UNIVERSITATEA TEHNICĂ "GH ASACHI" IAŞI
FACULTATEA AUTOMATICĂ ŞI CALCULATOARE
SPECIALIZAREA CALCULATOARE ŞI TEHNOLOGIA INFORMAŢIEI

DISCIPLINA RETELE DE CALCULATOARE PROIECT

Aplicatie pentru transfer de fisiere – implementare printr-un mecanism de control al congestiei (similar TCP)

Coordonator,
Prof. Nicolae-Alexandru Botezatu

Studenti,
Mălăncuș Horia (1309A)
Susanu Alexandru-Cătălin(1309B)

Iaşi, 2020

Protocolul TCP:

TCP (Transmission Control Protocol) este un protocol folosit de aplicații care au nevoie de confirmare de primire a datelor. Efectuează o conectare virtual full duplex între doua puncte terminale, fiecare punct fiind definit de către o adresa IP și de către un port TCP. În special, TCP oferă încredere, asigura livrarea ordonata a unui flux de octeți de la un program de pe un computer la alt program de pe un alt computer aflat în rețea. Pe lângă sarcinile sale de gestionare a traficului, TCP controlează mărimea segmentului de date, debitul de informație, rata la care se face schimbul de date, precum și evitarea congestionării traficului de rețea.

Spre deosebire de TCP, UDP reprezintă tot un protocol de comunicație si care presupune o comunicație fără conexiune. Practic, UDP este un protocol ce nu oferă siguranța sosirii datelor la destinație (nu dispune de mecanisme de confirmare); totodată nu dispune nici de mecanisme de verificare a ordinii de sosire a datagramelor sau a datagramelor duplicate. Avantajul pe care îl prezintă protocolul UDP este viteza de transmitere a pachetelor, întrucât nu este necesară confirmarea recepției acestora.

Descrierea generala a algoritmului:

Algoritmul TCP de evitare a congestiei (TCP TAHOE) constă în 2 faze și anume: faza de "slow start" și faza de evitare a congestiei. În primul rând, faza slow start constă într-o strategie prin care algoritmul evită trimiterea mai multor date decat lățimea de bandă a rețelei poate transmite, evitând în acest fel, congestia. Cu alte cuvinte, specific algoritmului de evitare a congestiei este prezența ferestrei de congestie, care în cadrul fazei de "slow start" pornește de la o dimensiune standard (1/2/4/10 MSS), aceasta dublându-se după o perioadă de timp egală cu RTT (Round Trip Time). În felul acesta, fereastra de congestie își va mări exponențial dimensiunea până când se va atinge o valoare de prag. În al doilea rând, faza de evitare a congestiei intervine ulterior celei de "slow start", moment în care creșterea ferestrei de congestie nu se mai realizează exponențial, ci liniar, fiind incrementată cu 1 MSS, dupa fiecare RTT. Această creștere liniară are loc până la pierderea unui pachet, acest lucru fiind constatat atunci când confirmarea pachetului nu s-a realizat într-un timp prestabilit (timeout).

Gestionarea pierderii unui pachet are loc în felul următor: valoarea dimensiunii ferestrei de congestie va fi resetată la valoarea sa inițială, iar valoarea de prag, până la care are loc creșterea exponențială, va fi redusă la jumătate din valoarea dimensiunii ferestrei de congestie de dinainte de pierderea pachetului, faza de "slow start" fiind din nou inițializată.

O metodă îmbunătățită de detectare a pierderii unui pachet este cea de "fast retransmit" care constă în retrimiterea pachetului în momentul în care sender-ul primește 3 mesaje de confirmare identice. Astfel, procesul de transmitere al pachetelor este optimizat, nemaifiind necesară așteptarea timeout-ului pentru detectarea pierderii pachetelor.

Interfața grafica a aplicatiei:

Interfata aplicatiei consta in 2 view-uri, unul pentru sender si unul pentru receiver, asadar trebuie deschise 2 instante ale aplicatiei. Fiecare view contine o bară de meniu prin intermediul căreia se va putea schimba view-ul, dar si un meniu pentru setari.

