

# **PROIECT RC-P**

## **APLICAȚIE PENTRU TRANSFERUL DE FIȘIERE FOLOSIND PROTOCOLUL CU FEREASTRĂ GLISANTĂ**

### **Studenti**

Constantin - Cosmin Cojocaru  
Radu - Andrei Budeanu

### **Profesor coordonator**

Nicolae - Alexandru Botezatu

# Cuprins

	Pagina
1. Context.....	3
2. Protocolul cu fereastra glisanta (Sliding window protocol).....	4
2.1. Descrierea protocolului.....	4
2.2. Implementarea prin metoda Go Back N.....	5
2.3. Implementarea prin metoda selectării repetitive (Selective Repeat).....	6
3. Descrierea aplicației.....	7
3.1. Metoda de implementare a protocolului cu fereastră glisantă.....	7
3.2. Standardul folosit pentru conexiunea dintre noduri.....	7
3.3. Limbajul de programare și bibliotecile aferente.....	7
3.4. Structura pachetelor.....	7
3.5. Structura aplicației.....	10
3.5.1. Interfața aplicației.....	10
3.5.2. Funcționalitatea aplicației.....	11
Referințe.....	15

# 1. Context

## Cerințe

Sa se implementeze o aplicație pentru transferul de fișiere între două noduri de rețea folosind protocolul cu fereastra glisanta.

Constrângeri:

- Două view-uri: unul pentru transmisie, unul pentru recepție
- Două sau mai multe perechi de instanțe ale aplicației (tx-rx) trebuie să poată să ruleze pe aceleași mașini sau în același LAN
- Comunicația va fi implementată prin datagrame UDP
- Formatul pachetelor va fi stabilit de echipe
- View-ul pentru recepție să permită „pierderea” voită a unor pachete pentru a putea demonstra funcționarea mecanismului
- Posibilitatea de configurare a parametrilor de funcționare (dimensiune fereastră, temporizări...)
- Se va alege una dintre următoarele variante de implementare: Go back n, Selective repeat fără Nack, Selective repeat cu Nack, implementarea TCP pentru controlul fluxului

## 2. Protocolul cu fereastră glisantă (Sliding Window Protocol)

### 2.1. Descrierea protocolului

Protocolul cu fereastră glisantă este un protocol al nivelului legătură de date care facilitează transmiterea secvențială și mai eficientă a pachetelor de date de la un sender la un receiver. Acest protocol vine ca o îmbunătățire al protocolului Stop and Wait.

Principiul de funcționare al protocolului Stop and Wait: sender-ul trimite un pachet receiver-ului, urmând ca pachetul următor să fie transmis doar din momentul în care se primește mesajul de confirmare al primirii pachetului.

Îmbunătățirea adusă de protocolul cu fereastră glisantă este dat de crearea unui buffer de dimensiune  $N$  pachete la nivelul sender-ului (fereastră) care se trimit secvențial, secvențial urmând să se trimită și semnalele de confirmare.

Stop and Wait protocol

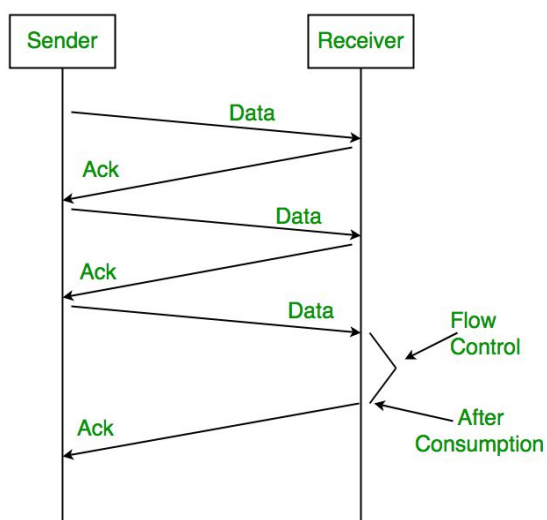


Fig 1.a

Sliding window protocol

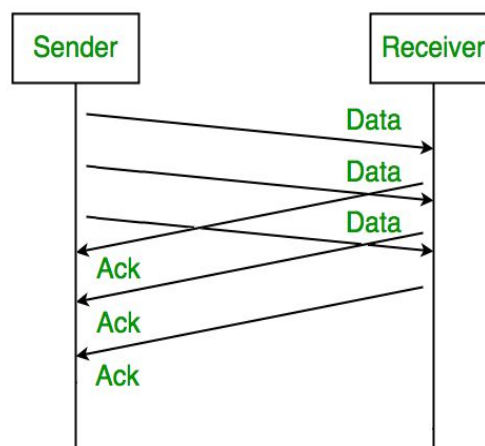


Fig. 1.b

Sursă imagini: GeeksforGeeks

Pentru implementarea acestui protocol există mai multe implementări și vor fi prezentate după cum urmează: Go Back N, Selective repeat fără Nack, Selective repeat cu Nack.

## 2.2. Implementarea prin metoda Go Back N

Metoda de implementare Go Back N este cea mai simplă metoda de implementare dintre cele enumerate.

La nivelul sender-ului este implementat un timer care reține timpul de la ultimul semnal de confirmare primit, iar fereastra acestuia este de mărime N pachete. Sender-ul va trimite secvențial pachetele din fereastra până la pachetul N.

