_INET pentru IPv4, AF_INET6 pentru IPv6
    char sa_data[14];
};

// adresa de socket in Internet
struct sockaddr_in {
    unsigned short sin_family;    // AF_INET pentru IPv4, AF_INET6 pentru IPv6
    unsigned short int sin_port;  // port (trebuie convertit in network/host order)
    struct in_addr sin_addr;      // adresa IP
};
```

- Pasi pentru comunicatie:

Client	Server
`// se creeaza socket-ul care va fi folosit la comunicatie` `int sock = socket(AF_INET, SOCK_DGRAM, 0);`	

`// se completeaza adresa si portul`
`// serverului la care ne conectam`
`struct sockaddr_in addr;`
`addr.sin_family = AF_INET;`
`addr.sin_port = htons(12345);`
`inet_aton("127.0.0.1", &addr.sin_addr);`
	`// se completeaza informatii despre`
	`// adresa si portul pe care ascultam`
	`struct sockaddr_in addr;`
	`addr.sin_family = AF_INET;`
	`addr.sin_port = htons(12345);`
	`addr.sin_addr.s_addr = INADDR_ANY;`

-	`// se asociaza socketul cu portul ales`
	`int b = bind(sock, (struct sockaddr*)`
	`&addr, sizeof(addr));`

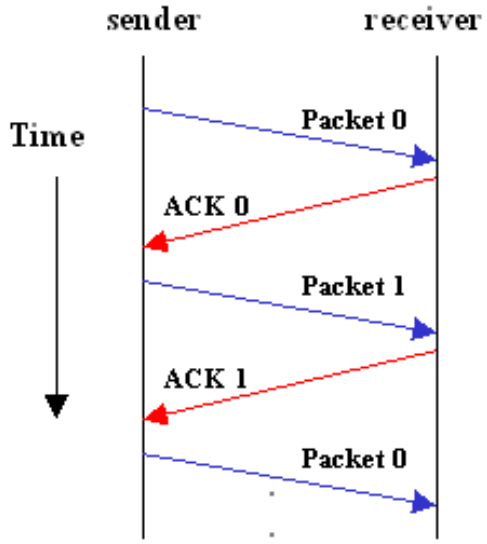
`// se trimit date catre server`
`char buf[100];`
`int s = sendto(sock, buf, 100, 0,`
`(struct sockaddr*) &addr, sizeof(addr));`
	`// se primesc date de la client`
	`char buf[100];`
	`struct sockaddr_in cli_addr;`
	`socklen_t socklen = sizeof(cli_addr);`
	`int r = recvfrom(sock, buf, 100, 0,`
	`(struct sockaddr*) &cli_addr, &socklen);`

`// se pot primi date de la server`
`// folosind functia recvfrom() (apelata`
`// la fel ca la server)`
	`// se pot trimite date la client`
	`// folosind functia sendto() (apelata`
	`// la fel ca la client)`

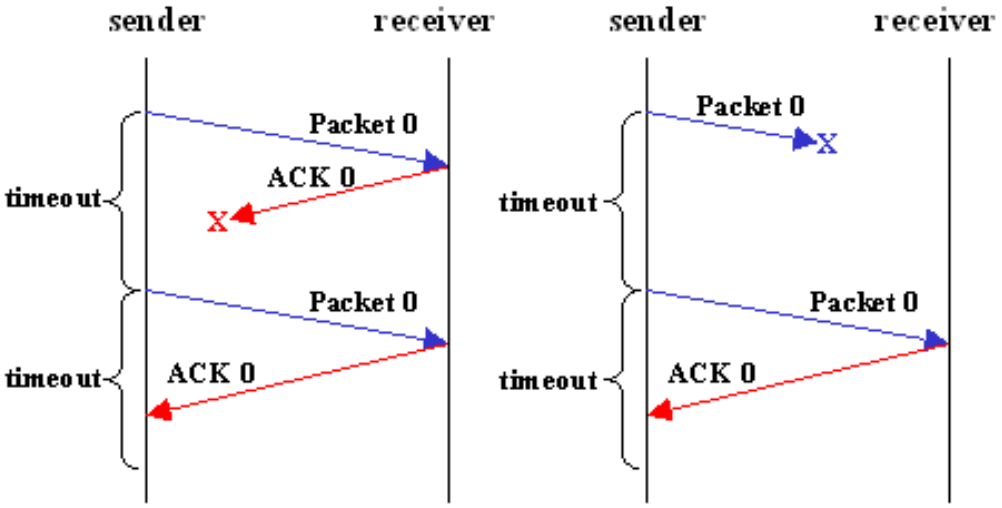
`// se inchide socket-ul` `close(sock);`	

Stop and wait

- Se trimite un pachet si apoi se asteapta confirmare (acknowledgment - **ACK**)



- Ce se intampla daca un pachet se pierde?
- Dar daca un ACK se pierde?



Link-uri

[Lab OCW]
[RFC UDP (768)]
[Simulator]
[Formular feedback]
[C Crash Course]
[Guide to Network Programming]