Proiect Rețele de calculatoare
Transfer de fișiere – fereastră glisantă

Studenți: Tanasă Ciprian-Ionuț

Heghea Mihail-Cristian

Grupa: 1309A

În protocoalele anterioare, cadrele cu date erau transmise într-o singură direcţie. În cele mai multe situaţii practice, este necesar să se transmită date în ambele direcţii. O modalitate de a realiza transmisia de date full-duplex este de a avea două canale de comunicaţie separate, fiecare dintre ele fiind utilizat pentru traficul de date simplex (în direcţii diferite). Dacă se face aceasta, vom avea două circuite fizice separate, fiecare cu un canal "direct" (eng.: forward) pentru date şi un canal "invers" (eng.: reverse) pentru confirmări. În ambele cazuri lărgimea de bandă a canalului invers este irosită aproape în totalitate.

Următoarele trei protocoale sunt protocoale bidirecţionale care aparţin unei clase de protocoale numite protocoale cu fereastră glisantă (eng.: sliding window). Cele trei diferă între ele în termeni de eficienţă, complexitate şi necesar de tampoane, aşa cum vom vedea mai târziu. În cadrul acestora, ca în toate protocoalele cu fereastră glisantă, fiecare cadru expediat conţine un număr de secvenţă cuprins între 0 şi o valoare maximă. Maximul este de obicei 2n -1, ca numărul de secvenţă să se încadreze exact într-un câmp de n biţi. Protocoalele cu fereastră glisantă pas-cu-pas utilizează n=1, restricţionând numerele de secvenţă la 0 şi 1, dar versiuni mai sofisticate pot utiliza o valoare arbitrară a lui n.

Esenţa protocoalelor cu fereastră glisantă este aceea că, la orice moment de timp, emiţătorul menţine o mulţime de numere de secvenţă care corespund cadrelor pe care are permisiunea să le trimită. Se spune că aceste cadre aparţin ferestrei de transmisie (eng.: sending window). Similar, receptorul menţine de asemenea o fereastră de recepţie (eng.: receiving window), ce corespunde mulţimii de cadre care pot fi acceptate. Fereastra emiţătorului şi fereastra receptorului nu trebuie să aibă aceleaşi limite minime şi maxime şi nici măcar aceeaşi dimensiune. În unele protocoale ele au dimensiune fixă, dar în altele ele pot creşte sau scădea pe măsură ce cadrele sunt emise sau recepţionate.

Câmpul de confirmare conţine numărul ultimului cadru recepţionat fără eroare. Dacă acest număr corespunde cu numărul de secvenţă al cadrului pe care emiţătorul încearcă să-l transmită, emiţătorul ştie că a terminat cu cadrul memorat în tampon şi poate prelua următorul pachet de la nivelul său reţea. Dacă numărul de secvenţă nu corespunde, el trebuie să continue să trimită acelaşi cadru. De fiecare dată când este recepţionat un cadru, un alt cadru este trimis de asemenea înapoi.

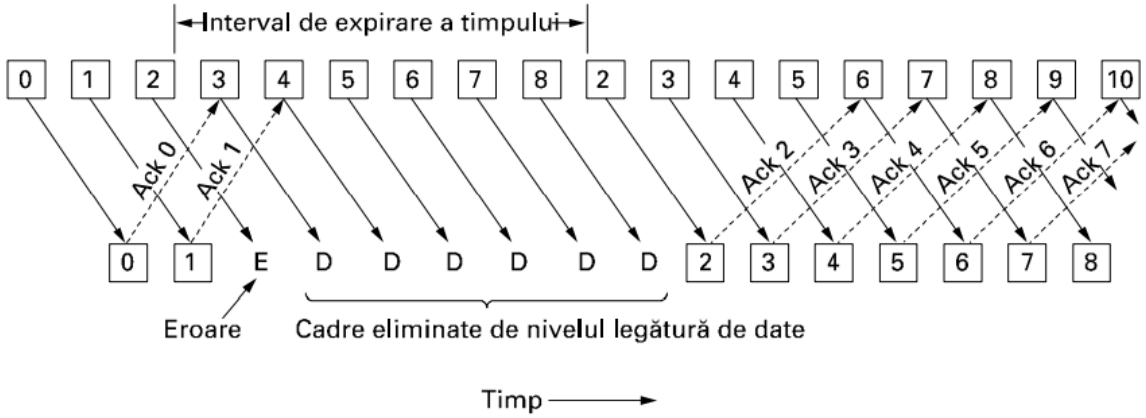
Exista doua moduri de baza de tratare a erorilor in prezenta benzii de asamblare. Un mod, numit revenire cu n pasi (go back n), este ca receptorul sa elimine pur si simplu cadrele care urmeaza, netrimitand confirmari pentru cadrele eliminate.