La nivelul sender-ului avem două cazuri: primirea sau neprimirea semnalului de confirmare. La primirea semnalului de confirmare al primului pachet din fereastra, fereastra va face slide cu o poziție. Există însă posibilitatea ca primele k pachete să fie primite, dar pentru acestea se pierd semnalele de confirmare, însă sender-ul să primească semnalul de confirmare de la pachetul k + 1. În acest caz fereastra va face slide peste toate pozițiile pachetelor pentru care nu s-a primit confirmarea inclusiv peste poziția pachetului k + 1. Acest lucru se întâmplă deoarece primirea unui semnal de confirmare mai mare decât poziția actuală a ferestrei semnifică faptul că acele pachete k au fost recepționate de receiver. În cazul în care sender-ul nu primește semnal de confirmare după un anumit timp (timp de timeout), acesta va retrimite întreaga fereastră.

La nivelul receiver-ului avem un buffer de intrare de dimensiune 1. Receiver-ul trebuie să primească secvențial pachetele pentru a fi trimise mai departe către aplicație.

La nivelul receiver-ului avem două posibilități: primirea următorului pachet sau primirea unui alt pachet care este după pachetul care ar trebui să fie primit. La primirea pachetului, receiver-ul verifică dacă acesta este pachetul următor ultimului pachet primit. Dacă acesta nu este următorul pachet, atunci receiver-ul va respinge acest pachet, dar și pachetele care mai vin. După un timp de timeout, receiver-ul va primi pachetul pe care îl așteaptă, retrimis de sender.

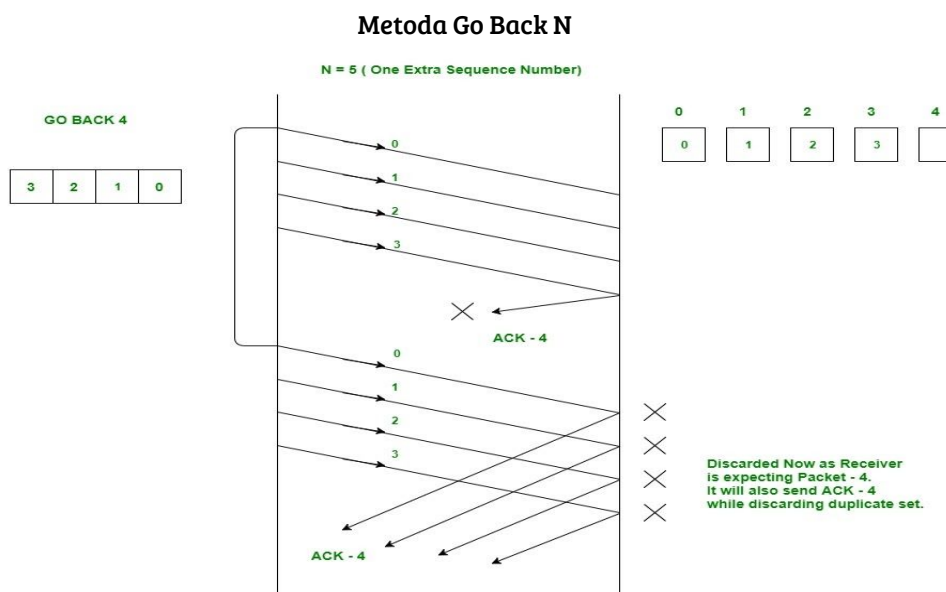


Fig. 2

Sursă imagini: GeeksforGeeks

### 2.3. Implementarea prin metoda selectării repetitive (Selective Repeat)

Dacă în timpul transmisiei pachetelor exista multe pierderi, algoritmul GO Back N devine foarte ineficient, utilizând prost lățimea de bandă. Pentru a se veni cu o îmbunătățire, s-a dezvoltat sistemul cu selecție repetitivă, care de data aceasta va retransmite doar pachetele pierdute, care nu ajung la destinație, sau va retransmite pachetele ale căror confirmări nu au ajuns înapoi la sursă.

Spre deosebire de varianta GO Back N, acest sistem este mai greu de implementat, deoarece trebuie adăugată o anumită logică suplimentară pentru cele două noduri. Pentru nodul de transmisie, se poate opta pentru una dintre cele două implementări: selective repeat cu NACK sau selective repeat fără NACK. Pentru varianta cu NACK, retransmiterea pachetelor va fi cerută de către receptor printr-un mesaj de tipul “Not ACKnowledged” atunci când pachetele sunt eronate sau când se primește un pachet cu ordinea mai mare decât următorul din fereastra de recepție, iar transmițătorul va trebui să fie în stare să proceseze aceste mesaje și să transmită respectivele pachete. În varianta fără NACK, transmițătorul va trebui să măsoare timpul scurs de la transmiterea unui pachet. Dacă acest timp depășește o valoare de timeout și transmițătorul nu primește un mesaj de tipul ACK (acknowledge - ACK), atunci pachetul respectiv se va retransmite. Pentru receptor, va trebui implementată, de asemenea, o fereastră care este de aceeași dimensiune cu cea a transmițătorului, care va ordona pachetele pentru a fi salvate în ordinea corespunzătoare. În același timp, nodul care primește pachetele va trebui să genereze câte un semnal de validare pentru fiecare pachet primit și lipsit de erori de transmisie (dacă s-au putut detecta).

