Dağıtık Algoritmalar ve Dağıtık Programlar

- **Program**

  Bir program

  biçimsel tanımlanmış notasyon = programlama dilindeki bir algoritma,

  dil işlemsel(uygulanabilir) Semantik(=Gerçekleştirim) olarak düşünülebilir.

- **Dağıtık Program**

  Bir dağıtık program bir programlama dilinde tanımlanmış dağıtık bir algoritmadır.

  Problem: Dağıtık algoritmaların formülize edilebileceği neredeyse hiçbir programlama dili yoktur.

  Dağıtık algoritmalar iki bileşenlden oluşur:

  - Lokal süreçlerin tanımı

    genellikle işletim sisteminin iletişim operasyonlarını destekleyen programlama dilleriyle yapılır.

  - Topoloji tanımı

    genellikle otonom işlemciler üzerinde gerçekleştirilen süreçlerin yapılandırma/kurulumlarıyla yapılır.

  Her ikiside genellikle dökümanların homojen olmayan bir koleksiyonunda(program kodu, yapılandırma dosyaları,...) tanımlanmışlardır.

- **İkilem(Dilemma)**

  Dağıtık programlar olarak çalıştırılabilen dağıtık algoritmalar gerçekte uygulamaya özel detaylardır, anlaşılmaları zor ve gerçekleştirimleri pahallıdır.

  Gerçekleştirilemeyen sahte kod(Pseudocode) olarak ifade edilen dağıtık algoritmalar inandırıcı, ikna edici değillerdir. („sıkıcı“).

Dağıtık Algoritmadan Dağıtık Programa

- **Dağıtık Algoritma**
  Sahte Kod (Pseudocode) + Model (genellikle dolaylı)
- **Dağıtık program**
  Gerçekleştirim çeşitlilikleri:
  Programlama dilleri
  + desteklenen altyapı
  belirsiz, uygulama bağımlı Semantik
  ~> belirsiz Model

Dağıtık Programdan Dağıtık Uygulamaya

- **Dağıtık Program**

  iletişim kuran süreçler formunda bir çözüm

- **Dağıtık Uygulama**

  Uygulama odaklı probleme düzgün ayrılmış sistemler üzerine kurulmuş çözüm.

- **Dağıtık Uygulama ve Dağıtık Program**

  Şartlarla kesinlikle ayrılmış değillerdir ve genellikle eş anlamlı olarak kullanılabilirler
  - Dağıtık uygulamalar altyapı yerine uygulamaya dayalıdır.
  - Dağıtık uygulamalar aynı zamanda modüller/katmanlar olarak tanımlanabilen birçok dağıtık uygulamadan oluşabilirler.

Dağıtık Uygulamalar – Tarihçe

Neden Dağıtık

Uygulamanın ihtiyacına göre sistem dağıtılmıştır.

Dağıtıklıktan Kaynaklanan Problemler

- Eşleri bulma => Adlandırma Servisleri
- Sıralama(Marshalling): yapılandırılmış verileri dil ve platform sınırlarına göre taşıma(Büyüklük, Veri bit sıralaması, Karmaşık Verilerin Yapıları)

Çözümler 1

~ 1980 Dağıtıklık Donanım(Hardware) üzerinde gerçekleştirildi. (Ağ aygıtları için sürücüler)

~ 1990 Dağıtıklık bir işletim sistemi platformu üzerinde gerçekleştirildi.
(Ağ- / Soket-Programlama)
Windows - 1992'den itibaren TCP/IP desteği

~ 2000 Soket programlamada programlama dili desteği (çok ince bir katmanda)
Pionier Java

~ 1990 – 2000 Dağıtık İşletim Sistemleriyle ilgili yoğun araştırma
Ağ düğümleri üzerinde katman olarak işletim sistemi,
süreçlerin dağıtıklığı görünmez.

Başarısız oldu.

