Dağıtık Sistem Nedir ?

Ağ üzerindeki bilgisayarlarda bulunan donanım veya yazılım bileşenlerinin yalnız mesaj göndererek haberleştikleri sistem. [Coulouris]

Dağıtık sistem, farklı bilgisayarlardaki donanım ve yazılım bileşenleri arasında haberleşme ve koordinasyonunun sadece mesajlaşma yoluyla sağlanabildiği ağ olarak tanımlanır.

Dağıtık sistemi diğer bilgisayar ağlarından farklı kılan özelliğin, ağın varlığı ve işlevinin kullanıcıya görünmemesi olduğu söylenebilir. Örneğin kullanıcı bir program çalıştırdığında, en iyi işlemciyi seçmek, giriş dosyalarını bu işlemciye aktarmak ve dönen sonuçları uygun yere koymak işlemcinin yürüttüğü işlerdir. Çok işlemcili bir dağıtık sistemde çalışmakta olmasına rağmen, kullanıcıya görünen ise tek bir sanal işlemcidir. Diğer ağlarda, kullanıcının kendisi bir makineye bağlanır, iş isteğinde bulunur, dosya gönderir vs. Diğer bir deyişle, ağ yönetim işlemleri kullanıcı tarafından yürütülür.

Dağıtık sistem, bir ağ üzerine kuruludur. Yazılımla, ağdaki bileşenler arası işlevsel uyumluluk ve kullanıcı açısından saydamlık sağlanır. Dolayısıyla, dağıtık sistemin diğer ağlardan farkı donanım değil, yazılım katmanından – özellikle işletim sisteminden- kaynaklanmaktadır.

Dağıtık sistemlerin var oluş amacı, kaynakları paylaşmaya duyulan gerekliliktir. Bu kaynaklar donanımsal bileşenler (disk, yazıcı) olabileceği gibi, dosyalar, veri tabanı, nesneler gibi yazılım varlıkları da olabilir.

Bu büyük ağ da bulunan donanımlar kullanıcıya tek bir bilgisayar gibi davranır ve en iyi performansı sağlamayı amaçlar.

1. İzole değildir.
2. Cloud un temelini oluşturur
3. Senkron ve replike’dir
4. Yedekli çalışır

Dağıtık Mimari Yapı Gereksinimleri

Dağıtık mimarinin yapılandırılması çalışabilmesi için çeşitli bileşenlerin birbiriyle uyumlu ve senkron şekilde çalışılabilmesi gerekmemektedir.

1. İşletim sistemleri
2. Donanım mimarileri
3. İletişim mimarileri
4. Programlama dilleri
5. Yazılım ara yüzleri
6. Güvenlik ölçüleri
7. Bilgi gösterimleri

Dağıtık sistem

Avantajları	**Dezavantajları**
Maliyet	Merkezi yapı zorunluğu
Erişim kolaylığı – hız performans	Güvenlik
Hesaplama ve depo alanı	Her dilde yazım desteği yok
Ölçeklenebilir	İletişim
Güvenilir(Reliable)	
Dağıtım kolaylığı (Web)	
Güvenlik	