Selective Repeat cu NACK

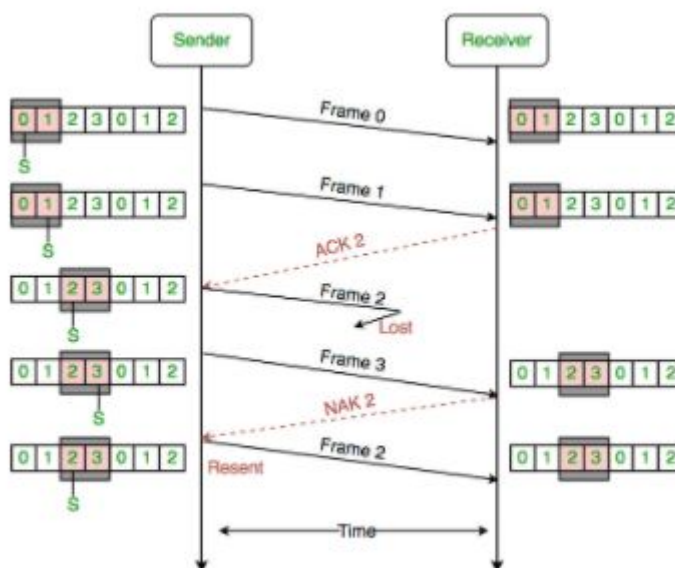


Fig. 3

Sursa imaginii: GeeksforGeeks

## 3. Descrierea aplicației

### 3.1. Metoda de implementare a protocolului cu fereastra glisanta

Pentru sistemul de transfer al fișierelor, se va implementa varianta cu selecție repetată (Selective Repeat) fără NACK. Asta înseamnă ca transmițătorul va avea cate un timer pentru fiecare pachet trimis din fereastra curentă, iar dacă acest timer va depăși valoarea de timeout stabilita pana la primirea unui semnal de validare, va trebui să se retransmisa respectivul pachet.

### 3.2. Standardul folosit pentru conexiunea dintre noduri și formatul pachetelor

Aplicația va realiza conexiunea între sender si receiver folosind standardul UDP (User Datagram Protocol). Acest standard permite transferul între noduri fără a se stabili o “înțelegere” în prealabil (three hand shake-ul).

### 3.3. Limbajul de programare și bibliotecile aferente

Pentru implementarea aplicației ce transfera fișierele, s-a folosit limbajul Python, utilizând doar biblioteca *sockets* (<https://docs.python.org/3/library/socket.html>) pentru conectarea la stiva de comunicație și respectiv între cele două noduri care comunică. Pentru suportul de rulare pseudo paralelă s-a folosit biblioteca *threading* (<https://docs.python.org/3/library/threading.html>). Pentru comunicarea între firele de execuție de la nivelul sender-ului s-a folosit biblioteca *Queue*. Pentru determinarea dimensiunii fișierului s-a folosit biblioteca *pathlib*. Pentru implementarea view-urilor s-a folosit biblioteca *PyQt5*. Pentru determinarea datei și orei curente se va folosi biblioteca *datetime*.

Alte biblioteci folosite: sys, os, time, random.

Pentru implementarea aplicației s-a folosit paradigma de programare orientată pe obiecte.

### 3.4. Structura pachetelor

Sender-ul și receiver-ul gestionează 4 tipuri de pachete:

#### 1) Pachet de tip DATA:

Aceste pachete conțin datele efective ale fișierului. Dimensiunea întregului pachet este cuprinsa între 68 de octeți și 64 de kiloocteți - 1 ((2 la puterea 16) - 1). Acesta este împărțit după cum urmează:

- Primul octet reprezinta tipul pachetului. Pentru pachetul de date, acesta are valoarea 0x1.
- Următorii 3 octeți reprezinta numărul unic de ordine al pachetului (pot fi maxim (2 la puterea 24) - 1 pachete de orice tip în total pentru un fișier).
- Următorii 64 - 64k octeți - 4 (octeții pentru tip și număr) - 1 reprezintă datele efective ale fișierului transmis dintr-un pachet. Aceasta valoare este aleasă de către utilizator din interfața grafica.

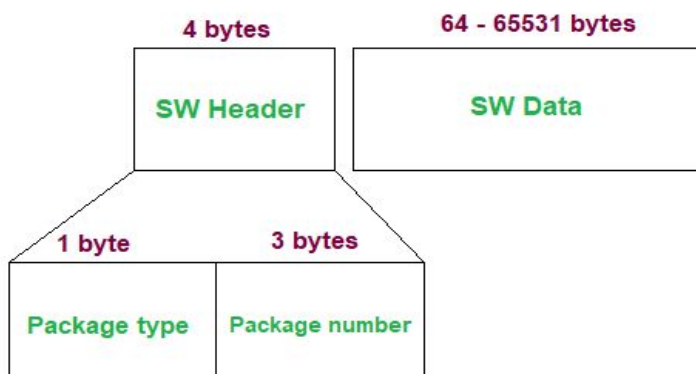


Fig .4 - Formatul pachetelor de date

## 2) Pachet de tip INIT (inițializare):

Exista un singur pachet de acest tip și este transmis primul de către sender. Acesta conține informații importante pentru receiver pentru a cunoaște modul de gestionare a pachetelor de date pe care urmează să le primească. Pachetul de inițializare are dimensiunea de 64 de octeți și este împărțit după cum urmează:

- Primul octet reprezintă tipul pachetului. Pentru pachetul de inițializare acesta are valoarea 0x0.
- Următorii 3 octeți reprezintă numărul unic de ordine al pachetului (pot fi maxim (2 la puterea 24) - 1 pachete de orice tip în total pentru un fișier).
- Următorii 3 octeți reprezintă numărul de pachete care vor fi transmise de către sender.
- Următorii 2 octeți reprezintă dimensiunea pachetelor de date care urmează să fie trimise. Maximul dimensiunii pachetului (64k octeți - 1).
- Următorul octet reprezintă dimensiunea ferestrei.
- Următorii 54 de octeți (restul până la 64) sunt destinați numelui și extensiei fișierului. Numele fișierelor care depășesc această dimensiune vor fi trunchiate.



