



# ŞƏBƏKƏ

# PROQRAMLAŞDIRMA

NET.FRAMEWORK-DA  
ŞƏBƏKƏ PROQRAMLAŞDIRMA

TCP VƏ UDP SOKETLƏR, UNICAST  
BROADCAST, MULTICAST

HTTP, SMTP, FTP ŞƏBƏKƏ  
PROTOKOLLARININ İSTİFADƏSİ

# Dərs №1

## Net.Framework-da şəbəkə proqram- laşdırması

### Mündəricat

1. Şəbəkə proqramlaşdırması nədir .....	3
2. Şəbəkə proqramlaşdırmasının məqsəd və vəzifələri .....	5
3. Şəbəkə nədir .....	7
4. OSI modeli .....	8
5. Baza terminlər .....	10
6. Socket class-ı .....	17
7. Asinxron soketlər .....	24
8. Ev tapşırığı .....	37

# 1. Şəbəkə proqramlaşdırması nədir

Bu dərsdə biz “şəbəkə proqramlaşdırması” adlanan yeni təlim kursuna başlayırıq. Kursun adından göründüyü kimi, burada öyrənəcəyimiz texnologiyaların müxtəlif kompüterlərin şəbəkə üzrə qarşılıqlı əlaqəsi üçün nəzərdə tutulmuş tətbiqlərin proqramlaşdırmasıyla birbaşa əlaqəsi vardır.

XX əsrin altmışıncı illərindən başlayaraq dünyada, məlumatların ötürülməsi üçün şəbəkələrin yaradılması işləri başlandı. Telefon şəbəkələrinin istifadəsi ilə bir neçə kompüterin qarşılıqlı əlaqə imkanlarının ictimai nümayişi 29 Oktyabr 1969 –cu il saat 21:00 da baş tutmuşdur. Şəbəkə iki terminaldan ibarət idi, biri Kaliforniya universitetində, digəri isə ondan 600 km aralıda – Stenford universitetində yerləşirdi.

Tərtibat ABŞ-ın (DARPA) Qabaqcıl Müdafiə Araşdırma Layihələri Agentliyi tərəfindən koordinasiya edilirdi. Burada yaradılmış şəbəkənin daha vacib xüsusiyyətlərindən biri imtinaya davamlılıq idi, yəni, şəbəkənin bir hissəsinin məhv olması zamanı (məsələn, ölkə ərazisinə nüvə zərbələrinin nəticəsində) şəbəkənin qalan hissəsinin işi davam etdirə bilməsi.

Beləcə, XX əsrin 70-ci illərindən başlayaraq məlumat ötürülməsi şəbəkələrinin istifadəsinə başlandı. Təbii ki, proqram məhsullarının öz aralarında qarşılıqlı əlaqələrinin

təmin edilməsi üçün nəzərdə tutulan proqram təminatını yazmağa tələbat yarandı.

Ancaq o vaxt şəbəkə digər texniki standartlarla qurulmuş digər şəbəkələrlə qarşılıqlı əlaqədə ola bilmirdi. 1970-ci illərin sonundan 80-ci illərin ortalarına qədər standartlaşdırılan məlumat ötürülməsi protokolları inkişaf etməyə başladı.

XX əsrin 90-cı illərinə qədər şəbəkə, əsasən elektron məktubların göndərilməsi üçün istifadə edilirdi, elə o zaman da məktub göndərilməsinin ilk siyahıları, xəbər qrupları və elan lövhələri yarandı.

1983-cü il Yanvarın 1-dən ARPA şəbəkəsi iş üçün NCP protokolunun əvəzinə TCP/IP protokolunu istifadə etməyə başladı. TCP/IP indiyədək şəbəkələrin birləşdirilməsi (“qatlaşdırılması”) üçün uğurla tətbiq olunur. Elə məhz o vaxtdan da (1983-cü ildə) ARPA şəbəkəsinin adında “İnternet” termini yarandı.

## 2. Şəbəkə proqramlaşdırmasının məqsəd və vəzifələri

---

Məlumat ötürülmə şəbəkələrinin bu cür sürətli inkişafı dövründə, şəbəkənin qırılması zamanı mövcudluğu mümkün olmayan yeni tətbiq calss-ları yarandı. Bu cür tətbiqlər **server - müştəri** tətbiqləri adını aldı.

Şəbəkədə işləyərkən bir kompüter **server** rolunu oynayır, digəri isə **müştəri** kimi qoşulur, burada nəzərdə tutulan odur ki, hər iki kompüter eyni qoşulma üsulundan (məsələn kabel tipi, signalın genişliyi, modulyasiya üsulu) və ünsiyyət üçün eyni “dildən” istifadə etməlidirlər.

Kompüterlər arasındakı mübadilə üsullarını standartlaşdırmaq üçün məlumat ötürülməsi **protokolları** adlanan, məlumat mübadiləsi qaydalarının xüsusi dəsti istifadə olunur.

Burada vəzifə minimum iki hissəyə ayrılır: tətbiq məntiqinin bir hissəsi serverdə, digəri isə müştəridə realizə edilir. Burada daha çox server hissəsində istifadəçi interfeysinin olması nəzərdə tutulmur. Bundan əlavə, müştəri hissəsi, serverlə şəbəkə arasında qarşılıqlı əlaqə üsulları ilə olan istifadəçi interfeysini özündə təmsil edir.

Müştəri-server tətbiqləri şəbəkədə olan kompüterlər arasında hesablama əməllərini bölüşdürməyə imkan verir, bu da bir neçə məşində yerinə yetirilən əməlləri optimizə etməyə imkan yaradır.

Müştəri-server tətbiqləri, həmçinin məsələləri hissələrə ayırmağa imkan yaradır, bir məsələnin həllini şəbəkədə olan bir neçə kompüterə ötürür və bu cür texnologiya **hesablamaların bölüşdürülməsi** adını almışdır.

Beləcə, şəbəkə tətbiqləri tərtibatının əsas məqsədi, lokal şəbəkədə (intranet) və yaxud bütün dünya şəbəkəsi üzrə (internet) səpələnmiş kompüterlərin effektiv və təhlükəsiz qarşılıqlı əlaqəsinin yaradılmasıdır.

### 3. Şəbəkə nədir

Gəlin bu dərstdə istifadə edəcəyəmiz terminləri müəyyənləşdirək. Beləliklə, ilk olaraq “şəbəkə” sözü altında nəyi başa düşəcəyimizi müəyyən edək.

**Şəbəkə - rabitə kanallrında bir-biri ilə əlaqələndirilmiş kompüterlər və ya qurğular qrupudur.**

Bütün bu qurğuları (kompüterlər, marşrutizatorlar, şlüz, printerlər) şəbəkə qovşaqları adlanır. Qovşaqlar bir-biriylə kanallarla birləşir, kanallar kimi istənilən rabitə əlaqələri ola bilər (kabel, optik kabel, optik atmosfer, radio və s.). Bu **rabitə kanalları** ilə şəbəkə qovşaqları bir-birinə **ismaric** yollayaraq öz aralarında qarşılıqlı əlaqədə ola bilər.

Şəbəklər öz ölçülərinə görə **lokal** – bir bina daxilində və ya rayon daxilində bir-birinə yaxın yerləşən binalardakı qovşaqları birləşdirən və **qlobal** - bir neçə lokal şəbəkələri öz aralarında birləşdirən – ola bilər.