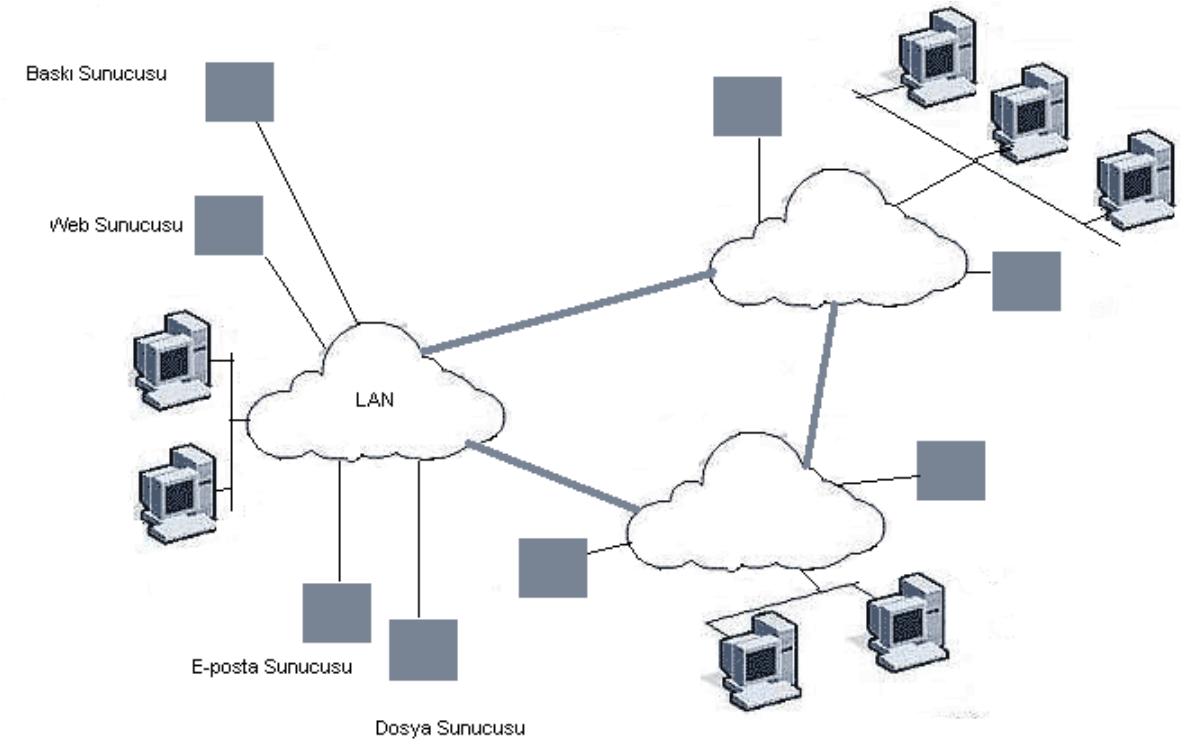
Şeffaflık: Kaynaklar ve işlemlerin ağ üzerinde dağıtık olduğu açık değildir
Açıklık: Servislerin sentaks ve semantikleri ile ilgili standart kurallar
Ölçeklenebilirlik: Daha fazla kullanıcı/kaynak, coğrafya, yönetim

Dağıtık Sistem Örnekleri

Internet

Internet, geniş bir dağıtık sistemdir. Internet kullanıcıları WWW, e-posta, dosya transferi servislerinden yararlanırlar.

İç Ağlar (Intranet)



Şekil 1. İç Ağ

Şekil 1'de, birkaç yerel ağdan oluşan bir iç ağ görülmektedir. İç ağlarda kaynak paylaşımı sağlayan ve sıklıkla kullanılmakta olan sunucular, şekilde gösterilmiştir. (Her yerel ağ, tüm bu sunucuları içermeyebilir.)

Dağıtık Sistem Güvenliği

Bir dağıtık sistemin güvenliği, süreçleri ve aralarındaki etkileşim için kullanılan kanalları güvenli hale getirmek ve süreçlerin kapçıkladığı nesneleri izinsiz erişime karşı korumakla sağlanır.

Servisin Reddi (Denial of Service)

Ddos atak olarak bilinen bu atak günümüzde bir çok bilişim sisteminin, büyük kapasiteli devlet ve banka ağlarının kilitlenmesine kadar gidebilmektedir. Sunuculara sürekli istek göndermek suretiyle sunucunun hizmet kapasitesinin doldurulup kilitlenmesidir. Ağa çok fazla mesaj gönderilerek fiziksel kaynakların (bant genişliği gibi) tüketilmesi, istemcilerin hizmet almasının engellenmesi veya yavaşlatılması da bu tip bir saldırıdır. İyi huylu kullanıcı ile kötü huylu ayırt edilemezse, sistemin çökmesi ile sonuçlanır. Çok büyük bant genişliği ve firewall maliyetlerini yanında getirmektedir.