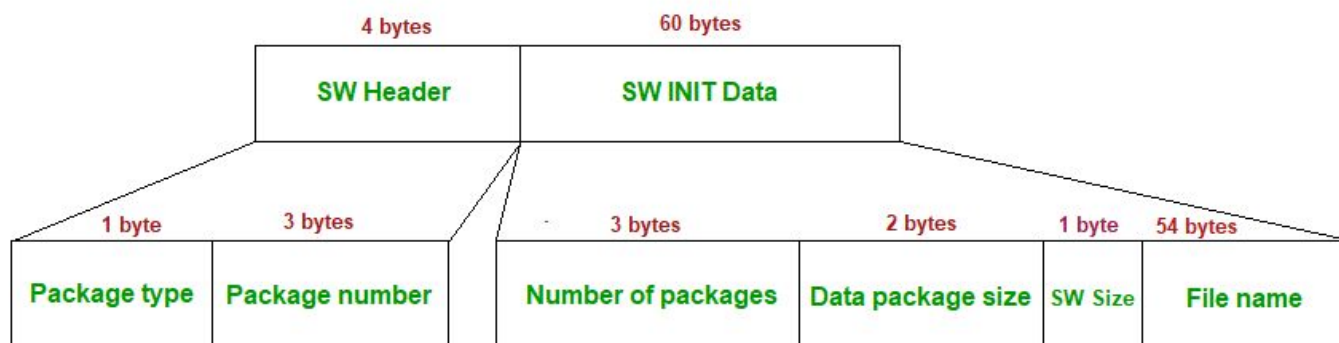


Fig .5 - Formatul pachetelor de inițializare

### 3) Pachetul de tip ACK:

Acest pachet este creat la nivelul receiver-ului și transmis către sender în momentul primirii unui pachet de tip INIT sau DATE. Pachetul de tip ACK (acknowledge) are dimensiunea de 4 octeți și este structurat după cum urmează:

- Primul octet reprezintă tipul pachetului. Pentru pachetul de tip acknowledge acesta are valoarea 0x2.
- Următorii 3 octeți reprezintă numărul de ordine al pachetului care tocmai a fost primit.



Fig .6 - Formatul pachetelor de acknowledge

### 4) Pachetul de tip CHECK:

Acest pachet este creat la nivelul sender-ului, trimis către receiver și înapoiat imediat la primire. Acest mecanism este folosit pentru verificarea disponibilității transmiției de date la o combinație de IP și port. Pachetul de tip CHECK (acknowledge) are dimensiunea de 4 octeți și este structurat după cum urmează:

- Primul octet reprezintă tipul pachetului. Pentru pachetul de tip acknowledge acesta are valoarea 0x3.
- Următorii 3 octeți sunt redundanți.

### 5) Pachetul de final:

Acesta este primit de către receiver ca un ultim pachet trimis de către sender care să confirme că transmisia fișierului s-a încheiat. Acesta mai este primit de către receiver la închiderea forțată a sender-ului care să confirme faptul că sender-ul s-a oprit. Acesta mai este primit și în momentul în care aplicația este oprită de către utilizator. Pachetul de final are dimensiunea de 4 octeți și este structurat după cum urmează:

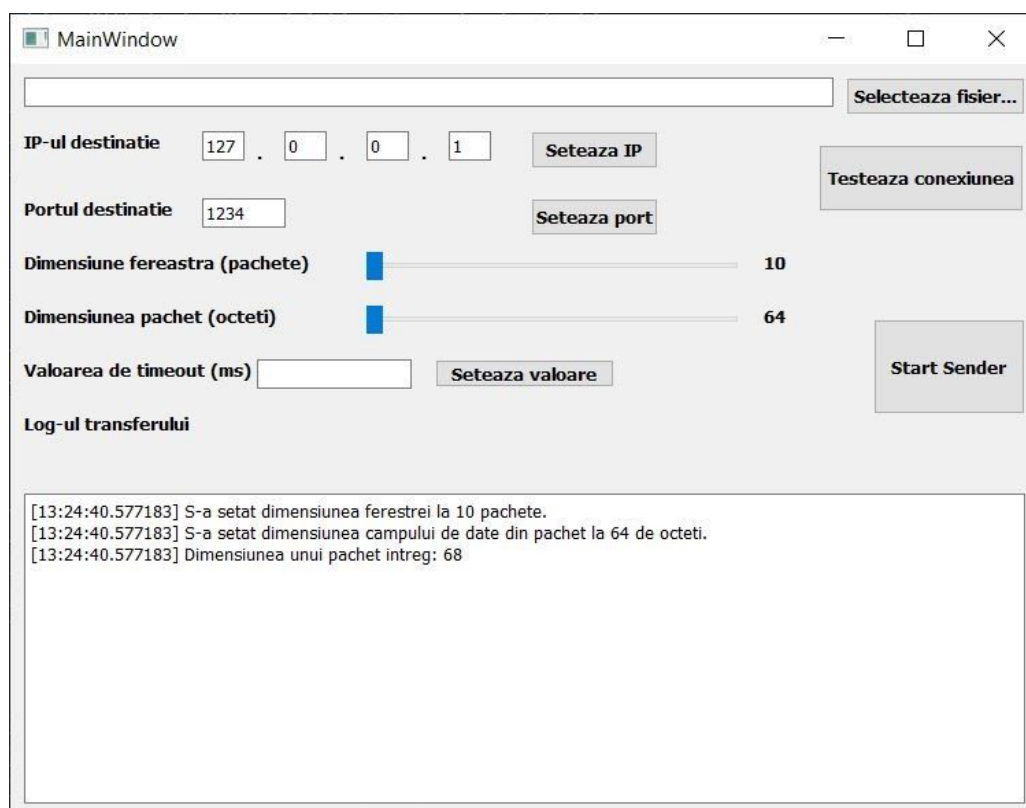
- Primul octet reprezintă tipul pachetului. Pentru pachetul de final acesta are valoarea 0xFF.
- Următorii 3 octeți este numărul pachetului cu valoarea cea mai mare a numărului de ordine: 0xFFFFF.