## 4.OSI modeli

Rabitə kanalları ilə məlumat mübadiləsi protokollarının formalizəsi məqsədi ilə, ISO təşkilatı tərəfindən standart kimi OSI (Open System Interconnection) yeddimərhələli model adını almış protokol modeli qəbul edilmişdir. Bu model çərçivəsində, şəbəkədə qovşaqların qarşılıqlı əlaqəsinin əsas vəzifələri qeyd edilmişdir.

OSI modelinin hər mərhələsinin öz xüsusi vəzifələri vardır və onların hər biri yuxarıda yerləşən və caridən aşağıda yerləşən mərhələlərlə bağlıdır.

Bunlar həmin o yeddi mərhələdir:

7	tətbiqi (Application),
6	təqdimat (Presentation),
5	sessiya (Session),
4	nəqliyyat(Transport),
3	şəbəkə (Network),
2	kanal (Data Link)
1	fiziki (Physical)

Şəbəkə üzrə iki kompüterin qarşılıqlı əlaqə problemləri bu qarşılıqlı əlaqəni proqramlaşdıran proqramçının şəbəkənin işini dəqiq anladığı halda həll oluna bilər.

Siz indicə “Şəbəkə texnologiyalarına giriş” kursuna qulaq asdınız. Bu kursda siz ətraflı olaraq OSI modelini, lokal və qlobal şəbəkələrin qurulmasını və fəaliyyət funksiyalarını öyrəndiniz.



Gəlin, OSI modelinin bu və ya digər mərhələsinin vəzifəsini sizin yadınıza salaraq proqramçını ilk növbədə maraqlandıran məqamları göstərək.

**Fiziki mərhələdə** şəbəkənin qurulmasına tətbiq edilən məlumat ötürülməsinin mühiti, ötürülən informasiyanın kodlaşdırma üsulları, şəbəkənin qurulmasına tətbiq edilən birləşdiricilər və kabellər müəyyən olunur.

Tətbiqi məsələlərin həlli ilə məşğul olan proqramçıya bu mərhələdə nəşə etmək lazım deyil.

**Kanal mərhələsi** ilk növbədə ötürmə mühitinin əlverişliliyini yoxlayır. İkincisi, xətlərin tapılma və korreksiya mexanizmlərini realizə edir.

**Şəbəkə mərhələsi** (Network layer) bir neçə şəbəkəni birləşdirən vahid nəqliyyat sisteminin yaranmasına xidmət edir. Burada bu şəbəkələr informasiya ötürməsinin müxtəlif prinsiplərini istifadə edə və strukturuna uyğun sərbəst təşkil oluna bilər!

Bu mərhələdə qovşaqların məntiqi (kanal mərhələsinin fiziki ünvanlarından fərqli olaraq) ünvanlanması istifadə olunur. İş burasındadır ki, kanal mərhələsinin fiziki ünvanlarını yalnız lokal şəbəkə daxilində istifadə etmək olar.

## 5.Baza terminləri

Adətən (və ya adəti üzrə) şəbəkə mərhələsində IP protokolu istifadə edilir, yəni şəbəkədəki hər bir qovşağa bu şəbəkə üçün hansısa unikal IP ünvan verilir.

Proqramçı şəbəkə tətbiqinin realizə olunması zamanı qoşulmaq istədiyi qovşağın IP ünvanını və ona qoşulma icra edilən qovşağın IP ünvanını bilməlidir.

Şəbəkə mərhələsində proqramçı, şəbəkə API – socket-in (yuva) xüsusi obyektini yaradır, bu, konkret şəbəkə ünvanı və nəqliyyat protokolu class-ı ilə əlaqələndirilir.

Soketdən istifadə edərək proqramçı, şəbəkədə olan digər soketlə qarşılıqlı əlaqəni həyata keçirə bilər.

**Nəqliyyat mərhələsi** – Burada daha ətraflı danışmaq lazım gələcək. Belə ki, nəqliyyat mərhələsində biz ilk olaraq tətbiqlərlə qarşılaşırıq. Nəqliyyat mərhələsində biz port anlayışı və ya son nöqtə (end point) ilə qarşılaşırıq, o şəbəkə birləşməsinə qəbul etməyə və ya şəbəkənin digər qovşağında yerləşən tətbiqə birləşməyə hazır olan tətbiqi təsvir edir. Nəqliyyat mərhələsində proqramçı protokollardan hansını istifadə edəcəyini seçməlidir. Nəqliyyat mərhələsinin bir neçə yayılmış protokolları mövcuddur. Proqramçılar tərəfindən tez-tez istifadə edilən protokollar TCP (Transmission Control Protocol) və UDP (User Datagram Protocol), daha az-az isə istifadə edilənlər isə RTP (Real-time Transport Protocol) və başqa protokollardır.

**TCP** protokol məsafədəki qovşağa paketin çatdırılması üçün zamanət zəruri olduqda istifadə olunur. TCP protokolundan iş zamanı müştəri seansın əvvəlində serverlə bağlantını quraşdırır, sonda isə bağlantını kəsir. Məlumatların ötürülməsi yalnız bağlantı olduqda mümkündür. Burada qəbul edən tərəf ötürən tərəfə məlumatların alınmasına təsdiq yollayır, ötürülən məlumatların təhrifi və ya itirilməsi zamanı ötürən tərəf paketin ötürülməsini təkrar edir.

**UDP** protokolunda məlumatların ötürülməsi çatdırılmasına zamanət vermədən və daimi bağlantıya dəstək vermədən icra edilir. Məlumat ötürülməsinin tamlığının təmin edilməsi yuxarı mərhələ protkolları ilə icra edilməli və ya rədd edilməlidir.

UDP protokolu sürətin vacib olduğu yerdə daha çox istifadə olunur, məlumatların bir hissəsinə isə laqeydlik göstərmək olar.

Əgər ötürülən kadrların ardıcılığının saxlanması vacibdirsə, onda RTP protokolundan istifadə etməyə dəyər. Bu protokol adətən multimedia (nitq, video) məlumatlarının ötürülməsi zamanı tətbiq olunur. RTP protokolu UDP protokoluna əsaslanır və hər paket daxilində ötürülmə zamanı paket ardıcılığını bərpa etməyə imkan yaradan informasiyanın mövcudluğu ilə fərqlənir.

**Sessiya mərhələsi** – bağlantının qoşulma qaydasını, yəni sessiyanın inisializasiya və keçirilmə qaydasını müəyyən edir. **Bu mərhələnin realizəsi bütünlüklə proqramçılardan üzərinə düşür.**

Məhz bu mərhələdə müştəri soketi və serverdə olan soket arasında göndərilən ismarıç mübadiləsi müəyyən edilir.