Gezici Kod (Mbile Code)

En yaygın gezici kod örneği, applet'lerdir. Bir tarayıcı çalıştırmış olan kullanıcı, kodu web sunucusunda tutulan bir appletin linkini seçerse, kod kullanıcının makinesine indirilerek çalıştırılır; bu makinedeki kaynaklara erişir.
Gezici kod, sunucuların yükünü azaltmak açısından avantajlıdır. Ancak kullanıcının kaynaklarına zarar verecek şekilde kötüye kullanılabileceğinden, bir güvenlik tehdididir.

Dağıtık Sistemlerin Kullanım Amaçları

Kullanıcıya yönelik (neden dağıtık sistem?)
Kullanıcı ve kaynakların kolayca bağlanması
Şeffaflığın sağlanması
Teknik hedefler (nasıl başarılır?)
Açık olarak
Ölçeklenebilir olarak
Bu hedeflere bakmak şu soruyu cevaplamaya yardımcı olur:
"dağıtık bir sistem kurmak, uğraşmaya değer mi?"

Dağıtık Sistemlerin Gerekliliği

İşlevsel ayrım:
- Kapasite ve amaç bakımından farklı bilgisayarların varlığı:
  - İstemciler ve Sunucular
  - Veri toplama ve veri işleme

Doğal dağıtıklık:
- Bilgi:
  - Farklı bilgiler farklı kişiler tarafından oluşturulup kontrol edilir (Web sayfaları gibi)
- İnsan:
  - Bilgisayar destekli işbirliği (sanal ekipler, mühendislik, sanal ameliyat)
- Süpermarket zincirleri için perakende ve stok sistemleri

Güç dengesizliği ve yük çeşitliliği:
- İşlem yükünün farklı bilgisayarlar arasında dağıtılması.

Güvenilirlik:
- Farklı yerlerde uzun süreli koruma ve veri yedekleme (çoğaltma).

Ekonomi:
- Birçok kullanıcı arasında yazıcı paylaşımı ile sahip olma masrafının azaltılması.
- Bir bilgisayar ağından süper-bilgisayar elde etme.

Dağıtık sistemlerdeki bilgisayarlar ayrı kıtalar üzerinde, aynı bina veya aynı oda içerisinde bulunabilir. DS'in getirdikleri:

Birlikte ve birbirinden bağımsız çalışan sistemler

İşlerini birbirinden bağımsız yaparlar

Aynı zamanda program çalıştırır, bütün bir işleme kaynağı gibi görünür, birlikte çalışırlar

İşlemler mesaj alışverişiyle anlaşırlar.

Heterojen (çeşitlilik, farklılık): networks, hw, os, PLan

Ortak saat yok: Bilgisayarlar saatlerini kısıtlı derecede senkronize edebilir

Bağımsız bozulma: biri bozulsa da diğerleri çalışmaya devam eder.

Paralel işlemler

Birlikte çalışan bağımsız bileşenler

Mesaj gönderme ile iletişim

Ortak bellek yok (No shared memory)

Kaynak paylaşımı

Yazıcı, veri tabanı, diğer servisler

Genel sistem durumu belirli değil (No global state)

Hiçbir işlem, sistemin genel durumuna dair bir bilgi edinemez Ortak saat yok

İşlemler için sadece kısıtlı derecede saat eşitleme mümkün olabilir.

1. Kullanıcılarla kaynakları birleştirmek: Erişim servisleri uygulama çalıştırır

Belirli kaynaklar:

Yazıcılar, bilgisayarlar, işleme gücü, veri

Neden paylaşım?