View-ul sender-ului(clientului) conține:

- 2 entry-uri care permit introducerea IP-ului si a portului receiver-ului
- un buton care permite selectarea fișierului de transmis
- un buton care după introducerea datelor necesare conectării va efectua conexiunea
- 2 label-uri, unul pentru dimensiunea curenta a ferestrei de congestie(cwnd) si unul

pentru valoarea de prag(sstresh)

- o fereastră în care se pot afișa diverse mesaje în legătură cu conexiunea sau transmiterea pachetelor.

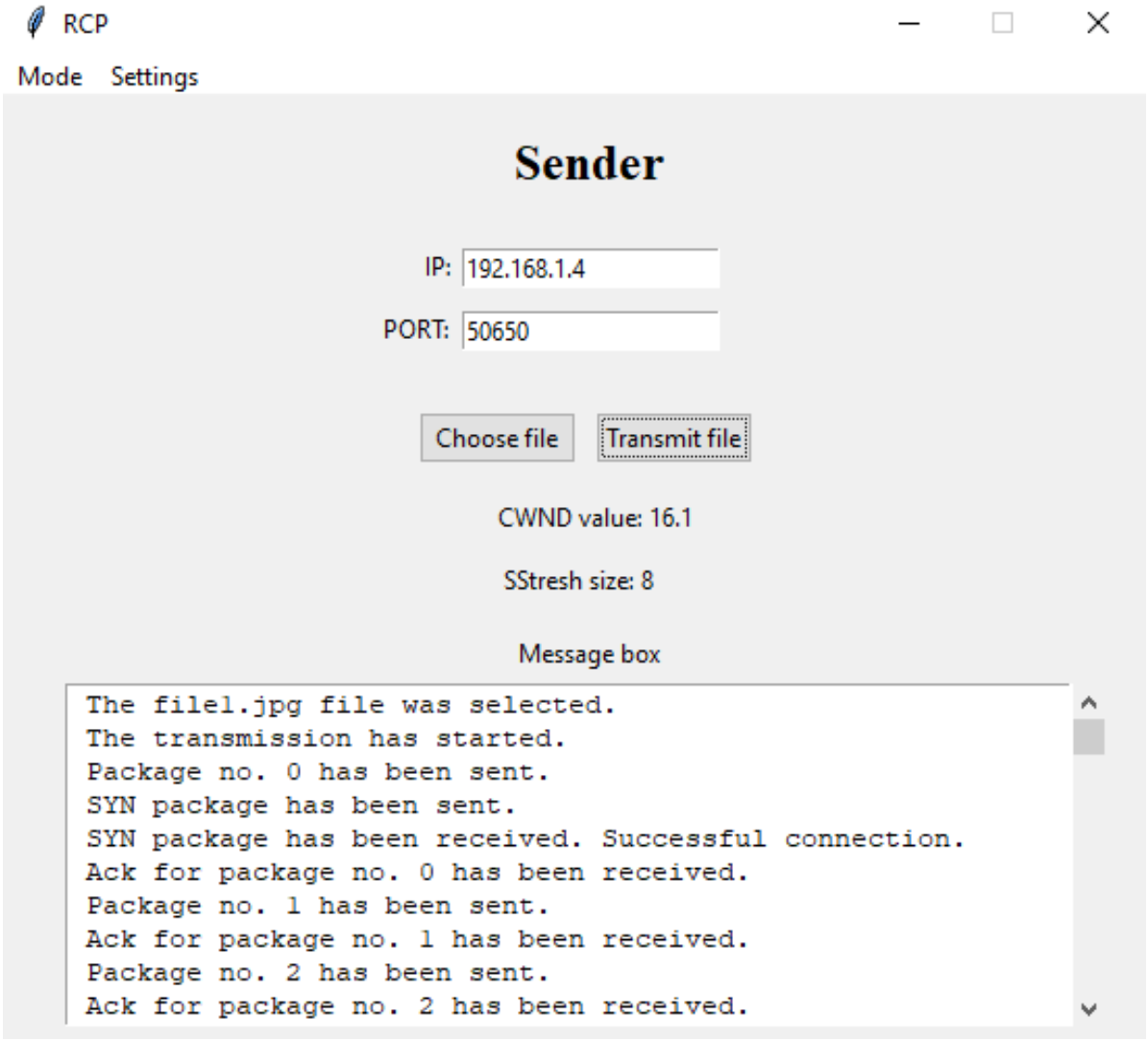


Figura 1- View-ul de transmisie

View-ul receiver-ului(serverului) conține:

- un buton ce va duce la crearea serverului
- un label ce indica ce procentaj din fisier a fost receptat
- o fereastră în care se vor putea afișa diverse mesaje în legătură cu conexiunea sau recepția pachetelor

