Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică și Microelectronică

Catedra Tehnologia Informației

Laboratorul 5.

Programarea în rețea

A efectuat: studenta gr. FI-171 Ciocanu Carolina

A examinat: Buldumac Oleg

Chisinau, 2020

**Condiții:**

Aplicație Client-Server UDP

Să se creeze o aplicație Client-Server UDP utilizând Sockets API

**Pentru nota 9 și 10:**

• să se creeze GUI(Graphical User Interface) pentru aplicație

**Atenție:**

• Nu se admite aplicații simple de genul Echo Client-Server

**Întrebări**

• Ce este un protocol orientat către tranzacții, fără conexiune ?

• Ce tipuri de aplicații beneficiază în general de utilizarea protocolului UDP ?

• De ce protocolul UDP nu garantează că datele vor fi transmise cu succes ?

• Diferența dintre blocking si non-blocking sockets

• În protocolul TCP există Three Way Handshake, de ce în UDP nu există ?

• Numiti cele 2 apeluri de sistem necesare pentru a crea un server UDP

• Care este rolul metodei bind() ?

• Care este rolul metodelor sendto() și recvfrom() ?

• Care este dimensiunea antetului unui pachet UDP în octeți ?

• Într-o conexiune UDP, clientul sau serverul trimite mai întâi datele ?

• Care este adresa de loopback IPv6 și care este rolul ei ?

• De unde știe un sistem de operare ce aplicație este responsabilă pentru un pachet primit din

rețea ?

• Datele primite prin recvfrom() au întotdeauna aceeași dimensiune cu datele trimise cu sendto()

?

• Este acceptabil să închei execuția programului dacă este detectată o eroare de rețea ?

• De ce nu este folosit algoritmul Nagle în protocolul UDP ?

• Ce instrumente listează socket-urile UDP deschise în sistemele de operare Windows și Linux ?

• Același program poate folosi UDP și TCP ?

• Diferența dintre aplicații UDP Unicast, Broadcast, și Multicast

• Ce face mai ușor multiplexarea cu UDP decît cu TCP ?

• În protocolul UDP este un antet „Total length”, cum se calculează și care este rolul lui ?

**• Ce este un protocol orientat către tranzacții, fără conexiune ?**

Când o parte a unei aplicații dorește să trimită pachete către cealaltă parte a aplicației, programul de expediere trimite pur și simplu pachetele. Deoarece nu există nicio procedură de strângere de mână înainte de transmiterea pachetelor de date, datele pot fi livrate mai devreme. Acest lucru face ca serviciul fără conexiuni să fie ideal pentru aplicații simple orientate către tranzacții, dar nu există nici un transfer de date fiabil, astfel încât o sursă nu știe niciodată sigur ce pachete au ajuns la destinație. Serviciul de conectare pe Internet se numește User Datagram Protocol (UDP)

**• Ce tipuri de aplicații beneficiază în general de utilizarea protocolului UDP ?**

Avantajele și dezavantajele UDP

UDP are o serie de avantaje pentru diferite tipuri de aplicații, inclusiv: Fără întârzieri de retransmisie - UDP este potrivit pentru aplicațiile sensibile la timp care nu își pot permite întârzieri de retransmisie pentru pachetele abandonate. Exemple includ Voice over IP (VoIP), jocuri online și media de streaming.UDP: orice lucru în care nu îți pasă prea mult dacă primești toate datele întotdeauna

Tunneling / VPN (pachetele pierdute sunt ok - protocolul tunel are grijă de el)

Transmiterea media (cadrele pierdute sunt ok)

Jocuri care nu le pasă dacă primești fiecare actualizare

Mecanisme de difuzare locală (aceeași aplicație care rulează pe mașini diferite „descoperindu-se” reciproc)

**• De ce protocolul UDP nu garantează că datele vor fi transmise cu succes ?**

UDP nu garantează securitatea și integritatea datelor: Absența autentificării reciproce între destinatar și destinatar asigură o viteză excelentă de transmitere a UDP - cu toate acestea, protocolul nu poate garanta nici completitudinea, nici securitatea pachetelor de date. Secvența corectă a pachetelor trimise nu este, de asemenea, garantată. Din acest motiv, serviciile care utilizează UDP trebuie să-și asigure propriile măsuri pentru corectare sau protecție.