Ekonomi

İşbirliği, Bilgi değiş-tokuşu (grup çalışması)

Paylaşım problemleri

Güvenlik

İstenmeyen işbirlikleri

2. Şeffaflık: Kaynaklar ve işlemlerin ağ üzerinde dağıtık olduğu açık değildir
3. Açıklık: Servislerin sentaks ve semantikleri ile ilgili standart kurallar
4. Ölçeklenebilirlik: Daha fazla kullanıcı/kaynak, coğrafya, yönetim
5. Uzun süre bozulmadan çalışma

Model Yönü İletişim İletişim nasıl sağlanır

– senkron iletişim

Mesaj gönderme ve alma ¨Eş zamanlı¨

– asenkron iletişim

Gönderme ve alma bağlı değil (Buffer, Kanallar, vb.)

Mesajlar sınırlı bir zamanla iletilirler ya da maksimum bir zaman yoktur.
Mesajlar mutlaka bir zamanda iletilirler ya da kayıp vardır.

– FIFO ya da not FIFO

Bir kanaldan önce gönderilen mesajın önce alınıp alınamayacağının kararı

– Noktadan-noktaya veya Yayın(Broadcast)

Bir mesaj bir anda sadece bir sürece veya daha fazlasına iletilebilir.

– Ne gönderilebilir

Sadece değerler,

referanslar

ve aktif elemanlar(Süreçler) gönderilebilir.

Multiprocessor (çok işlemcili) sistemler: DS değil

Shared memory (ortak bellek)

Bus-based interconnection network (kablolu bağlantı)

Örnek: İki veya daha fazla CPU'lu SMP'ler (symmetric multiprocessor)

Çoklu bilgisayar sistemleri: DS değil

No shared memory (ortak bellek yok)

Homojen donanım ve yazılım

Massively Parallel Processors (MPP)

Tightly coupled high-speed network

PC/Workstation clusters

High-speed networks/switches based connection.

Internet: birbirine bağlı çeşitli bilgisayar ağları.

Uygulamalar iletişimi mesajlarla sağlar

Dağıtık sistem kullanıcıları www, email, dosya gönderme gibi servislerden faydalanır

Intranet: bir organizasyon tarafından yönetilen alt ağ. Yerel güvenlik politikasıyla sınırlıdır.

Uydu bağlantısı, fiber-optik kablolar gibi hızlı iletim kapasitesi olan ağ altyapısı ile bağlıdırlar

Kablosuz ağlar

Genel internet uygulamaları ve servisleri:
Müzik, radyo, TV kanalları, video konferans erişimi için multimedia servisleri, ve çoklu kullanıcı desteği.

Internet, çeşitli servisler barındıran birçok farklı bilgisayar türlerinden oluşan büyük bir ağ topluluğudur.
Genel bir intranet;
Internetin bir alt parçası olup, ayrıca yönetilir ve kaynakların ağ içerisinde paylaşımını destekler (dosya/depolama sistemleri ve yazıcılar)

Dağıtık Sistem Ürün Ortamı Sorun ve Çözümleri

Online kitapçı (World Wide Web üzerinde)

Müşteriler kendi bilgisayarlarıyla sizin bilgisayarınıza (web sunucunuza) bağlanabilir:

Stokunuza bakabilir

Sipariş verebilir

…

Ya . . . ?

Müşteriniz çok farklı bir donanım sistemi kullanıyorsa? (PC, MAC,…)

… farklı bir işletim sistemi? (Windows, Unix,…)

… farklı bir veri gösterim biçimi? (ASCII, EBCDIC,…)

Çeşitlilik (networks, h/w, os, prog lan, impl)

Veya . . . ?

İşinizi ve bilgisayarlarınızı Güney'e taşımak isterseniz (hava şartlarından dolayı)?

Müşterileriniz Güney'e taşınırsa (daha muhtemel)?

Konum / Taşınma / Dağılım şeffaflığı

Ya . . . ?

İki müşteri aynı ürünü aynı anda sipariş verirse?