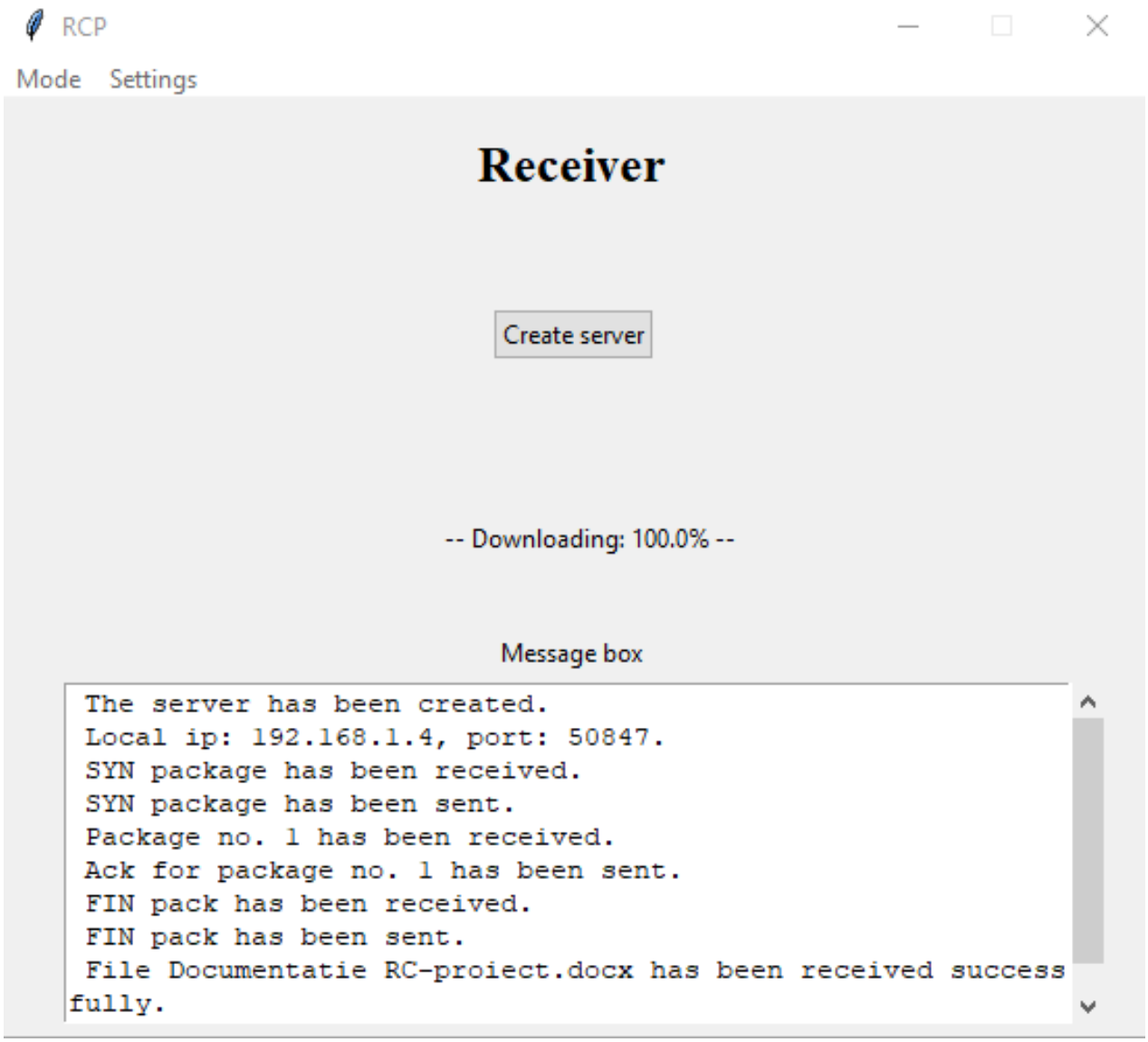


Figura 2- View-ul de receptie in urma primirii unui fisier

Meniul de setari permite:

- selectarea dimensiunii initiale a ferestrei de congestie(cwnd)
- selectarea valorii initiale a pragului(sstresh)
- selectarea dimensiunii unui pachet privind numarul de octeti
- selectarea procentului de pachete carora li se va face drop(0-50%)