Cealalta strategie generala de tratare a erorilor atunci cand este folosita banda de asamblare se numeste repetare selectiva (selective repeat). Cand aceasta este utilizata, un cadru incorect este respins, dar toate cadrele corecte care il urmeaza sunt memorate. Cand contorul de timp al emitatorului expira, cel mai vechi cadru neconfirmat este retransmis.

Protocol de revenire cu n paşi (Go Back n)

Până acum am făcut presupunerea tacită că timpul de transmisie necesar pentru ca un cadru să ajungă la receptor plus timpul de transmisie a confirmării este neglijabil. Uneori această presupunere este în mod cert falsă. Pentru a mări eficiența de utilizare a lățimii de bandă soluţia constă în a permite emiţătorului să transmită până la w cadre, în loc de unul singur. Cu o alegere potrivită a lui w emiţătorul va putea să transmită continuu cadre pentru un timp egal cu timpul de tranzit, fără a umple fereastra.

Mai jos este prezentat protocolul de revenire cu n paşi pentru cazul în care fereastra receptorului are dimensiune unu. Cadrele 0 şi 1 sunt primite şi confirmate corect. Cadrul 2, totuşi, este alterat sau pierdut. Emiţătorul, care nu ştie de această problemă, continuă să trimită cadre până când timpul pentru cadrul 2 expiră. Apoi se întoarce la cadrul 2 şi o ia de la început cu el, trimiţând din nou cadrele 2, 3, 4 etc.



Protocolul Datagramelor Utilizator (UDP)

User Datagram Protocol este un protocol de comunicație pentru calculatoare ce aparține nivelului Transport (nivelul 4) al modelului standard OSI.

Împreună cu Internet Protocol (IP), acesta face posibilă livrarea mesajelor într-o rețea. Spre deosebire de protocolul TCP, UDP constituie modul de comunicație fără conexiune. Este similar cu sistemul poștal, în sensul că pachetele de informații (corespondența) sunt trimise în general fără confirmare de primire, în speranța că ele vor ajunge, fără a exista o legătură efectivă între expeditor și destinatar.

Antetul UDP este alcătuit din 4 câmpuri fiecare având lungimea de 2 octeți.

Biti	0 - 15	16 - 32
0	Portul sursa	Portul destinație
32	Lungime	Suma de control
64	Date	

- Portul sursa - în adresarea bazata pe IPv4 acest câmp este opțional. Daca nu este utilizat acest câmp, are valoarea zero; când reprezinta informație semnificativa, el va indica portul inițiator al procesului de transmisie a datagramelor.
- Portul destinație - spre deosebire de portul sursa, câmpul este obligatoriu și indica portul de recepție
- Lungime - acest câmp indica lungimea în octeți a datagramei: antet plus secțiune de date (valoarea minima a lungimii este 8). Limita teoretica a unui pachet UDP este de 64 KB.
- Suma de control - asigura imunitatea la erori; se calculează ca fiind complementul fata de 1 (pe 16 biți) a pseudo-antetului cu informații extrase din antetul IP, antetului UDP și a câmpului de date, eventual se completează cu zerouri pentru a atinge lungimea stabilita.