Concurrency (Aynı anda kullanım)

Veya . . . ?

Stok bilginizi tutan veri tabanı çökerse?

Sipariş esnasında müşterinizin bilgisayarı çökerse?

Fault tolerance (Hata payı)

Ya . . . ?

Birileri veri çalmak için sisteminize girmeye çalışırsa?

… bilgi çekerse?

… müşteriniz sipariş verir de, sonradan vermedim diyerek ürün teslimini reddederse?

Security (Güvenlik)

Veya . . . ?

Öyle başarılı olursunuz da, milyonlarca insan aynı anda online mağazanızı aynı anda ziyaret ederse?

Scalability (Ölçeklenebilirlik)

Sistem kurulurken …

Bütün yazılımı tek başınıza mı yazmak istersiniz (network, database,…)?

Güncellemeler, yeni teknoloji takibi?

Reuse (tekrar kullanılabilirlik) ve **Openness (açıklık)** (Standartlar)

Heterogeneity (Çeşitlilik)

Çeşitli bileşenler birbiriyle uyumlu şekilde çalışabilmeli

Distribution transparency (Dağınıklık şeffaflığı)

Dağınıklığın varlığı mümkün oldukça kullanıcıdan saklanmalı

Fault tolerance (Hata payı)

Bir bileşenin bozulması (kısmi bozukluk) tüm sistemin bozulmasına sebep olmamalı

Scalability (Ölçeklenebilirlik)

Sistem, artan kullanıcı sayısına rağmen verimli çalışmaya devam edebilmeli

Sisteme yeni kaynaklar eklenerek performans artışı sağlanabilmeli.

Dağıtık sistemler çeşitli bileşenler birbiriyle uyumlu şekilde çalışabilmeli

İşletim sistemleri

Donanım mimarileri

İletişim mimarileri

Programlama dilleri

Yazılım arayüzleri

Güvenlik ölçüleri

Bilgi gösterimleri

Şeffaf DS: Kullanıcılara, tek bir bilgisayar sistemi gibi görünür, bağımsız bilgisayarlar topluluğu olarak değil.

Şeffaflık çeşitleri

Erişim: Veri gösterimindeki farklar, ve kaynaklara nasıl erişildiği gizlenir. Yerel ve uzak kaynaklara erişim aynı işlemlerle sağlanır. Örn., Network File Systems (Ağ Dosya Sistemleri)

Konum: Bir kaynağın bulunduğu yer gizlenir. Kaynaklara, fiziksel konumları bilinmeden erişilir. Alan adının (domain name) makine adresinden ayrımı gibi.

Migration (Taşınma): Kaynağın yer değiştirme durumu gizlenir

Relocation (Yeniden konumlandırma): Kaynağın, kullanım esnasında, yer değiştirme durumu gizlenir

Migration / relocation şeffaflığı, bir sistemdeki kullanıcıların veya uygulamaların işlemlerini etkilemeden taşınabilmelerine olanak sağlar.

Çalışma esnasında (runtime) bir isim sunucusundan (name server) bir başkasına geçiş yapma

Bir vekil veya işlemin (agent/process) bir düğümden (node) diğerine taşınması gibi

Şeffaflık

Her zaman istenmez: Farklı kıtalarda bulunan kullanıcılar (context-aware), time-zone (zaman-dilimi), hız

Her zaman mümkün olmaz: Hataların gizlenmesi (bir bilgisayar yavaş mı, bozuk mu)

Yüksek şeffaflık ile performans arasındaki denge

Replication (Kopyalama): Bir kaynağın birden fazla yerde yedeğinin (kopyasının) tutulduğu gizlenir. Kopyalanmış kaynaklara, sadece bir kopya varmış gibi erişilir. Güvenilirlik ve performans kopyalarla artırılır, ama kullanıcıların veya uygulama geliştiricilerinin kopyalardan haberi olmaz.