~ 1990 OSI : Oturum / Sunum Katmanı (Session-/Presentation-Layer)

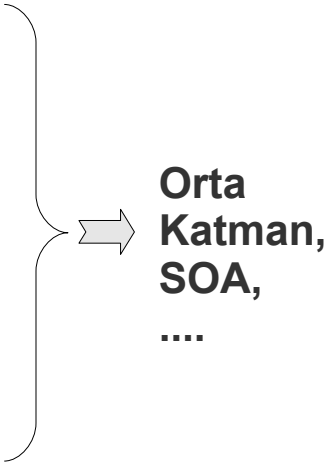
~ 2000 Orta Katman

~ 2005 SOA, Web … dağıtıklığın kutsal kasesi için arama devam etmektedir …

Dağıtık Uygulamalar – Tarihçe

90'ların Ortasındaki Dağıtım Platformları

- **Normal**: **Assembler'da sürücü yazımı yerine layer-4-API kullanılarak yazılmış C programları (Ağ Programlama / Soketler).**
- **Araştırma Konusu, tecrübeli geliştiriciler tarafından kullanım**
  - **Layer-4 tarafından soyutlama**
    örn. OSI: (*Presentation- / Session-Layer)*
  - **Özelleştirilmiş hizmetler tarafından desteklenme**
    örn. Adlandırma servisleri
  - **Standard Kod Blokları (üretilen / sağlanan)**
    örn. RPC
  - **Kütüphanelerde standart çözümlerin hazırlanması**
    örn. XDR
  - **Programlama dillerinde entegre edilmesi**



- **Karşı konulamaz artış: Web-Teknolojileri**
  Sorunları ve çözümleri basitleştirmek
  - **Sadece bir protokol: HTTP**
  - **(neredeyse) sadece bir serileştirme konsepti: Text, XML, JSON**

Orta Katman Üzerinde Dağıtık Uygulamalar

Orta Katman

Hedef ***Dağıtıklık uygulama geliştiricisi için büyük ölçüde şeffaf(görünmez) olmalıdır.***

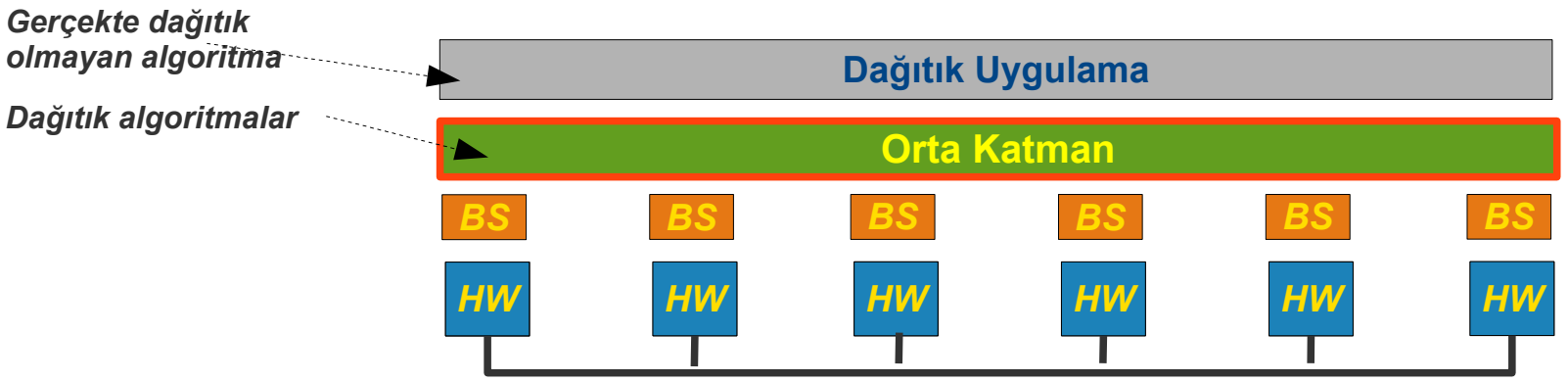
Bunun için yazılım soyutlamaları(Software Abstraction), örn.