## 3.5. Structura aplicației

### 3.5.1. Interfața aplicației

La nivelul sender-ului se permite selectarea unui fișier pentru a putea fi trimis, cât și o serie de parametri ce țin de funcționarea protocolului cu fereastra glisantă, cum ar fi:

- IP-ul și portul receiver-ului
- dimensiunea ferestrei în număr de pachete
- dimensiunea câmpului de date din interiorul unui pachet în octeți
- valoarea de timeout pentru care se așteaptă mesajul de ACK (în milisecunde)



MainWindow

Selectează fișier...

IP-ul destinație: 127 . 0 . 0 . 1    Setează IP

Portul destinație: 1234    Setează port

Dimensiune fereastră (pachete): 10

Dimensiunea pachet (octeți): 64

Valoarea de timeout (ms):    Setează valoare

Testează conexiunea

Start Sender

Log-ul transferului

```
[13:24:40.577183] S-a setat dimensiunea ferestrei la 10 pachete.
[13:24:40.577183] S-a setat dimensiunea câmpului de date din pachet la 64 de octeți.
[13:24:40.577183] Dimensiunea unui pachet întreg: 68
```

Fig. 7 - Fereastra principală pentru instanța senderului

De asemenea, pentru a tine evidenta tuturor evenimentelor și pentru o depanare mai ușoară, s-a implementat și un sistem de logging care afișează într-un chenar de tip text momentul în care s-a produs un eveniment și tipul acestuia.

La nivelul receiver-ului se permite stabilirea următorilor parametrii:

- IP-ul (fie adresa de loopback, fie cea din LAN) și portul la care se vor aștepta pachete de la sender
- probabilitatea de pierdere a pachetelor (în procente)

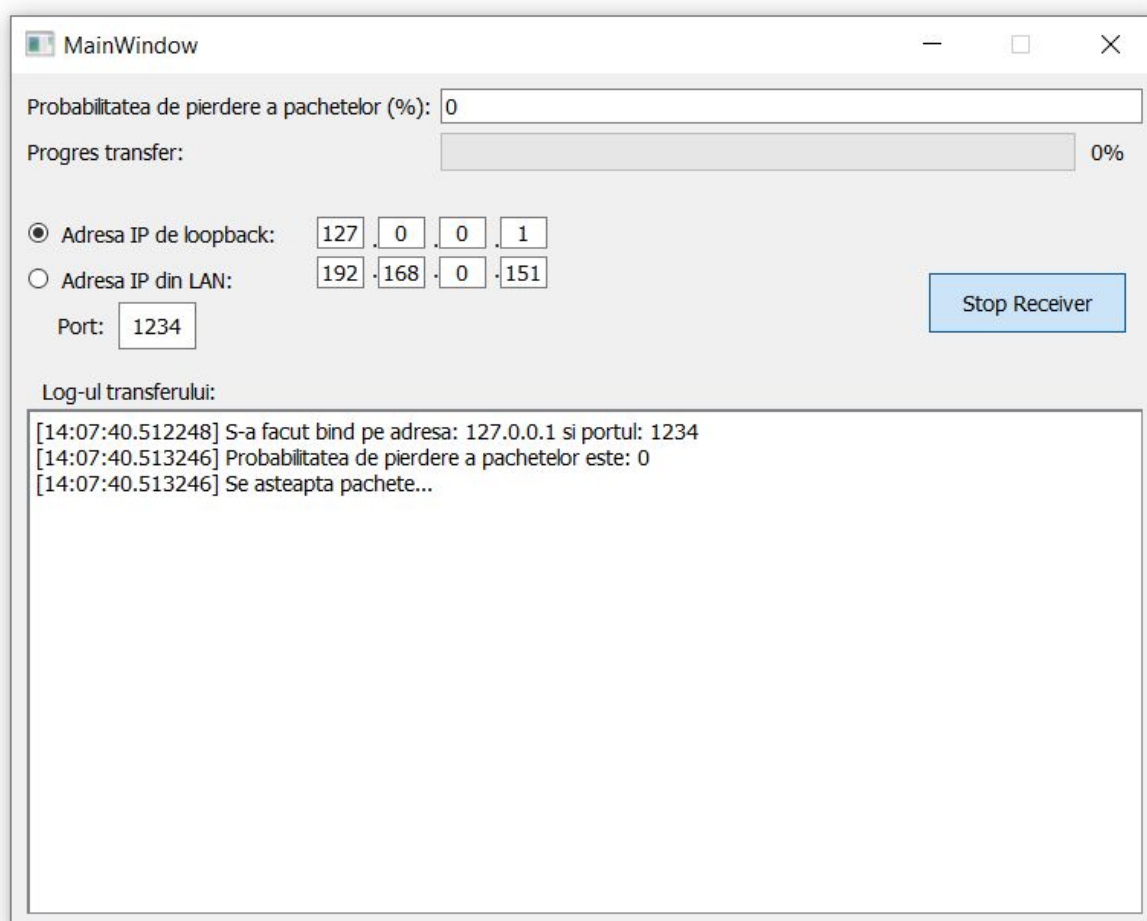


Fig. 8 - Fereastra principală pentru instanța senderului

Pentru a tine evidenta evenimentelor produse la nivelul receiver-ului s-a implementat un log în care se vor scrie evenimentele produse, timpul producerii acestora, cat și detalii despre acestea.