Concurrency (Birlikte çalışma): Bir kaynağın birden fazla kullanıcı tarafından paylaşıldığı gizlenir. Bir işlem, diğer bazı işlemlerin de aynı kaynaklara erişmekte olduğunun farkında olmamalıdır

Failure (Bozulma): Bir kaynağın bozulma veya düzelme durumu gizlenir. Bozulmalar olsa da, görevler tamamlanabilir. Mesaj iletim tekrarı, bir ağ sunucusu düğümünün bozukluğu, web sitesini çökertmemeli.

Persistence (Süreklilik): Bir yazılım kaynağının bellekte veya disk üzerinde olma durumu gizlenir.

Performans: Yük değişimine göre performansı artırmak için sistemin tekrar yapılandırılması sağlanır. Örn., bileşenlerin dinamik olarak eklenip kaldırılması. Kullanıcı sayısı artınca, doğrusal yapılardan basamaklı (hierarchical) yapılara geçilmesi.

Scaling (Ölçekleme): Sistemin ve uygulamaların, sistem yapısında veya uygulama algoritmalarında değişikliğe gerek olmadan genişletilmesine olanak sağlar.

Ağ şeffaflığı: erişim + konum şeffaflıkları

Hata algılama

Checksums (sağlama), heartbeat (kalp atışı), …

Hata maskeleme

Bozuk mesajların tekrar gönderimi, fazlalık, …

Hataya müsamaha

Exception handling (istisna işleme), timeouts (zaman aşımları),…

Hatadan kurtulma

Rollback (geri sarma) mekanizmaları,…

Sistem Verimliliği ve Ölçeklenebilirlik

Sistem, küçük bir Intranet'ten Internet'e kadar uzanan birçok farklı ölçekte verimli olarak çalışabilmeli.

Kaynak ve kullanıcı sayısında belirgin bir artış olsa da etkili olarak çalışmaya devam edebilmeli.

Zamanla her şey çoğalır, kullanıcılar artar, bilgisayar sayısı büyür, veri miktarı fazlalaşır, ...

Sistemler nasıl ölçeklenebilir olur? Sadece donanım eklemekle mümkün değil. Bir makinenin 20 kullanıcıya hizmet vermesi, iki makinenin 40 kullanıcıya hizmet vereceği anlamına gelmez.

Resource (Kaynak):

Yalnız bir kullanıcı ⇒ kısıtlı verim

Çoklu istemci talepleri: birlikte erişim

Paylaşılan kaynaklara çoklu erişimin desteklenip yönetilmesi:

Birlikte çalışılan ortamdaki bir nesnenin güvende olması için

işlemleri, verileri tutarlı kalacak şekilde, senkronize olmalı (eşitlenmeli) (örn. Banka hesabı).

Bu, dağıtık olmayan sistemlerde semafor kullanımından daha zordur.

Availability (Ulaşılabilirlik)

Kaynak erişim yollarındaki çakışmalara karşı koruma.

Örn. Servis saldırılarının reddi

Reddedilememe

Bir bilginin gönderilme / alınma delili

Örn. Dijital imza

Encryption (Şifreleme)

Örn. Blowfish, RSA

Authentication (Doğrulama)

Örn. password (şifre), açık anahtarla yetkilendirme

Authorization (Yetkilendirme)

Örn. erişim kontrol listeleri

Dağıtık sistemler her yerde bulunur.

Internet, dünyanın her bir yanındaki kullanıcıların, her bir yandaki servislere erişimlerini sağlar.

Kaynak paylaşımı dağıtık sistem kurmaya teşvik eden etmenlerin başta gelenidir.

DS kurulumu birçok zorluğu beraberinde getirir:

Çeşitlilik, Açıklık, Güvenlik, Ölçeklenebilirlik, Hata denetimi, Birlikte çalışma, Şeffaflık.