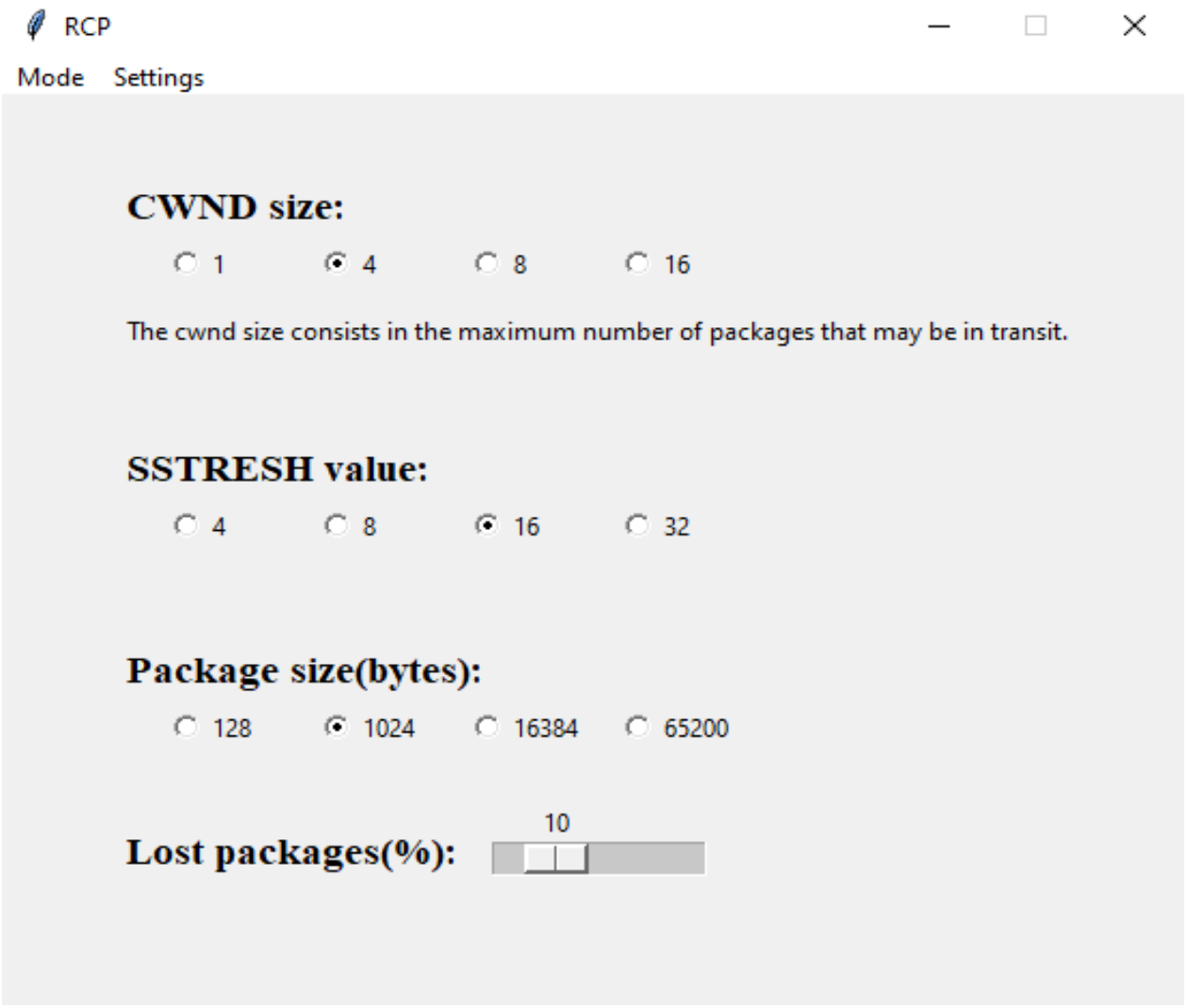


Figura 3- Meniul de setari

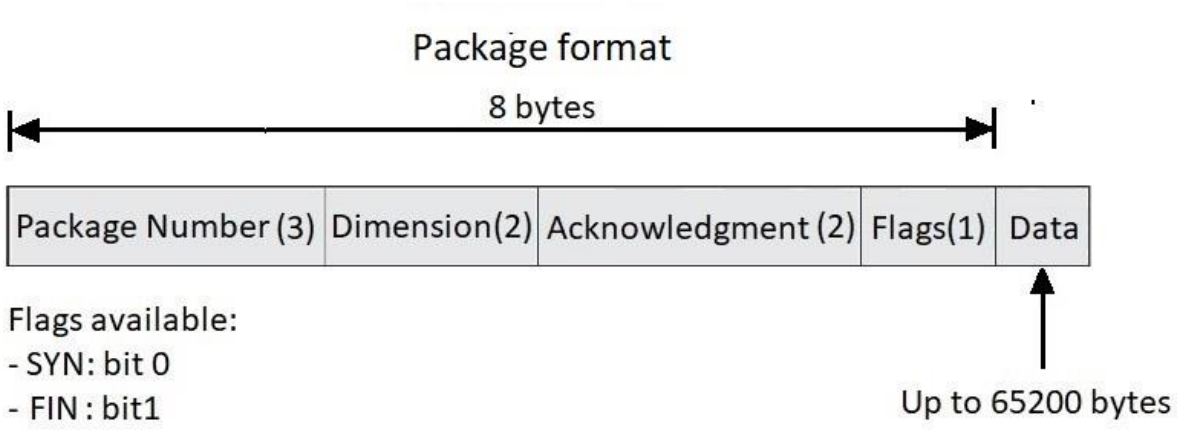
Accesarea meniului de setari este optionala, campurile avand valori prestabilite. In cazul in care se doreste modificarea valorilor campurilor este suficienta modificarea doar pentru view-ul specific sender-ului intrucat acesta ii va transmite receiver-ului prin pachetul SYN atat dimensiunea pachetului, cat si procentul de pachete la care se renunta(dimensiunea ferestrei si valoarea de prag nu sunt transmise, fiind specifice sender-ului).

Formatul pachetelor

Structura unui pachet este urmatoarea:

- 3 octeti pentru numarul pachetului
- 2 octeti pentru dimensiunea pachetului
- 2 octeti pentru acknowledgment
- 1 octet pentru flag-uri: SYN(bitul 0) si FIN(bitul 1)
- un numar variabil de octeti de date ce va putea fi selectat din meniul de setari(128/1024/16384/65200).