Implementare

Vom implementa două aplicații scrise în Python care să realizeze transferul de fișiere utilizând metoda Go Back N. Primul pachet trimis către receptor va conține informații despre transfer precum: numărul de octeți utilizați pentru reprezentarea numarului de pachet, numele fișierului, dimensiunea fișierului. Restul pechetelor vor conține un câmp prin intermediul căruia va fi determinat tipul pachetului

(acknowledge, not-acknowledge sau pachet ce conține date), numărul pachetului și datele.

Implementare aplicație de trimitere

Pentru partea de trimitere se instanțiază un obiect de tip Sender care are trei metode mai importante cum ar fi:

- readFile care realizează citirea fișierului selectat pentru trimitere și determină numărul de pachete în care este împărțit fișierul;
- sendInfo care se ocupă cu trimiterea pachetului ce conține informatții despre transfer: primul câmp reprezintă tipul pachetului, al doilea numărul de pachete, al treilea portul pe care aplicația de trimitere așteptă pachetele de confirmare și numele fișierului;
- sendData care realizează efectiv transferul care este realizat după cum urmează: inițial se trimit un număr de pachete egal cu dimensiunea ferestrei după care se așteaptă primirea pachetelor de confirmare. Dacă recepția nu se realizează într-un anumit interval de timp se transferă din nou toate pachetele din fereatră. Pe masură ce se primesc pachetele de confirmare, fereastra este glisată cu un cadru.

Pentru realizarea interfeței grafice am ales folosirea interfeței Tkinter. Din interfața grafică se poate configura IP-ul si portul aplicației de trimitere precum și IP-ul și portul aplicației de recepție. Tot din interfața grafică se poate selecta fișierul pentru transfer utilizând o fereastră de tipul filedialog.

Implementare aplicație pentru recepție

Aplicația receiver.py are o interfață din care se poate alege adresa IP, câmpul este completat automat cu adresa IP de pe placa de rețea, însă dacă PC-ul este conectat într-o rețea locală prin WIFI, este necesară introducerea adresei IP alocate. Al doilea câmp este cel pentru ‚port', care trebuie completat pentru a putea rula aplicația. În cazul în care se dorește deschiderea mai multor instanțe ale aplicației, portul trebuie să fie diferit.

Pentru a porni aplicația în faza de ‚așteptare' a transferului, se apasă butonul ‚Start', în momentul în care aplicația sender începe să trimită pachete, progresul este afișat în ‚progress bar-ul de pe interfață.

Interfața generează un Thread, care va rula codul efectiv al aplicației „recevier.py” în paralel cu interfața, astfel se poate vizualiza în timp real si progress bar-ul.

Pe parcurul rulării aplicației, mai multe mesaje de informare sunt scrise în fișierul „logging.txt”, astfel se poate urmări cum s-a realizat trasferul și cum au fost retransmise pachetele pierdute.

Aplicația așteaptă un prim pachet care conține informații despre trasfer, numărul de pachete ce trebuie așteptate, portul pe care trebuie să trimită ACK și numele fișierului transferat. După primirea pachetului cu informații se așteaptă pachetele cu date. Se așteaptă mai întâi pachetul cu numărul de secvență 1. Atunci când acesta ajunge, pentru a simula „pierderea” pachetelor, există o șansă de 0.3% ca acel pachet să nu fie folosit și să nu se trimită ACK către „sender”, astfel acel pachet, cât și restul pachetelor din fereastră sunt trimise iar. Dacă pachetul este primit, se trimite ACK către „sender” și se așteaptă pachetul cu numărul de secvență 2, la fel si pentru restul pachetelor. La fiecare pachet primit, datele din pachet sunt salvate în fișier, astfel scrierea efectivă a datelor are loc pe parcurs. După terminarea transferului, dacă se dorește așteptarea unui alt trasfer, se poate apăsa iar pe „Start”, eventual se modifica portul.