Dağıtık sistemler küreselleşmeyi sağlar:

Topluluk (Sanal takımlar, kuruluşlar, sosyal ağlar)

Science (e-Science) (Bilim)

Business (e-Bussiness) (İş)

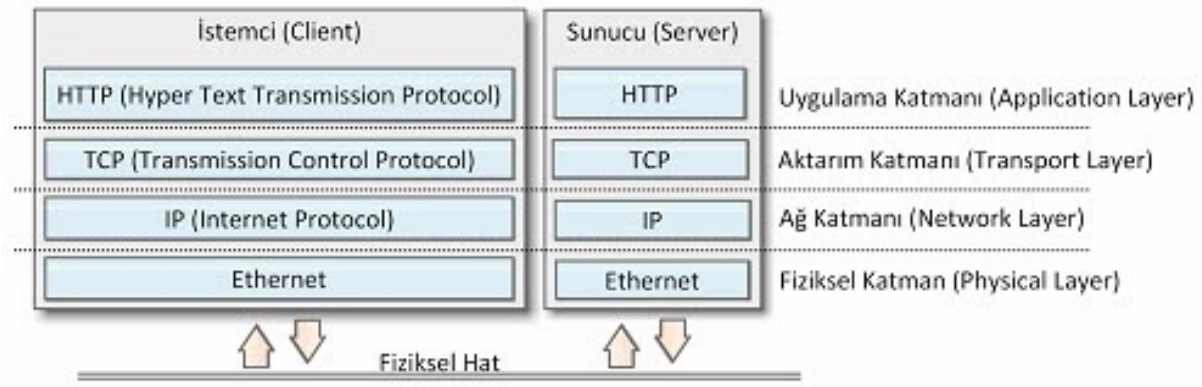
Uzak Sunucu Yöntemini Çağırmak İçin Gerekli Adımlar

Uzak programlama ile sunucu katmanındaki bileşene ait bir yöntemi veya kısaca uzak sunucu yöntemini çağırmak için aşağıdaki adımlar izlenir:

1. **Keşif** (discovery): Sunucunun ağdaki adresinin belirlenmesi işlemidir.
2. **Müzakere** (negotiation): İstemci ile sunucunun, bir sözleşme (conract) üzerinden, nasıl iletişim yapacağı konusunda anlaşmaya varmasıdır.
3. **Uzak işlevin çağrılması** (invocation): Sunucu işlevlerinin uzaktan çağrılıp, çalıştırılması işlemidir. Bu işlem **çağırma** (calling) olarak da adlandırılır.

Bu adımlar izlenirken karşılaşılan güçlükler aşağıda sıralanmıştır;

1. Sunucudaki nesnenin referansını elde etme,
2. Nesne yaratma (instantiation) ve ömür (lifetime) yönetimi,
3. Ağ üzerinde veriler bit bit veya bayt bayt aktarılır. Bu nedenle nesneye dayalı yöntemlerin bayt akışı (byte stream) kullanarak çağırılmasının yaratacağı problemler,
4. Çok kullanıcılı yazılımlarda güvenlik,
5. Güvenilmez ağ bağlantılarına (unreliable network connections) karşı önlemler,
6. Zaman uyumsuz (asynchronous) işlemler.



Yukarıda bahsedilen 3. madde üzerinde durmak gerekir. Sunucu bileşenleri nesne olarak tasarlandıklarından, nesneye dayalı yöntemleri (method) çağırmak için internet üzerinde bilgi iletişimi teknolojisi olarak bayt akışı (byte stream) kullanılır. Bunu bir web servisinden yararlanacak A ve B gibi iki uygulama ile açıklayalım;

1. A uygulamasının web servisinden yaptığı istekleri B uygulaması açığa çıkarabilir.
2. A uygulamasının isteklerinin biriktiği **protokol yığını** (protocol stack) parçalara bölünerek düşük düzey veri paketleri olarak, byte akışı üzerinden, servise gönderilir. Bu durumda B uygulaması sözkonusu paketleri tersine kullarak çağrılan işlevi açığa çıkarabilir.