Pachetele vor fi transmise dupa urmatorul principiu: sender-ul va seta fiecarui pachet un numar unic de identificare, dar si campul dimensiune care va fi completat cu numar de octeti de date transmisi. Receiver-ul la randul sau trebuie sa trimita inapoi un pachet care va avea setat acelasi numar de identificare, dar si campul ack completat cu valoarea campului dimensiune a pachetului pentru care se trimite ack-ul.



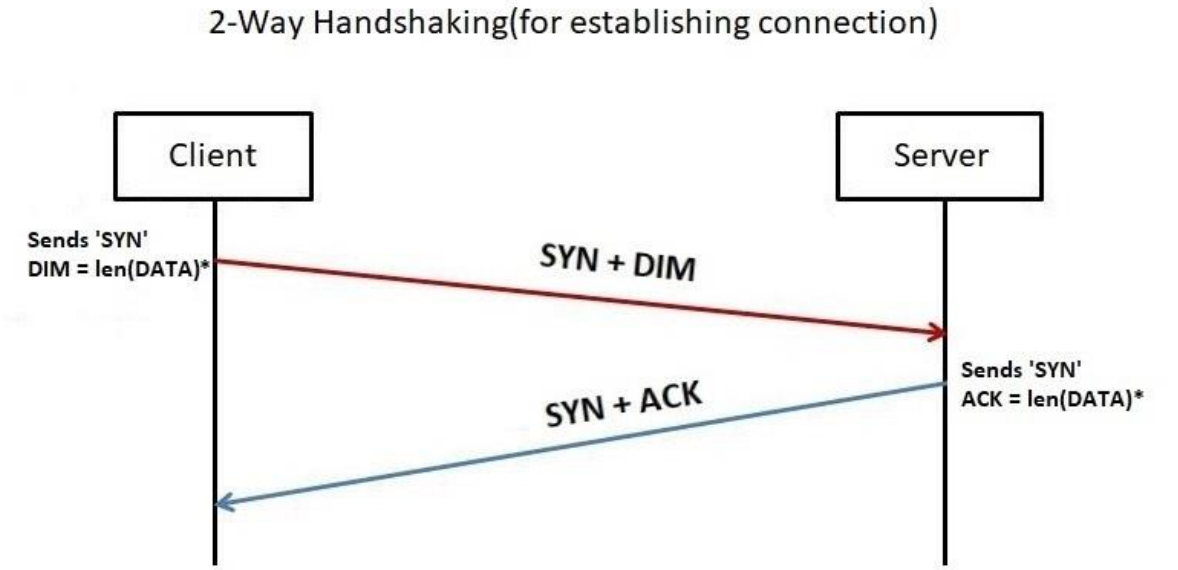


Figura 3- Formatul pachetelor si initializarea conexiunii

Pachetul SYN, de initializare a conexiunii, pe care sender-ul il trimite serverului va avea setat flag-ul SYN si va contine in campul de date denumirea fisierului, numarul total de pachete ce trebuie transmise, dimensiunea unui pachet si procentajul de pachete la care se va renunta. De asemenea, campul dimensiune va fi completat cu lungimea campului de date. La randul sau, serverul va trimite catre client un pachet care va avea setat flag-ul SYN, iar campul ack va avea valoarea egala cu campul dimensiune a pachetului SYN trimis de catre client.

Conexiunea se va incheia de asemenea printr-un proces de tip 2-way handshake, in ideea ca pachetul trimis de catre client va avea setat numarul de pachet, dar si flag-ul FIN, campul de date ramanand gol, iar server-ul va trimite la randul lui un pachet ce avea avea setat acelasi numar de pachet, dar si flag-ul FIN.