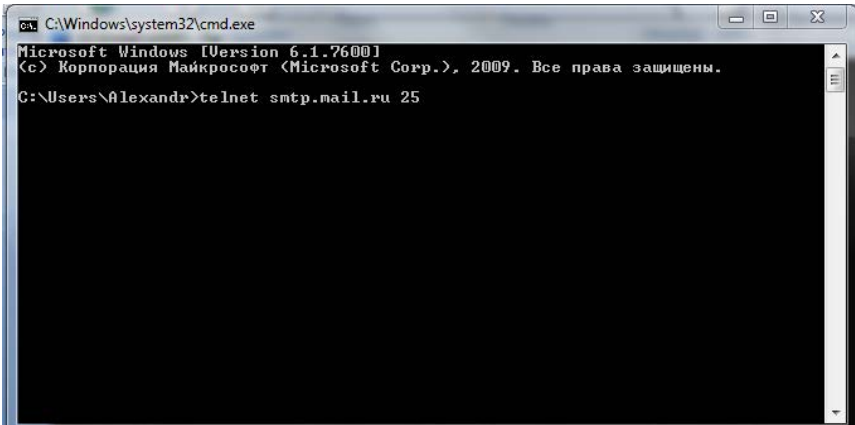
**Təqdimat** mərhələsində ötürülən ismarıcın forması müəyyən edilir. Burada proqramçı öz ismarıcının təqdimatını seçir, yəni, o məlumatları hansı şəkildə ötürüb qəbul edəcək. Bu binar ardıcillıq şəklində, ya da mətn rejimində olan (ASCII və ya UNICODE) komandalar şəklində olacaqmı və ya bəlkə də məlumatlar XML şəklində göndərilə daha yaxşı olar, və yaxud da onları Soap-da realizə etmək lazımdır? Şifrələnmə tətbiq edilsin, ya yox? Bu təqdimati mərhələdə olan sualları öz şəbəkə mübadiləsinin protokol stekinin tərtibatı zamanı həll etmək lazımdır.

**Tətbiqi (Application)** – bu OSI modelinin ən yuxarı mərhələsidir. Bu mərhələ yuxarı mərhələnin funksionallığını realizə edir: poçt ötürülməsi (SMTP), məsafəli serverdən (POP) poçt ismarıclarının qəbulu, faylların (FTP və TFTP) şəbəkə ilə ötürülməsi, veb-səhifələrinin (HTTP və HTTPS) baxılması, səsin VoIP (SIP) ilə ötürülməsi və digər standart protokollar. Proqramçı öz tətbiqinin tərtibatı zamanı trivial tətbiqlərin həlli üçün müvafiq protokola ciddi əməl etməlidir.

Əgər iş prosesi zamanı tətbiqi mərhələyə aid öz protokolunu tətbiq etməyə zəruriyyət yaranarsa (bu isə çox tez-tez yaranır), onda öz protokol serverinə və serverdən cavab sisteminə çox diqqətlə komanda sistemini tərtib etmək lazımdır.

Beləliklə, bu OSI modelinin bütün yeddi mərhələsidir. Gördüyünüz kimi, modeldə şəbəkə strukrunun proqram və aparat hissəsi ayrılmışdır. İlk iki mərhələ - şəbəkənin aparat vasitələriylə işləyən mərhələlərdir, onlar şəbəkənin topologiyasından, şəbəkə qurğularından aslıdır. Qalan beş yuxarı mərhələ şəbəkə qurulmasının texniki xüsusiyyətlərindən az aslıdırlar. Siz digər şəbəkə texnologiyalarına keçid edə bilərsiniz lakin, bu yuxarı mərhələlərin proqram vasitələrində heç bir dəyişiklik tələb etməyəcək.

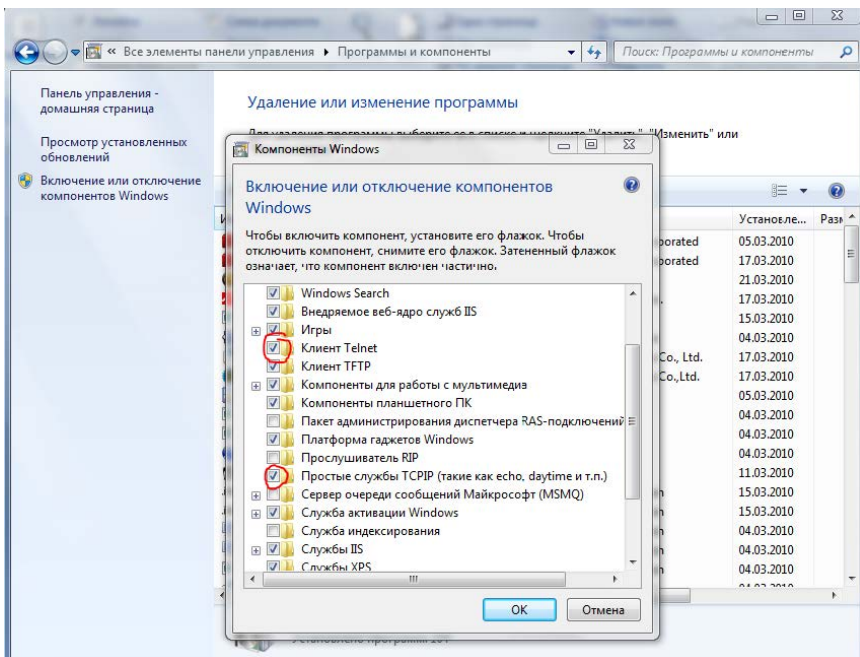
SMTP serveri ilə poçt ismarıcı yollanmasının sadə nümunəsini misal çəkək. Bunu, telnet proqramından istifadə edərək etmək olar. Bu çox lazımlı utilitidir, məsafəli qovşaqla TCP bağlantısını (nəqliyyat mərhələsi) qurmağa və ASCII sətiri ( təqdimat mərhələsi) şəklində komanda ötürülməsini (tədbiqi mərhələ) təmin etməyə imkan verir. Telnet müştəri SMTP serveri ilə bağlantını ( sessiya mərhələsi) - telnet smtp.mail.ru: - komandası köməylə inisializasiya edir:



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.
C:\Users\Alexandr>telnet smtp.mail.ru 25
```

burada smtp.mail.ru — qovşağın adı, 25 isə — SMTP serverinin standart TCP portudur.

Yeri gəlmişkən, Windows 7 pro sahibləri özlərinə Telnet müştərisini idarə panelinin “proqramlar və komponentlər” (программы и компоненты) -> windows komponentlərinin qoşulub, söndürülməsi (включение или отключение компонентов windows) yarağından quraşdırmalıdır. Telnet bizə şəbəkə tətbiqlərinin testi zamanı lazım olacaq, bundan əlavə, TCP/IP sadə xidmətlərini qoşun, onlar da bizə sadə müştəri tətbiqlərinin testlənməsi zamanı lazım olacaqlar:



Smtp serveri ilə bağlantının qurulmasından sonra server müştəriyə 220 kodu və özü haqda informasiya olan sətri

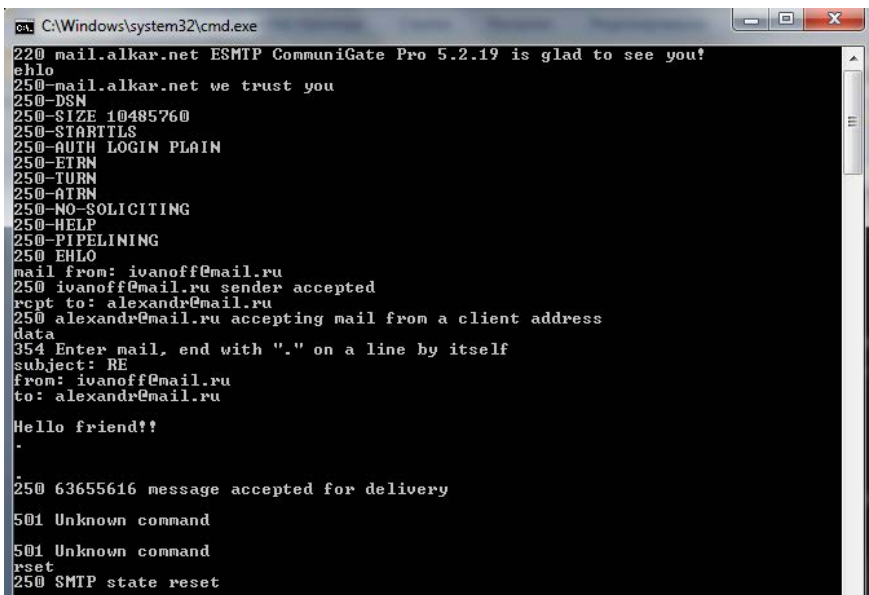
yollayır. Müştəri serverə salamlama yollayır “ehlo” (yəni, server müştəriyə birinci yollayır, müştəri isə serverə cavab verir – ünsiyyət qaydası sessiya mərhələsi ilə təyin olunur.)

Məlumat mübadiləsi ASCII yeddibitli kodlaşma sətiri şəklində baş verir, bu SMTP protokolunun təqdimat mərhələsiylə müəyyən olunur.

Serverə verilən komandalar və serverdən gələn cavablar tədbiqi mərhələ ilə müəyyən edilmiş və müvafiq RFC–821 (ESMTP üçün RFC–2821) sənəddə təsvir olunmuşdur.

Müvafiq sənədlərlə bu saytda tanış olmaq olar: <http://ietf.org>.

Beləliklə, təxmini olaraq bu cür:



```

C:\Windows\system32\cmd.exe
220 mail.alkar.net ESMTP CommuniGate Pro 5.2.19 is glad to see you!
ehlo
250-mail.alkar.net we trust you
250-DSN
250-SIZE 10485760
250-STARTTLS
250-AUTH LOGIN PLAIN
250-ETRN
250-TURN
250-ATRN
250-NO-SOLICITING
250-HELP
250-PIPELINING
250 EHLO
mail from: ivanoff@mail.ru
250 ivanoff@mail.ru sender accepted
rcpt to: alexandr@mail.ru
250 alexandr@mail.ru accepting mail from a client address
data
354 Enter mail, end with "." on a line by itself
subject: RE
from: ivanoff@mail.ru
to: alexandr@mail.ru
Hello friend!!
.
250 63655616 message accepted for delivery
501 Unknown command
501 Unknown command
rset
250 SMTP state reset

```

Düzdür, sonda biz məktubu yollamadıq, yoxsa ki ünvanını uydurduğumuz alan tərəf, tanımadığı ünvandan

məktub aldıqda təəccüblənəcəkdik. Bundan əlavə, smtp.mail.ru serveri auditifikasiya prosedurunu tələb edir, bu da yuxarıda göstərilən fragmentdə yoxdur. Və yaddan çıxarmayın ki, məktubun başlığında yollayanın IP adresi yazılacaqdır.

Windows NT ailəsi əməliyyat sistemlərində şəbəkə üzrə digər sistemlə qarşılıqlı əlaqə imkanı – 1983-cü ildə UNIX ailəsi əməliyyat sistemlərində şəbəkə qarşılıqlı əlaqəsinin realizəsinin sadələşdirilməsi üçün tətbiq edilən, uyğunlaşan və Berkeley Sockets kitabxanasının demək olar ki tam analoqu olan - soketlər tərəfindən təmin edilir.

SOCKET termini yuva kimi tərcümə olunur. (bu terminlə əl kommutasiyalı telefon stansiyalarında olan yuvaları adlandırırdılar).

Windows NT-dən başlayaraq, əməliyyat sistemlərində Windows Sockets kitabxanasının ikinci versiyası tətbiq olunur. Bu kitabxana WinAPI-dəki tərtibatçıya tam əl çatandır, lakin, burda idarəolunmaz kod mövcuddur. Təbii ki, .Net Framework yaradıcıları, tərtibatçıya şəbəkə tətbiqlərinin yazılması üçün rahat alət təqdim edərək, özündə idarəolunmaz kodu inkapsulasiya edən idarəedilən kitabxanayı yaratmışlar.

Bu alət, WinSock2 idarəolunmaz kitabxana və Berkeley Sockets UNIX sistemləri ilə tam uyğunlaşan olduğu üçün müxtəlif platformalarda qovşaqların fəaliyyət göstərə biləcəkləri bölüşdürülmüş sistemləri qurmağa imkan yaradır.



## 6. Socket class-ı

Şəbəkə ilə işləmək üçün class-lar System.Net və System.Net.Sockets ad fəzalarında yerləşirlər.

### **System.Net.Sockets.Socket class-ı**

Beləliklə, Socket classı - Microsoft.Net Framework platformasında Berkeley soketləri interfeysini realizə edən class-dır. MSDN – də göstərildiyi kimi, Socket, şəbəkə qarşılıqlı əlaqəsi üçün metod və xassə dəstinə malikdir. Socket class-ı ProtocolType sadalanmasında olan, istənilən kommunikasiya protokollarını istifadə edərək məlumatların ötürülmə və alınmasını icra etməyə imkan yaradır:

Üzvün adı	Təsvir
IP	Protokol IP.
Icmp	Protokol ICMP.
Igmp	Protokol IGMP.
Ggp	Protokol GGP.
IPv4	Protokol IPv4.
Tcp	Protokol TCP.
Pup	Protokol PUP.
Udp	Protokol UDP.
Idp	Protokol IDP.
IPv6	Protokol IPv6.
Raw	Protokol Raw IP.
Ipx	Protokol IPX.
Spx	Protokol SPX.
SpxII	Protokol SPXII.

Dəstəklənən protokolların tam siyahısını siz MSDN – də görə bilərsiniz.

Socket class-ı obyektinin təyin edilməsindən aslı olaraq, soketlər aktiv və passivlərə bölünürlər.

Aktiv soket – müştəri tərəfindən məsafəli serverlə əlaqə qurmaq üçün nəzərdə tutulub.

Passiv soket – server müştərisidir, müştərilərin onunla bağlanmasını gözləyir.

Soketin təyin olunmasından aslı olaraq (aktiv və ya passiv) onunla işin alqoritmi də dəyişir.

Bundan başqa sinxron və asinxron soket anlayışı da mövcuddur.

İş burasındadır ki, soketlərdən istifadə edərək ismaric mübadiləsinin təşkili zamanı biz bütün qoşulmuş müştərilərə xidmət göstərməyə imkan yaradan mexanizmi nəzərə almalıyıq. Bu problemin həlli üçün sinxron soketlərdən istifadə zamanı adətən platformanın çoxhəlli icrasının təminat mexanizmindən istifadə olunur.

Asinxron soketlərdə sinxronlardan fəqrlə olaraq daxilinə qurulmuş tədbirlərin çoxaxınlı emalı iştirak edir, bu asinxron rejimdə məlumat mübadiləsi etməyə imkan verir.

### **Müştəri tərəfindən TCP protokolunu istifadə edərək sinxron bağlantının qurulması:**

Müştəri tərəfindən bağlantını qurmaq üçün aşağıdakıları etmək lazımdır:

1. Socket tipli obyektə ona şəbəkənin tipini göstərərək yaratmaq (aşağıda göstərilən nümunədə AddressFamily.

InterNetwork (IPv4), nəqliyyat protokolu tipi SocketType.Stream (TCP) və ProtocolType)

2. Connect metodunu çağırmaq, parametr kimi ona qoşulmaq lazım olan məsafəli maşın və portun IP ünvanını inkapsulyasiya edən IPAddress class-ının obyektini ötürmək.

3. Uğurlu bağlantı nəticəsində ismarıc mübadiləsinə, tədbiqi, təqdimat və sessiya protokol mərhələlərinə uyğun olaraq, ismarıc yollamaq üçün Send və qəbul etmək üçün Receive metodlarını istifadə edərək başlamaq olar.