**• Diferența dintre blocking si non-blocking sockets**

Sockets pot fi în modul de blocare sau de blocare, care este controlat prin apelarea funcției ioctlsocket. În modul de blocare, apelurile API socket de recv, send, connect (doar TCP) și accept (numai TCP) vor bloca la nesfârșit până când acțiunea solicitată a fost efectuată. În modul care nu blochează, aceste funcții revin imediat. select se va bloca până când priza este gata. selectează un parametru de timp care controlează timpul de așteptare pentru finalizarea acțiunii sau o eroare returnată.

**• În protocolul TCP există Three Way Handshake, de ce în UDP nu există ?**

UDP (User Data Protocol, User Datagram Protocol) este un protocol corespunzător TCP. Este un protocol fără conexiune. Nu stabilește o conexiune cu cealaltă parte, ci trimite direct pachetele de date!

UDP este potrivit pentru aplicații în care doar o cantitate mică de date sunt transmise simultan și cerințele de fiabilitate nu sunt ridicate. De exemplu, folosim adesea comanda "ping" pentru a testa dacă comunicarea TCP / IP între cele două gazde este normală. De fapt, principiul comenzii „ping” este de a trimite pachete ICMP la cealaltă gazdă, iar apoi cealaltă gazdă confirmă că pachetul este primit. Mesajul dacă sosește pachetul este redat în timp, apoi rețeaua este conectată. Eficiența sa de comunicare este ridicată, dar fiabilitatea sa nu este la fel de bună ca TCP.

**• Numiti cele 2 apeluri de sistem necesare pentru a crea un server UDP**

Listen() si Accept()

Folosim apelul de sistem recvfrom pentru a aștepta o datagramă de intrare pe o anumită adresă de transport (adresa IP și numărul portului). UDP este un protocol de date (mesaj) fără conexiune, nesigur, deci nu este nevoie să ascultați noi conexiuni - datagramele pot intra în orice ordine din orice sursă. Se datorează faptului că UDP nu este orientat către conexiune, așa cum este TCP, deci nu are niciun sens în ascultare / acceptare, de parcă există SOCK\_STREAM.

**• Care este rolul metodei bind() ?**

bind () definește o relație între soclul pe care l-ați creat și adresele disponibile pe gazda dvs. De exemplu, puteți lega o priză pe toate adresele sau pe un anumit IP care a fost configurat pe un adaptor de rețea de către sistemul de operare al gazdei.

bind () definește o relație între soclul pe care l-ați creat și adresele disponibile pe gazda dvs. De exemplu, puteți lega o priză pe toate adresele sau pe un anumit IP care a fost configurat pe un adaptor de rețea de către sistemul de operare al gazdei.

Pentru o aplicație de server, utilizatorii le place uneori să restricționeze priza pentru a fi legați doar la o anumită adresă din motive de securitate sau doar pentru a cartona serviciile unde doresc să fie disponibile. Acesta este motivul pentru care majoritatea serverelor permit configurarea adreselor de legare pentru utilizatori cu fișierul de configurare al aplicației.

**• Care este rolul metodelor sendto() și recvfrom() ?**

UDP, de exemplu ar apela recvfrom (), astfel încât să obțină informațiile despre adresa sursă, să proceseze solicitarea, să creeze răspunsul și să o trimită la adresa respectivă prin sendto (). Nici o conexiune () implicată nicăieri, nu este posibil să nu folosiți send () sau recv ().

În schimb, clientul trimite doar un datagram pe server folosind funcția sendto care necesită adresa destinației ca parametru. recvfrom returnează adresa IP a clientului, împreună cu datagrama, astfel încât serverul poate trimite un răspuns clientului.