### 3.5.2. Funcționalitatea aplicației

Sender-ul ia fișierul selectat, îl citește și creează secvențial pachetele conform unui standard definit (Fig. 4, 5, 6), apoi le trimite către stiva de comunicație prin intermediu UDP, setând și timer-ul pentru verificarea timeout-ului, în același timp. În același timp, sender-ul este capabil să proceseze la

rândul lui mesaje de tip ACK în care va primi confirmarea ca un pachet a ajuns cu succes la destinație. Ordinea de transmisie a pachetelor va fi determinată de o fereastră de lungime N, care prezintă funcționalitatea descrisă mai sus în documentație, specifică pentru protocolul cu fereastră glisantă.

La rularea receiver-ului thread-ul principal se ocupă de afișarea interfeței grafice, cât și de verificarea datelor introduse în câmpurile de tip text.

La apăsarea butonului “Start Receiver”, se pornește un thread în care receiver-ul începe procedura de gestionare a pachetelor. Din acest moment receiver-ul așteaptă pachete. Inițial receiver-ul așteaptă pachetul de tip INIT de la sender pentru a cunoaște mai departe cum să gestioneze pachetele de date. Dacă pachetul de tip INIT este pierdut, toate celelalte pachete care vor ajunge vor fi aruncate. Odată ce pachetul de tip INIT este primit și se cunoaște dimensiunea pachetelor de date, numele fișierului și dimensiunea ferestrei, în continuare se vor gestiona pachetele de date.

Toate pachetele care s-au primit și care nu sunt aruncate din cauza probabilității de pierdere, sunt trimise către fereastră. Dacă primul pachet corespunde celui pe care îl așteptăm, atunci datele din acesta este scris instant în fișier, iar fereastră face slide, altfel pachetul este cache-uit în fereastră. Dacă pachetul următor este primit, urmează să se verifice secvențial dacă mai sunt pachete cache-uite în fereastră și urmează să fie scrise până când nu se mai găsește următorul pachet care ar urma să fie scris. La finalul execuției receiver-ului se va mai aștepta o perioadă de 10 secunde pentru a se putea trimite mesaje de acknowledged pachetelor pentru care confirmarea a fost pierdută.

La nivelul receiver-ului este implementat un timer de 10 secunde pentru a opri receiver-ul în cazul în care nu se primesc pachete în acest interval.

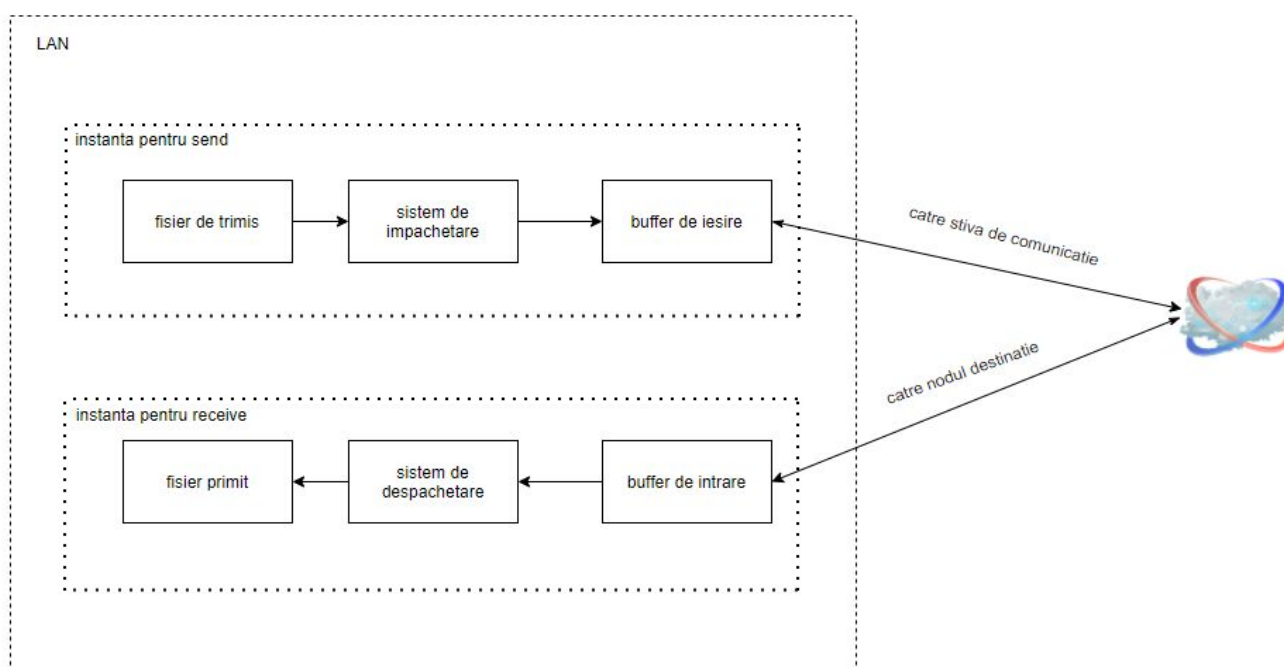


Fig. 9 - Schema principală care descrie conexiunea dintre cele două noduri

În partea senderului, la rularea aplicației thread-ul principal se ocupă cu afișarea interfeței grafice și gestionarea evenimentelor aferente acesteia. Atunci când se trimite un fișier, aplicația va lansa în execuție un thread care va împărți fișierul în pachete și va porni alte două thread-uri, după care va pune într-un buffer pachetele generate. Celelalte două thread-uri se ocupă cu trimiterea pachetelor către destinație, respectiv primirea mesaje de ACK pentru a valida transmisia completă a unui pachet și mutarea ferestrei cu o poziție mai departe. Astfel, s-a evitat o execuție secvențială a acestor pași, pentru o procesare mai eficientă în timp, deoarece un thread deschide fișierul, îl împarte în pachete și trimite pachetele într-un buffer, un thread ia pachetele din buffer și le trimite către destinație prin intermediul socket-ului, iar un thread așteaptă mesaje de ACK și face slide la fereastra.