```

IPAddress ip=IPAddress.Parse("207.46.197.32");
EndPoint ep = new EndPoint(ip, 80);
Socket s = new Socket(AddressFamily.InterNetwork,
                      SocketType.Stream, ProtocolType.TP);

try{
    s.Connect(ep);
    if (s.Connected)
    {
        String strSend="GET\r\n\r\n";
        s.Send(System.Text.Encoding.ASCII.GetBytes(strSend));
        byte[] buffer = new byte[ 1024];
        int l;
        do
        {
            l = s.Receive(buffer);
            textBox1.Text +=
                System.Text.Encoding.ASCII.GetString(buffer, 0, l);
        } while (l > 0);
    }
    else
        MessageBox.Show("Error");
}
catch (SocketException ex)

    MessageBox.Show(ex.Message);
}

```

Yuxarıda göstərilən nümunədə biz 207.46.197.32 (Microsoft.com) məsafəli hostun 80 portuna (http tətbiqi protokolun standart portu) qoşulduq və ora standart GET

sorğusunu (HTTP1.0 spesifikasiyasına uyğun olaraq) yolladıq. Server sorğulanmış səhifəni cari olaraq qaytardı (HTTP1.0 protokol virtual hostlarla işi dəstəkləmir).

Diqqət yetirin ki, biz sətirləri ASCII kodlaşdırmasında bayt massivlərinə təkrar kodlaşdırmağa olduq, belə ki, ötürülən məlumatların formatı HTTP protokollarının təqdimat mərhələsində müəyyən olunur.

Socketlə işlədikdən sonra onu düzgün şəkildə bağlamaq lazımdır, buna görə sıra ilə məlumat ötürülməsinin bloklanması üçün Shutdown metodu çağırılır və socketlə istifadə olunan idarəli və idarəolunmayan resursların azad edilməsi üçün Close çağırılır.

Shutdown metodu SocketShutdown sadalaması parametrlər kimi qəbul edir, onun qiymətindən aslı olaraq ismarıqların sonrakı göndərilməsi və qəbul edilməsi qadağan edilir.

Socketlə işin düzgün başa çatdırılması üçün növbəti kod blokunu əlavə edirik:

```
finally {
    s.Shutdown(SocketShutdown.Both);
    s.Close();
}
```

**Server tərəfindən bağlantını, (TCP) bağlantısının sinxron rejimdə qurulması ilə qəbul edirik: (passiv socketlə sinxron rejimdə iş)**

Server tərəfində TCP bağlantını qəbul edən soketi yaratmaq və qoşulmuş müştəri ilə məlumat mübadiləsini təmin etmək üçün aşağıdakıları etmək lazımdır:

1. Socket tipli obyekt, ona şəbəkənin tipini göstərərək yaratmaq (aşağıda göstərilən nümunədə AddressFamily. InterNetwork (IPv4), nəqliyyat protokolu tipi SocketType. Stream (TCP) və ProtocolType).

2. Bind socketinin metodunu çağıraraq, alınmış socketi serverdəki IP ünvan və portla bağlamaq. Bind parametr kimi özündə müştərinin qoşulacağı IP ünvan və portu inkapsulyasiya edən IPEndPoint class-nın obyektini qəbul edir.

3. Listen metodunu, ona parametr kimi emal gözləyən bağlantıların növbəsinin ölçüsünü verərək, çağırmaq.

4. Dövrün işində Accept metodu çağırılır, onun çağırılması bu icra axınının blok edilməsidir. Accept metodu müştərinin qoşulduğu zaman axının blokunu götürür və onunla yaradılan yeni Socket obyektini qaytarır, bunun vasitəsilə məsafəli müştəri ilə ismarıq mübadiləsi baş verir. Burada yaradılmış socketin nömrəsi dinləmə keçirilən portun nömrəsindən fərqlənir.

5. Accept metodunun alındığı socket vasitəsilə ismarıq mübadiləsinin sona çatdıqdan sonra bu socket bağlanır və Accept yenidən yeni müştərinin qoşulması üçün çağırılır.

Aşağıda göstərilən nümunədə IP 127.0.0.1 də 1024 portuna qoşulma gözləntisi baş verir, müştəri qoşulduqdan sonra ona cari vaxt və tarix olan sətir atılır, konsol pəncərəsinə isə müştərinin IP ünvanı və məsafəli müştərinin port nömrəsi çıxarılır:

```

Socket s = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.TCP);

IPAddress ip = IPAddress.Parse("127.0.0.1");
IPEndPoint ep = new IPEndPoint(ip, 1024);

s.Bind(ep);
s.Listen(10);

try
{
    while (true)
    {
        Socket ns = s.Accept();

        Console.WriteLine(ns.RemoteEndPoint.ToString());

        ns.Send(System.Text.Encoding.ASCII.GetBytes(DateTime.Now.ToString()));
        ns.Shutdown(SocketShutdown.Both);
        ns.Close();
    }
}
catch (SocketException ex)
{
    Console.WriteLine(ex.Message);
}

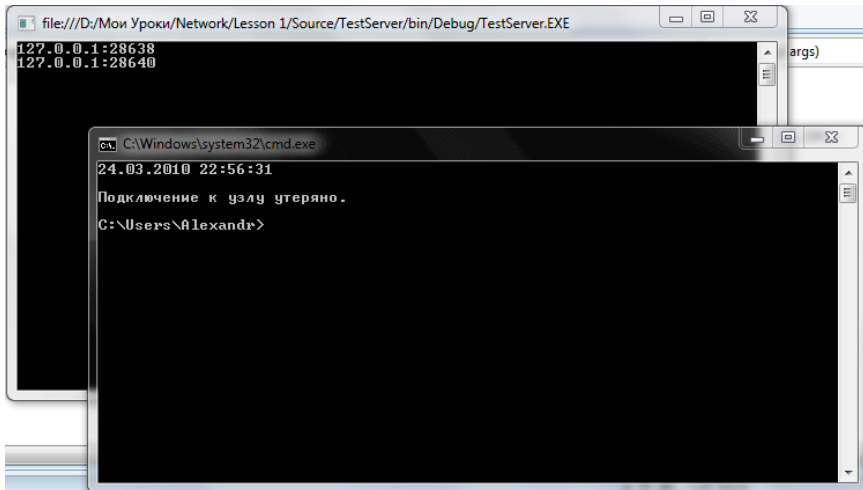
```

Bu proqramın işini telnet utilinin köməyilə, bağlantını localhost (127.0.0.1) 1024 portuna qoşulmanı icra edərək yoxlamaq olar.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.
C:\Users\Alexandr>telnet localhost 1024
```

İşlək serverin və müştərinin işinə nümunə aşağıdakı kimidir.



```
file:///D:/Мои Уроки/Network/Lesson 1/Source/TestServer/bin/Debug/TestServer.EXE
127.0.0.1:28638
127.0.0.1:28640

C:\Windows\system32\cmd.exe
24.03.2010 22:56:31
Подключение к узлу утеряно.
C:\Users\Alexandr>
```

Təbii ki, Bind metodunda sizin şəbəkə interfeysinizin IP ünvanını göstərsəz, onda qoşulma da bu ünvandan istifadə edərək olmalıdır.