**• Care este dimensiunea antetului unui pachet UDP în octeți ?**

O datagramă UDP constă dintr-un antet UDP și datele transportate. Mărimea unui antet UDP este de 8 octeți.

**• Într-o conexiune UDP, clientul sau serverul trimite mai întâi datele ?**

În general, aplicația server ar trebui să primească mai întâi datele pentru a stabili clientul căruia vor fi trimise datele. În caz contrar, eroarea -QERR\_NO\_DESIGNATED\_PEER poate fi returnată. În mod similar, blocul Stream Connect nu se „conectează” de fapt la o gazdă la distanță. În schimb, stabilește adresa care va fi folosită la trimiterea de pachete. Prima dată când este trimis un pachet, priza devine implicit legată de acea adresă. Prin urmare, atunci când utilizați blocul Stream Connect pentru a crea un flux, o operațiune de trimitere trebuie întotdeauna făcută mai întâi. Modul în care se comportă driverul de protocol odată ce este primit un datagram depinde de opțiunea peer. Consultați discuția despre opțiunea peer de mai jos pentru detalii. Opțiunea implicită este peer = one.

**• Care este adresa de loopback IPv6 și care este rolul ei ?**

În IPv6, adresa IPv6 rezervată pentru utilizarea în loopback este 0000: 0000: 0000: 0000: 0000: 0000: 0000: 0001/128. Această adresă loopback este atât de lungă și poate fi simplificată ca

:: 1/128. Adresa Loopback este un număr IP special (127.0. 0.1), care este desemnat pentru interfața loopback software a unei mașini. Interfața loopback permite profesioniștilor IT să testeze software-ul IP fără a-și face griji pentru driverele sau hardware-ul deteriorat sau corupt.

**• De unde știe un sistem de operare ce aplicație este responsabilă pentru un pachet primit din rețea ?**

Ruterele operează la nivelul 3 al [modelului OSI](https://ro.wikipedia.org/wiki/Modelul_OSI). Ele folosesc deci [adresele IP](https://ro.wikipedia.org/wiki/Adres%C4%83_IP) (de rețea) ale [pachetelor](https://ro.wikipedia.org/w/index.php?title=Pachet&action=edit&redlink=1) aflate în tranzit pentru a decide către care anume [interfață](https://ro.wikipedia.org/wiki/Interfa%C8%9B%C4%83) de ieșire trebuie să trimită pachetul respectiv. Decizia este luată comparând adresa calculatorului destinație cu înregistrările (câmpurile) din [tabela de rutare](https://ro.wikipedia.org/wiki/Tabel%C4%83_de_rutare). Aceasta poate conține atât înregistrări statice (configurate/definite de către administratorul rețelei), cât și dinamice, aflate de la ruterele vecine prin intermediul unor [protocoale de rutare](https://ro.wikipedia.org/wiki/Protocol_de_rutare).

**• Datele primite prin recvfrom() au întotdeauna aceeași dimensiune cu datele trimise cu sendto()?**

Argumentul la funcția sendto este structura adresei socket care conține adresa protocolului (de exemplu, adresa IP și numărul de port) al destinației. Mărimea acestei structuri de adrese socket este specificată de argumentul addrlen. Funcția recvform completează structura adresei de priză la care se face referire de la, scriind adresa de protocol a expeditorului datagramei. Numărul de octeți stocați în structura adresei socket-ului este de asemenea returnat procesului de apelare în ansamblu, așa cum este indicat de argumentul addrlen. Rețineți că ultimul argument la funcția sendto este o valoare întreagă, în timp ce ultimul argument la funcția recvfrom este un indicator către o valoare întreagă (un argument valoare-rezultat).

Ultimele două argumente ale funcției recvfrom sunt similare cu ultimele două argumente ale funcției accept: conținutul structurii adresei socket la finalizare ne spune cine a trimis datagrama (în cazul UDP) sau cine a inițiat conexiunea (în cazul TCP). Ultimele două argumente ale funcției sendto sunt similare cu ultimele două argumente ale funcției de conectare: populăm structura adresei socket cu adresa de protocol a destinatarului datagramei (în cazul UDP) sau adresa gazdă cu care va fi stabilită conexiunea (în cazul TCP).