Uzak programlama yapılırken bu problemleri bertaraf edecek önlemler alınmalıdır.

Günümüzde yazılım geliştirilen tüm platformlar TCP (Transmission Control Protocol) protokolünü desteklemektedir. Bu nedenle yazılımcılar genellikle bu protokolü kullanarak uzak programlama yaparlar.
Dağıtık programlamayı sağlayan bütün teknolojilerin temelinde Frank Buschmann tarafından 1996 yılında geliştirilen Simsar (broker pattern) adlı çözüm örüntüsü yatar. Bu nedenle uzak programlama yapacak programcıların bu örüntüyü mutlaka bilmesi gerekir.

MPI (Message Passing Interface) Mesaj Geçiş Arayüzü

Bir grup akademisyen ve çeşitli şirketlerden katılımcılar tarafından

Yaygın kullanım ve Taşınabilirlik amaçlanarak oluşturulan bir kütüphane standardıdır.

Sadece rutinleri belirler, implementasyonla alakası yoktur. Oldukça fazla bedava ve de özgür implementasyonu mevcut Sürüm 1'de sadece statik süreç oluşturma vardı

Sürüm 2'de dinamik olarak da kullanılabilmektedir.

MPI - Örnek

```
int main (int argc, char *argv[])
{
MPI_Init(&argc, &argv);
.
.
MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
if (myrank == 0)
        master();
else
        slave();
.
.
MPI_Finalize();
}
```

MPI KOMUTLARININ KISA TANIMLARI

MPI – Diğer Rutinler MPI_Send(buf, count, datatype, dest, tag, comm)

Bloke eden gönder rutini

MPI_Recv(buf, count, datatype, src, tag, comm, status)

Bloke eden alma rutini

MPI_Isend (...) / MPI_Irecv(...) - bloke etmeyen gönder/al

MPI_Wait() ve MPI_Test() ile denetlenebilirler MPI_Bcast() / MPI_Gather() / MPI_Scatter()

MPI_Reduce() - MPI_Gather gibi ama bir işlem yaparak alır MPI_Barrier() - Grup senkronizasyonu *

MPI_SendRecv() - İkili senkronizasyon

MPI – Örnek program

```
#include “mpi.h”
#include <stdio.h>
#include <math.h>
#define MAXSIZE 1000
void main(int argc, char **argv)
{
int myid, numprocs;
int data[MAXSIZE], i, x, low, high, myresult, result; char fn[255];
char *fp;
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
MPI_Comm_rank(MPI_COMM_WORLD,&myid);
if (myid == 0)
{ /* Open input file and initialize data */
        strcpy(fn,getenv(“HOME”));
        strcat(fn,”/MPI/rand_data.txt”);
        if ((fp = fopen(fn,”r”)) == NULL) {
            printf(“Can’t open the input file: %s\n\n”, fn);
exit(1); }
        for(i = 0; i < MAXSIZE; i++)
             fscanf(fp,”%d”, &data[i]);
   MPI_Bcast(data, MAXSIZE, MPI_INT, 0, MPI_COMM_WORLD);
   /* broadcast data */
   x = MAXSIZE/numprocs; /* Add my portion Of data */
   low = myid * x;
   high = low + x;
   myresult = 0;
   for(i = low; i < high; i++)
        myresult += data[i];
   printf(“I got %d from %d\n”, myresult, myid);
   /* Compute global sum */
   MPI_Reduce(&myresult, &result, 1, MPI_INT, MPI_SUM, 0,
```

```
            MPI_COMM_WORLD);
    if (myid == 0)
        printf("The sum is %d.\n", result);
    MPI_Finalize();
}
```

MPI – Derleme & Çalıştırma

A. ssh / rsh hazırlanır
B. Hesaplamada kullanılacak bilgisayarların listesi (hostfile) hazırlanır
C. mpicc ile programlar derlenir
D. mpirun ile çalıştırılır