## 7. Asinxron soketlər

Müştəri və server arasında yuxarıda tətbiq edilmiş belə sadə qarşılıqlı əlaqə metodunun yaşamaq hüququ vardır, lakin onun da qüsurları vardır. Məsələn müştəriyə serverlə bağlantını kifayət qədər uzatmaq lazımdırsa, onda müştəri soketi azad etməyincə, serverə başqa heç kim qoşula bilməz.

Bu vəziyyətdən çıxış yolu vardır. Çoxaxınlı emalı tətbiq etmək lazımdır, yəni hər müştəri ilə qarşılıqlı əlaqəni ayrıca icra axınında etmək lazımdır. Əlbəttə, çoxaxınlı emalı aşağıda göstərildiyi kimi əllə də etmək olar:

```
class Server
{
    delegate void ConnectDelegate (Socket s);
    delegate void StartNetwork(Socket s);

    Socket socket;
    IPEndPoint endPoint;

    public Server(string strAddr, int port)
    {
        endPoint = new IPEndPoint(IPAddress.Parse(strAddr), port);
    }
    void Server_Connect(Socket s)
    {
        s.Send(System.Text.Encoding.ASCII.GetBytes(DateTime.Now.ToString()));
        s.Shutdown(SocketShutdown.Both);
        s.Close();
    }
    void Server_Begin(Socket s)
    {
        while (true)
        {
```



```

        try
        {
            while (s!=null)
            {
                Socket ns = s.Accept();
                Console.WriteLine(ns.RemoteEndPoint.ToString());
                ConnectDelegate cd = new ConnectDelegate(Server_Connect);
                cd.BeginInvoke(ns, null, null);
            }
        }
        catch (SocketException ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
}

public void Start()
{
    if (socket != null)
        return;
    socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.IP);
    socket.Bind(endP);
    socket.Listen(10);
    StartNetwork start = new StartNetwork(Server_Begin);
    start.BeginInvoke(socket, null, null);
}

public void Stop()
{
    if (socket != null)
    {
        try
        {
            socket.Shutdown(SocketShutdown.Both);
            socket.Close();
            socket = null;
        }
        catch (SocketException ex)
        {
        }
    }
}
}

class Program
{
    static void Main(string[] args)
    {
        Server s = new Server("127.0.0.1", 1024);
        s.Start();
        Console.Read();
        s.Stop();
    }
}

```

Göstərilmiş nümunədə müştəri tərəfindən qoşulmanın icra edilməsi zamanı, müştəri ilə əlaqənin emalı üçün nümayəndə yaradılır. Bu nümayəndə asinxron işə qoşulur, bu da ayrıca axında qoşulmuş müştəri ilə qarşılıqlı əlaqəni təmin edir. Dinləyici socketin özü isə ayrıca axında qoşulmuşdur.

Sizə yuxarıda göstərilən kod çox mürəkkəb görünmədi ki? Ümid edirəm ki, yox. İş burasındadır ki, socketlər əlavə kod yazılmasa belə özü özlüyündə asinxron işləməyi bacarırlar.

Socket class-ında asinxron işin təmin edilməsi üçün müvafiq metodlar dəsti nəzərdə tutulmuşdur:

Adı	Təsviri
AcceptAsync	Daxil olan bağlantını qəbul etmək imkanı üçün asinxron əməliyyata başlayır.
BeginAccept	Yükənib. Məsafəli qovşağa qoşulmaq üçün məsafəli sorğunun icrasına başlayır.
BeginConnect	Yükənib. Məsafəli qovşağa qoşulmaq üçün məsafəli sorğunun icrasına başlayır.
BeginDisconnect	Məsafəli son nöqtədən söndürülmə üçün asinxron sorğunun icrasına başlayır.
BeginReceive	Yükənib. Qoşulmuş Socket obyektindən məlumatların asinxron qəbulunun icrasına başlayır.

BeginReceiveFrom	Göstərilmiş şəbəkə qurğusundan məlumatların asinxron icrasına başlayır.
BeginReceiveMessageFrom	Verilmiş SocketFlags obyektini istifadə edərək, məlumat baytlarının verilmiş sayının, məlumat buferinin göstərilmiş yerinə asinxron qəbuluna başlayır, həmçinin, paketin son nöqtənisini və informasiyasını saxlayır.
BeginSend	Qoşulmuş Socket obyektinə məlumatların ötürülməsinə başlayır.
BeginSendFile	Çox yüklənib. Qoşulmuş Socket obyektinə məlumatların ötürülməsinə başlayır.
BeginSendTo	Göstərilmiş məsafəli qovşağa məlumatların asinxron ötürülməsini icra edir.
ConnectAsync	Məsafəli qovşağa qoşulmaq üçün asinxron sorğunun icrasına başlayır.
DisconnectAsync	Məsafəli son nöqtədən kəsilmək üçün asinxron sorğunun icrasına başlayır.
EndAccept	Çox yüklənib. Daxil olan bağlantı cəhdini qəbul edir.
EndConnect	Qoşulmaq üçün, gözləyən asinxron sorğunu sona çatdırır.
EndDisconnect	Kəsilmək üçün, gözləyən asinxron sorğunu sona çatdırır.
EndReceive	Çox yüklənib. Kənara qoyulmuş asinxron oxumanı sona çatdırır.
EndReceiveFrom	Kənara qoyulmuş asinxron oxumanı müəyyən olunmuş son nöqtədən sona çatdırır.
EndReceiveMessageFrom	Kənara qoyulmuş asinxron oxumanı müəyyən olunmuş son nöqtədən sona çatdırır. Bu metod da həmçinin paket haqqında informasiyanı EndReceiveFrom metodundan daha çox göstərir.

EndSend	Çox yüklənib. Asinxron ötürülmənin kənara qoyulmuş əməliyyatını sona çatdırır.
EndSendFile	Fayl ötürülməsinin kənara qoyulmuş əməliyyatını sona çatdırır.
EndSendTo	Bir qırağa qoyulmuş müəyyən edilmiş yerə asinxron yollama əməliyyatını sona çatdırır.
ReceiveAsync	Qoşulmuş Socket obyektindən məlumatları almaq üçün asinxron sorğunu icrasına başlayır.
ReceiveFrom	Çox yüklənib. Dataqramm qəbul edir və mənbənin son nöqtəsini yadda saxlayır.
ReceiveFromAsync	Göstərilmiş şəbəkə qurğusundan məlumatların asinxron qəbulunun icrasına başlayır.
ReceiveMessageFrom	Göstərilmiş bayt sayını, verilmiş SocketFlags obyektı vasitəsi ilə verilənlər buferinin göstərilən yerinə qəbul edir, həmçinin son nöqtəni və paket informasiyasını yadda saxlayır.
ReceiveMessageFromAsync	Verilmiş SocketAsyncEventArgs.SocketFlags obyektindən istifadə edərək, baytların verilmiş sayının göstərilmiş məlumat buferinə asinxron qəbuluna başlayır, həmçinin paketin son nöqtəsini və informasiyasını saxlayır.
SendAsync	Qoşulmuş Socket obyektinə məlumatların asinxron ötürülməsinə başlayır.
SendPacketsAsync	Qoşulmuş Socket obyektinə yaddaşda fayl dəsti və ya məlumat buferinin asinxron ötürülməsini icra edir.
SendToAsync	Göstərilmiş qovşağa məlumatların asinxron ötürülməsini icra edir.