Detalii de implementare

Componentele principale ale aplicatiei sunt reprezentate de clasele:

- Server
- ServerPackagesHandler
- Client
- ClientPackagesHandler

Serverul este creat prin ’bind-ul’ unui obiect de tip socket la ip-ul interfetei de retea si la un port disponibil care este generat aleatoriu de catre sistemul de operare. Socket-ul folosit este setat ca fiind neblocant pentru a nu fi constransi in a receptiona secvential pachetele. De asemenea, server-ul are ca si subcomponenta un handler(ServerPackagesHandler) care il va ajuta la stocarea temporara a pachetelor ce nu sosesc in ordine. Server-ul mai contine un counter ce indica numarul pachetului care urmeaza a fi scris in fisier, pachetele a caror numar este mai mare decat counter-ul server-ului fiind oferite handler-ului pentru stocare temporara.

Serverul propriu-zis consta intr-un thread ce ruleaza intr-o bucla infinita atata timp cat mai sunt pachete de receptat. De fiecare data cand se primesc date pe socket se creeaza un pachet pe baza acestora pentru a putea fi interpretate. Daca lungimea campului de date nu coincide cu campul dimensiune setat atunci pachetului i se face automat ’drop’. Primul pachet asteptat de catre server este pachetul SYN care va avea setate numarul de pachet, dimensiunea pachetului si flag-ul SYN, iar in campul de date se vor regasi informatii precum numele fisierului transmis, numarul total de pachete ce urmeaza a fi transmise, dimensiunea unui pachet, dar si procentul de pachete carora li se va face ’drop’. La randul sau server-ul va trimite un pachet de acknowledgment care va avea numarul de pachet identic cu cel al pachetului SYN receptat, campul ack va fi egal cu campul dimensiune a pachetului SYN, iar flag-ul SYN va fi si el setat.

Atunci cand server-ul primeste un pachet obisnuit(care nu are flag-urile SYN si FIN active) se intrevad 3 cazuri. In primul caz numarul pachetului primit coincide cu valoarea counter-ului serverului, prin urmare datele pachetului sunt scrise direct in fisier si se trimite inapoi pachetul ack. De asemenea, server-ul interogheaza handler-ul pentru a vedea daca acesta detine pachete ulterioare pachetului receptionat pentru a le scrie si pe ele in fisier. In al 2-lea caz, numarul pachetului primit este mai mic decat cel al counter-ului, ceea ce inseamna ca pachetul a mai fost receptat, prin urmare se retrimite pachetul ack specific acestuia. In ultimul caz, numarul pachetului este mai mare decat valoarea counter-ului, prin urmare pachetul este dat handler-ului pentru stocare temporara, aceasta stocare realizandu-se doar in situatia in care pachetul nu exista deja in dictionarul handler-ului. Si pentru acest pachet server-ul trimite inapoi un pachet ack.

Ultimul pachet asteptat de catre server este pachetul FIN, ce va duce la inchiderea descriptorului de fisier, a socket-ului,dar si la iesirea din bucla in care rula server-ul, adica la inchiderea conexiunii. Inainte de inchiderea conexiunii, serverul a trimis la randul sau un pachet de ack cu flag-ul FIN activ si cu numarul de pachet corespunzator.

Am evidentiat deja faptul ca **handler-ul serverului**(ServerPackagesHandler) are rolul de a stoca temporar pachetele care nu sosesc in ordine, mai exact pachetele ulterioare pachetului asteptat, pachetul asteptat fiind cel care urmeaza a fi scris in fisier. Aceasta stocare se realizeaza folosind un obiect de tip dictionar, in care cheia este reprezentata de numarul pachetului, iar valoarea de pachetul propriu-zis. Handler-ul permite operatii precum adaugarea/eliminarea unui pachet sau verificarea existentei unui anumit pachet.

Clientul contine un counter care indica numarul pachetului care urmeaza a fi transmis, dar si un handler(ClientPackagesHandler) care il va ajuta la transmiterea si retransmiterea pachetelor. Clientul mai contine de asemenea variabilele cwnd ce indica dimensiunea ferestrei de congestie, sstresh corespunzatoare valorii de prag si o alta care indica numarul de pachete aflate in tranzit. Daca nu este configurat altfel, algoritmul porneste din starea de ’slow start’. Clientul consta intr-un thread ce ruleaza intr-o bucla infinita, bucla in care se sta atata vreme cat mai sunt pachete de transmis serverului.