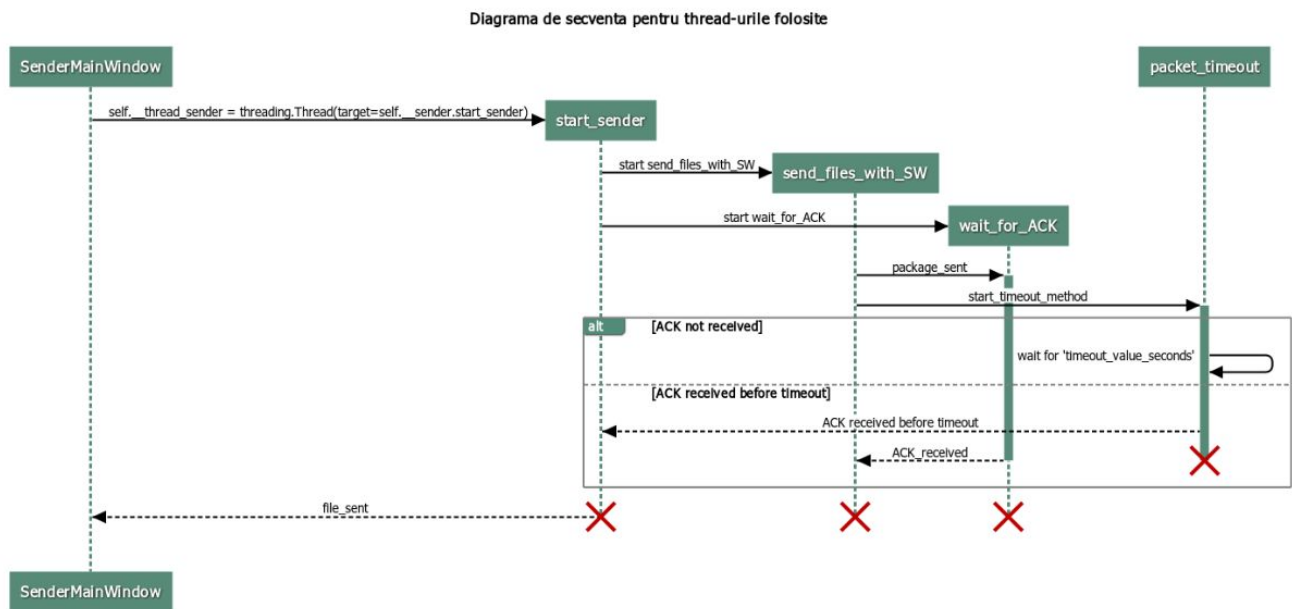


Fig. 10 - Diagrama de secvență a thread-urilor sender-ului

Thread-ul care trimite pachetele și thread-ul care primește mesajele de ACK comunică între ele prin intermediul unor variabile comune dar și prin intermediul unui sistem de “cache” al pachetelor. Practic, metoda **send\_file\_with\_SW** verifica la fiecare iterație dacă următorul pachet de transmis este încadrat în fereastra curentă. Dacă pachetul se afla în fereastra, atunci acesta trimite fișierul către receiver prin intermediul protocolului UDP, adaugă pachetul într-un dicționar de pachete iar apoi pornește un thread separat setându-i o valoare de timeout pentru a putea ști mai târziu dacă acel pachet a fost validat sau nu de către receiver. Dacă pachetul nu se afla în fereastra, atunci thread-ul rămâne blocat în structura while până când metoda **wait\_for\_ACK** va primi confirmările pentru pachetele din fereastra curentă și va face slide.

Pentru mesajele de ACK, s-a definit un dicționar menit să retina pachetele care au fost trimise recent și care așteaptă confirmarea. Dacă mesajul de ACK este primit până la expirarea timeout-ului, atunci pachetul respectiv este șters din dicționar. Dacă încă nu a ajuns confirmarea pachetului și valoarea de timeout a expirat, atunci metoda **packet\_timeout** se va ocupa cu retransmiterea acelui pachet.