## **Server tərəfindən bağlantını (TCP) bağlantısının asinxron rejimdə qurulması ilə qəbul edirik: (passiv soketlə asinxron rejimdə iş)**

Server tərəfində TCP bağlantını qəbul edən soketi yaratmaq və qoşulmuş müştəri ilə asinxron rejimdə məlumat mübadiləsini təmin etmək üçün aşağıdakıları etmək lazımdır:

1. Socket tipli obyektini ona şəbəkənin tipini göstərərək yaratmaq (AddressFamily.InterNetwork (IPv4) aşağıda göstərilən nümunədə, nəqliyyat protokolu tipi SocketType.Stream (TCP) və ProtocolType.IP).

2. Bind soketinin metodunu çağıraraq, alınmış soketi serverdəki IP ünvan və portla bağlamaq. Bind parametr kimi özündə müştərinin qoşulacağı IP ünvan və portu inkapsulyasiya edən IPAddress class-ının obyektini qəbul edir.

3. Listen metodunu, ona parametr kimi emal gözləyən bağlantıların növbəsinin ölçüsünü verərək, çağırmaq. İlk üç bənd tam olaraq sinxron rejimdə işin alqoritminə uyğun gəlir, ardınca dəyişikliklər olacaq.

4. AsyncCallback tipli nümayəndə yaratmaq lazımdır və parametr kimi nümayəndə və dinləyən soketi verərək BeginAccept metodunu çağırmaq lazımdır.

5. Müştərinin qoşulduğunda nümayəndə çağırılacaq, ona IAsyncResult parameter tipi AsyncState xassəsində bizim dinləyici soketimiz gələcək.

6. Qəbul edilmiş soketdə EndAccept metodu çağırılır, o Socketin yeni obyektini qaytarır, onun vasitəsilə məsafəli müştəri ilə ismarıq mübadiləsi keçir.

## 7. Yenidən BeginAccept çağırılır, iş dövrü təkrarlanır.

Analoji olaraq, imsarıcıların müştəriyə həm də asinxron göndərilməsini BeginSend metodunu istifadə edərək təşkil etmək olar, bunun üçün BeginSend-ə yollama əməliyyatının sona çatması zamanı çağırılan nümayəndəni və müştəri ilə məlumat mübadiləsi socketini ötürmək lazımdır.

Aşağıda göstərilmiş misal bağlantının asinxron rejimdə dinlənməsini və ismarıcın müştəriyə asinxron göndərilməsini işə salır:

```
class AsyncServer
{
    IPEndPoint endPoint;
    Socket socket;

    public AsyncServer(string strAddr, int port)
    {
        endPoint = new IPEndPoint(IPAddress.Parse(strAddr), port);
    }
    void MyAcceptCallbackFunction(IAsyncResult ia)
    {
        //dinləyici socketə keçidi alırıq
        Socket socket=(Socket) ia.AsyncState;
        //müştəri ilə məlumat mübadiləsi üçün socket alırıq
        Socket ns = socket.EndAccept(ia);

        //konsola bağlantı barədə informasiyanı çıxarıyıq
        Console.WriteLine(ns.RemoteEndPoint.ToString());

        //cari vaxtı müştəriyə asinxron yollayırıq,
        //göndərmə əməliyyatının sona çatması zamanı metod çağırılacaq
        //MySendCallbackFunction
        byte[] sendBufer =
            System.Text.Encoding.ASCII.GetBytes(DateTime.Now.ToString());

        ns.BeginSend(sendBufer, 0, sendBufer.Length, SocketFlags.None,
            new AsyncCallback(MySendCallbackFunction), ns);

        //asinxron Accept yeniləyirik
        socket.BeginAccept(new AsyncCallback(MyAcceptCallbackFunction),
            socket);
    }
    void MySendCallbackFunction(IAsyncResult ia)
```

```

{
    // müştəriyə məlumatların göndərilməsinin sona çatmasında
    // soketi bağlayırıq (əgər bizə məlumatlarla sonrakı mübadilə lazım olsa,
    // biz onu burada təşkil edə bilərdik)
    Socket ns = (Socket) ia.AsyncState;
    int n = ((Socket) ia.AsyncState).EndSend(ia);
    ns.Shutdown(SocketShutdown.Send);
    ns.Close();
}

public void StartServer()
{
    if (socket != null)
        return;
    socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
        ProtocolType.IP);
    socket.Bind(endP);
    socket.Listen(10);
    // asinxron Accept başlayırıq, müştərinin qoşulduğu zaman
    // əməli MyAcceptCallbackFunction çağırılacaq
    socket.BeginAccept(new AsyncCallback(MyAcceptCallbackFunction),
        socket);
}
}

class Program
{
    static void Main(string[] args)
    {
        AsyncServer server = new AsyncServer("127.0.0.1", 1024);
        server.StartServer();
        Console.Read();
    }
}

```

Müştəri tərəfində TCP socketin asinxron proqramlaşdırılması, müştəri tərəfindən asinxron rejiminin yaradılması üçün BeginConnect metodunun istifadəsi ilə icra edilə bilər. Bundan əlavə, müştəri tərəfində ismarıqların asinxron göndərilməsi və qəbulunu eynən serverdə proqramlaşdırma kimi istifadə etmək olar.

**Soket class-ının sinxron rejimdə UDP protokolu üzrə istifadəsi**

Əgər, UDP protokolu istifadə olunursa, yəni bağlantının qoşulması olmayan protokol, onda ismarıcı yollanması üçün SendTo metodundan istifadə etmək, deyaqramların alınması üçün ReceiveFrom metodlarını tətbiq etmək olar.