Clientul va crea initial un pachet SYN ce va avea activ flag-ul SYN, campul dimensiune va fi completat cu lungimea campului de date, iar pe campul de date vor fi transmise denumirea fisierului, numarul total de pachete ce urmeaza a fi transmise, dimensiunea unui pachet, dar si procentul de pachete carora serverul le va face ’drop’. Acest pachet, dar si urmatoarele, vor fi date handler-ului pentru a le transmite serverului.

In urma transmiterii pachetului SYN clientul va astepta pachetul de acknowledgment, iar cand acesta va fi primit, o resursa catre fisier va fi deschisa si un nou thread va fi creat, thread ce se va ocupa de receptionarea pachetelor ack. Prin urmare client-ul va avea 2 thread-uri ce lucreaza in paralel, unul pentru citirea datelor din fisier, formarea pachetelor si trimiterea acestora catre handler pentru transmitere si un alt thread ce se va ocupa de receptionarea pachetelor de acknowledgment. Clientul va transmite pachete atata timp cat numarul de pachete aflate in tranzit este mai mic decat dimensiunea ferestrei de congestie. Atunci cand un pachet de ack este receptionat, valoarea ferestrei de congestie este actualizata in functie de starea algoritmului, mai exact, daca algoritmul se afla in starea de ’slow start’ atunci fereastra va creste cu 1, iar daca starea activa este cea de evitare a congestiei, fereastra va creste cu 1/cwnd. De fiecare daca cand un pachet este retransmis, faza de ’slow start’ este reinitializata, noua dimensiune a ferestrei fiind 1, iar valoarea de prag va fi egala cu jumatate din dimensiunea ferestrei de dinaintea retransmiterii pachetului. Dupa ce toate datele din fisier au fost transmise, pachetul FIN va fi trimis, iar un timer va fi creat. Daca pachetul de acknowledgment pentru pachetul FIN transmis nu va fi receptionat intr-un anumit interval de timp, atunci conexiunea va fi incheiata automat.

Rolul **handler-ului clientului**(ClientPackagesHandler) este acela de a transmite pachetele si de a le retransmite pe cele care nu au primit un acknowledgment. Pana la primirea pachetelor de acknowledgment corespunzatoare, pachetele vor fi stocate de catre handler intr-un dictionar. Cheia dictionarului va fi reprezentata de momentul de timp la care pachetul a fost transmis, iar valoarea de pachetul propriu-zis. Pachetele pentru care nu se primeste acknowledgment-ul intr-un interval de 0.5 secunde(timeout) vor fi retransmise. De altfel, handler-ul consta si el intr-un thread care itereaza peste elementele dictionarului si retransmite pachetele atunci cand aceasta valoare de timeout este depasita. Avand in vedere faptul ca thread-ul itereaza continuu peste elementele dictionarului, iar unul din thread-urile clientului poate modifica dimensiunea acestuia, in ideea ca atunci cand clientul creeaza un

nou pachet acesta este adaugat in dictionar sau atunci cand primeste un pachet de ack pachetul corespunzator este eliminat din dictionar, trebuie asigurat un mecanism de sincronizare ce asigura faptul ca dictionarul nu este modificat de catre client in timp ce acesta este parcurs de catre handler.

Aceasta sincronizare este asigurata prin folosirea unor obiecte de tip lock, operatiile de adaugare/eliminare a unui pachet, dar si de iterare asupra dictionarului fiind astfel sigure.

Modul de utilizare al aplicatiei:

- se creeaza serverul(receiver's view)
- ip-ul si portul sunt introduse pe interfata sender-ului
- se alege fisierul(sender's view)
- se transmite fisierul(sender's view)