**• Este acceptabil să închei execuția programului dacă este detectată o eroare de rețea ?**

Programul trebuie să poată încheia execuția sa, normal sau anormal.

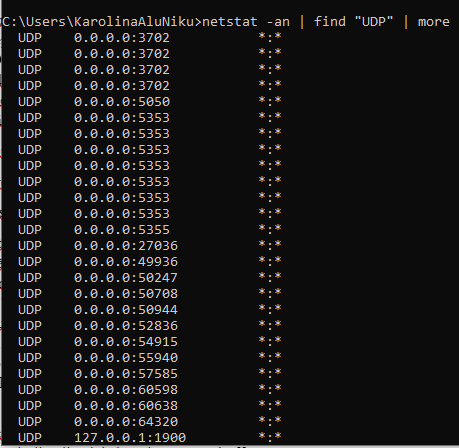
**• De ce nu este folosit algoritmul Nagle în protocolul UDP ?**

Acest algoritm interacționează prost cu recunoașterea întârziată TCP (întârziată ACK), o caracteristică introdusă în TCP aproximativ la aceeași perioadă la începutul anilor 1980, dar de un grup diferit. Cu ambele algoritmi activate, aplicațiile care realizează două scrieri succesive la o conexiune TCP, urmate de o citire care nu va fi îndeplinită decât după ce datele de la al doilea scriere au ajuns la destinație, experimentează o întârziere constantă de până la 500 de milisecunde, " Întârziere ACK ". Din acest motiv, implementările TCP oferă de obicei aplicațiilor o interfață pentru a dezactiva algoritmul Nagle. Aceasta se numește de obicei opțiunea TCP\_NODELAY. Pe Microsoft Windows, comutatorul de registru TcpNoDelay decide implicit.

O soluție recomandată de Nagle este să evite algoritmul să trimită pachete premature prin tamponarea scrierii aplicației și apoi curățarea buffer

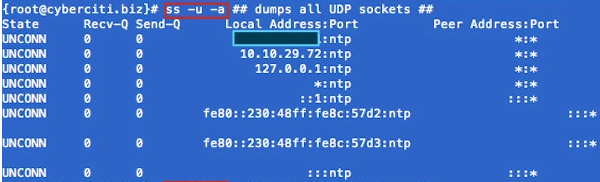
**• Ce instrumente listează socket-urile UDP deschise în sistemele de operare Windows și Linux ?**

Windows



Linux

# ss -u –a



**• Același program poate folosi UDP și TCP ?**

Nu numai că este posibil, dar întregul Internet a depins istoric de el.

Mai exact, serverul DNS BIND DNS folosit pentru a accepta interogări (UDP / 53) și cereri de transfer de zonă (TCP / 53) din aceeași instanță.

**• Diferența dintre aplicații UDP Unicast, Broadcast, și Multicast**

Datele sunt transportate printr-o rețea prin trei metode simple, adică Unicast, Broadcast și Multicast. Deci, să începem să rezumăm diferența dintre aceste trei:

Unicast: de la o sursă la o destinație, adică unu la unu

Broadcast: de la o sursă la toate destinațiile posibile, adică One-to-All

Multicast: de la o sursă la destinații multiple care afirmă un interes în primirea traficului, adică One-to Many

**• Ce face mai ușor multiplexarea cu UDP decît cu TCP ?**

În TCP, conexiunile persistă între pachete, deci mașina de destinație trebuie să țină evidența pachetelor care aparțin conexiunii. Pachetele UDP sunt evenimente unice, astfel că nu este necesar un fel de contabilitate.

**• În protocolul UDP este un antet „Total length”, cum se calculează și care este rolul lui ?**

Câmpul lungime antet UDP este lungimea antetului UDP plus datele UDP. Este într-adevăr redundant, deoarece această lungime poate fi calculată din câmpul lungimii totale a antetului IP în care lungimea datagramei UDP este lungimea totală IP minus lungimea antetului IP.