UDP dinləyici soketlə iş alqoritmi növbətidir:

1. Soket tipli obyektə ona şəbəkənin tipini göstərərək yaratmaq (aşağıda göstərilən nümunədə AddressFamily.InterNetwork

(IPv4), nəqliyyat protokolu tipi SocketType.Dgram (UDP) ı ProtocolType.IP);

```
socket=new Socket(AddressFamily.InterNetwork,
    SocketType.Dgram,
    ProtocolType.IP);
```

2. Alınmış soketi serverdəki IP ünvan və serverdəki portla Bind soketin metodunu çağıraraq bir-biriylə bağlamaq. Bind, parametrlər kimi müştərilərin qoşulacağı serverin IP ünvan və portunu özündə inkapsulyasiya edən IPEndPoint class-ının obyektini qəbul edir.

```
socket.Bind(new IPEndPoint(IPAddress.Parse("10.2.21.129"), 100));
```

3. Soketdə ReceiveFrom metodunu çağırmaq, burada cari axının icrası, giriş buferində məlumatların peyda olmasınadək dayandırılacaqdır. Əgər oxuma üçün buferin ölçüləri kifayət qədər olmayacaqsa, onda SocketException – un çıxarılması yarana bilər;



```
int l=rs.ReceiveFrom(buffer, ref ep);
String strClientIP=((IPEndPoint)ep).Address.ToString();
String str = String.Format("\nПолучено от {0}\r\n{1}\r\n" ,
    strClientIP, System.Text.Encoding.Unicode.GetString(buffer, 0, 1));
```

4. Məlumatların ötürülməsi üçün SendTo metodundan istifadə olunur (bu metodun təsvirini, müştəri UDP soketinin təhlili zamanı verəcəm);

Belə ki, ReceiveFrom çağırılması blok edəndir, onunla işi ayrıca icra axınına aparmaq olar, aşağıdakı nümunədə göstərildiyi kimi:

```
delegate void AddTextDelegate(String text);
System.Threading.Thread thread;
Socket socket;

public Form1 ()
{
    InitializeComponent();
}
void AddText(String text)
{
    textBox1.Text+=text;
}
void RecivFunction(object obj)
{
    Socket rs=(Socket) obj;

    byte[] buffer = new byte[ 1024];
    do
    {
        EndPoint ep = new IPEndPoint(0x7f000000, 100);
        int l=rs.ReceiveFrom(buffer, ref ep);
        String strClientIP= ((IPEndPoint)ep).Address.ToString();
        String str = String.Format("\n {0}\r\n{1}\r\n - dan alındı"
            strClientIP, System.Text.Encoding.Unicode.GetString(buffer, 0, 1));
        textBox1.BeginInvoke(new AddTextDelegate(AddText), str);
    } while (true);
}
private void button1_Click(object sender, EventArgs e)
{
    if (socket != null &&thread!=null)
        return;
```

```

socket=new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.IP);
socket.Bind(new IPEndPoint(IPAddress.Parse("10.2.21.129"), 100));

thread = new System.Threading.Thread(RecvFunction);
thread.Start(socket);

}
private void button2_Click(object sender, EventArgs e)
{
    if (socket != null)
    {
        thread.Abort();
        thread = null;
        socket.Shutdown(SocketShutdown.Receive);
        socket.Close();
        socket = null;
        textBox1.Text = " ";
    }
}

```

İsmarıcların UDP protokolu üzrə göndərilməsi heç bir sual yaratmır:

```

private void button4_Click(object sender, EventArgs e)
{
    Socket socket = new Socket(AddressFamily.InterNetwork,
        SocketType.Dgram, ProtocolType.IP);

    socket.SendTo(System.Text.Encoding.Unicode.GetBytes(textBox2.Text),
        new IPEndPoint(IPAddress.Parse("10.2.21.259"), 100));

    socket.Shutdown(SocketShutdown.Send);
    socket.Close();
}

```

Burda biz sadəcə soket yaradırıq və sinxron ismarıcı məqsədli IP ünvanına SendTo istifadəsi ilə göndəririk. Diqqət yetirin ki, məqsədli qovşaq kimi mən öz alt şəbəkəmin genişlənmə ünvanını göstərmişəm, yəni, mənim ismarıclarım alt şəbəkələrin bütün qovşaqlarına ünvanlanıb.

**UDP ilə asinxron rejimdə iş**

İsमारıcıların sinxron göndərilməsindən başqa, UDP soket həm də asinxron rejimdə işi dəstəkləyir.

UDP soketdən asinxron oxuma üçün `BeginReceiveFrom` metodundan istifadə olunur, asinxron göndərmə üçün isə `BeginSendTo`.

Aşağıda mən UDP soket vasitəsilə isमारıcıların asinxron qəbuluna misal göstərəcəm:

```
private void button1_Click(object sender, EventArgs e)
{
    if (socket != null)
        return;
    socket=new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.IP);
    socket.Bind(new IPEndPoint(IPAddress.Parse("10.2.21.129"), 100));

    state.workSocket = socket;
    RcptRes = socket.BeginReceiveFrom(state.buffer,
        0,
        StateObject.BufferSize,
        SocketFlags.None,
        ref ClientEP,
        new AsyncCallback(Receive_Completed), state);
}
void Receive_Completed(IAsyncResult ia)
{
    try
    {
        StateObject so = (StateObject)ia.AsyncState;
        Socket client = so.workSocket;
        if (socket == null)
            return;
        int readed = client.EndReceiveFrom(RcptRes, ref ClientEP);

        String strClientIP = ((IPEndPoint)ClientEP).Address.ToString();
        String str = String.Format("\n {0}\r\n{1}\r\n -dan alındı",
            strClientIP, System.Text.Encoding.Unicode.GetString(so.buffer, 0, readed));

        textBox1.BeginInvoke(new AddTextDelegate(AddText), str);

        RcptRes = socket.BeginReceiveFrom(state.buffer,
            0,
            StateObject.BufferSize,
```

```

        SocketFlags.None,
        ref ClientEP,
        new AsyncCallback(Receive_Completed),
        state);
    }
    catch (SocketException ex)
    {
    }
}

```

**Asinxron ismarıcların UDP soket vasitəsilə göndərilməsi də zəhmət tələb etmir:**

```

private void button4_Click(object sender, EventArgs e)
{
    Socket socket = new Socket(AddressFamily.InterNetwork,
        SocketType.Dgram,
        ProtocolType.IP);

    byte[] buffer=System.Text.Encoding.Unicode.GetBytes(textBox2.Text);
    SendRes = socket.BeginSendTo(buffer, 0, buffer.Count(),
        SocketFlags.None,
        (EndPoint)new IPEndPoint(IPAddress.Parse("10.2.21.259"), 100),
        new AsyncCallback(Send_Completed),
        socket);
}

void Send_Completed(IAsyncResult ia)
{
    Socket socket = (Socket)ia.AsyncState;
    socket.EndSend(SendRes);
    socket.Shutdown(SocketShutdown.Send);
    socket.Close();
}

```

Və sonda, biz qovşaqların adına IP ünvanla yox, qovşağın DNS adı üzrə müraciət etməyə öyrəşmişik. Qovşağın IP ünvana icazəsi üçün (və ya əksinə) System.Net.Dns statik class-ından istifadə edə bilərik, onunla mütləq MSDN-də tanış olacaqsınız.

Bu GetHostAddresses class-ının metodu verilmiş domen adıyla bağlı olan IP massivi qaytarır.

## 8. Ev tapşırığı

---

1. Şəbəkə saatlarını deytaqram protokolundan istifadə edərək yazmaq. Serverdə lokal şəbəkəyə cari tarix barədə informasiya olan, paket göndərmə xidməti qoşulur. Müştərilər cari tarixi təsvir edir. Zəng sistemlərini də yarada bilərsiniz.

2. Müştərilər arasında ismarıc mübadiləsi sistemini yazın. Sistem, müştərilərin ona qoşulduğu və onda ismarıc qoyduqları mərkəzi serveri istifadə etməlidir. İsmarıqların ünvanlandığı müştəri onları serverdən sorğu üzrə alır. Sistem (TCP) bağlantıya yönümlü protokolu istifadə etməlidir.