## Diagrame UML

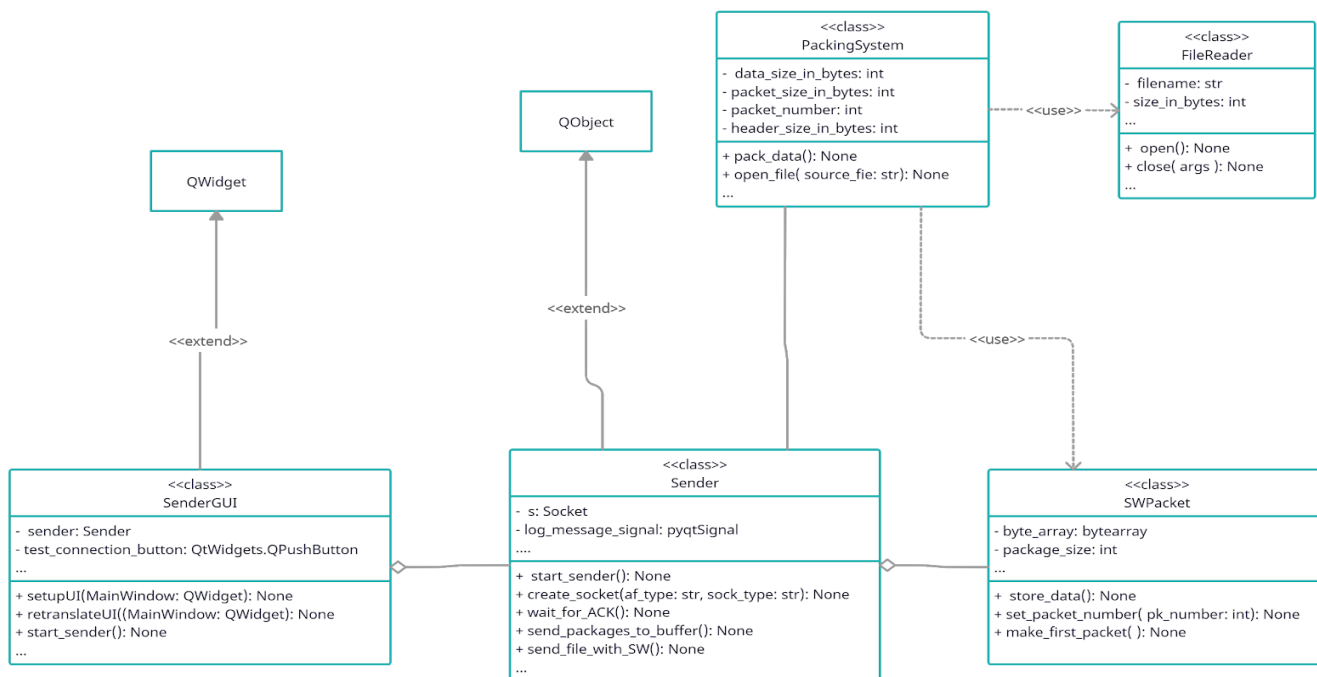


Fig.11 - Diagrama UML pentru instanța senderului

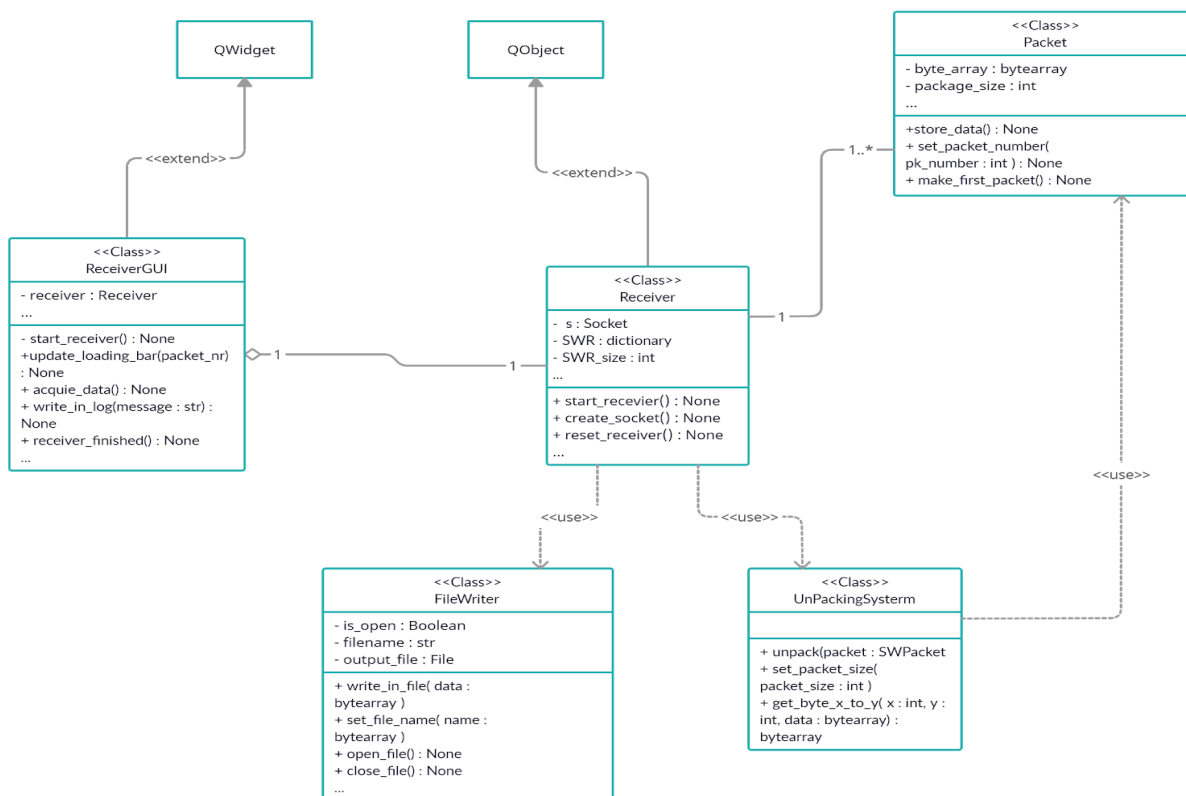


Fig.12 - Diagrama UML pentru instanța receiver-ului

**Referințe:**

- [1] <https://www.geeksforgeeks.org/sliding-window-protocol-set-1/>
- [2] <https://www.geeksforgeeks.org/sliding-window-protocol-set-2-receiver-side/>
- [3] <https://origin.geeksforgeeks.org/sliding-window-protocol-set-3-selective-repeat/>
- [4] <https://www.tutorialspoint.com/sliding-window-protocol>
- [5] <https://www.javatpoint.com/sliding-window-protocol>
- [6] Computer Networks 4th Edition by Andrew S. Tanenbaum