- **Protokoller**
- **(De-) serileştirme yardımları**

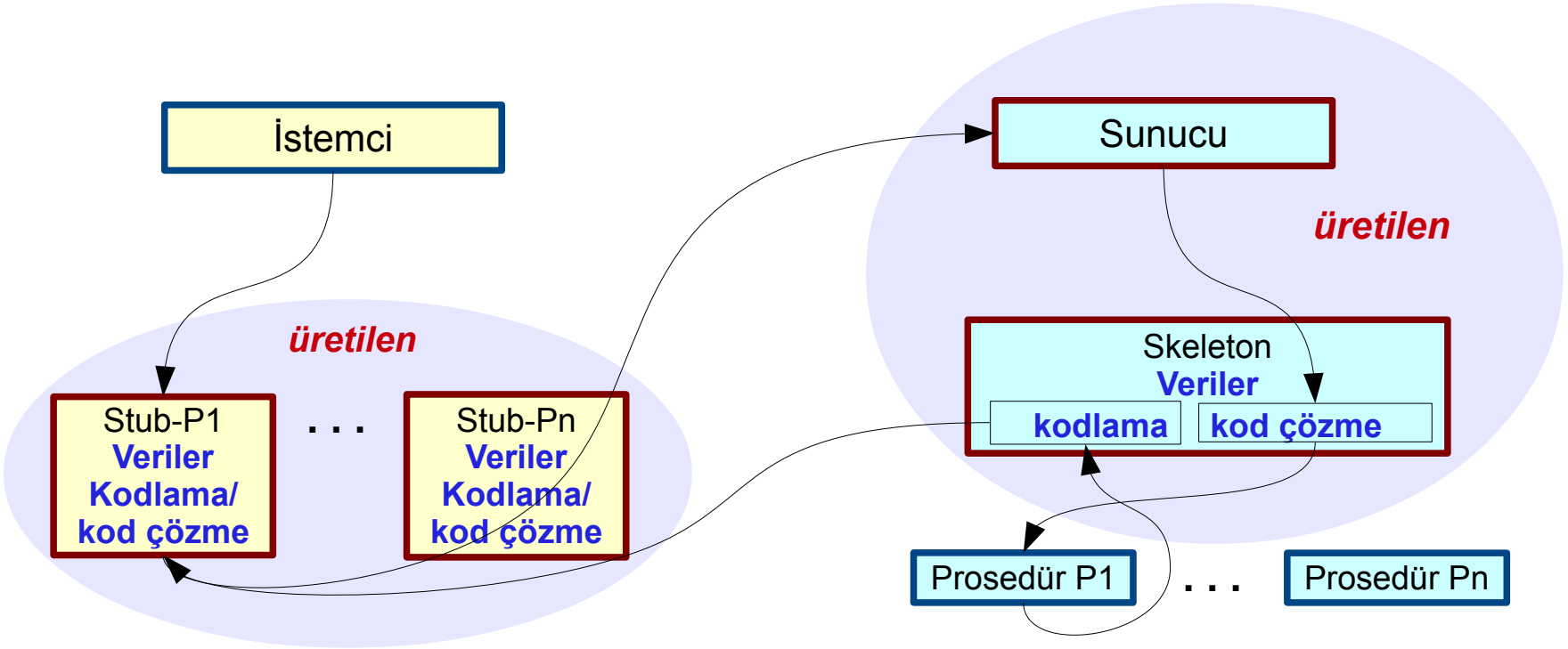
ve Araçlar ve Altyapı, örn.

- **Adlandırma hizmetleri (Eşlerin bulunması için)**
- **Broadcast**
- **Grup iletişimi**
- **…..**

ile sağlanmıştır



Orta Katman / Örnek RPC



Uygulama geliştiriciler oluşturulmuş kodlar ve kütüphane fonksiyonlarının yardımlarıyla dağıtık programlar ve prosedürler yazarlar.

Orta Katman / Örnek RMI

Dağıtık uygulama yazmak için savunulan strateji üç aşamalı yaklaşımdır.

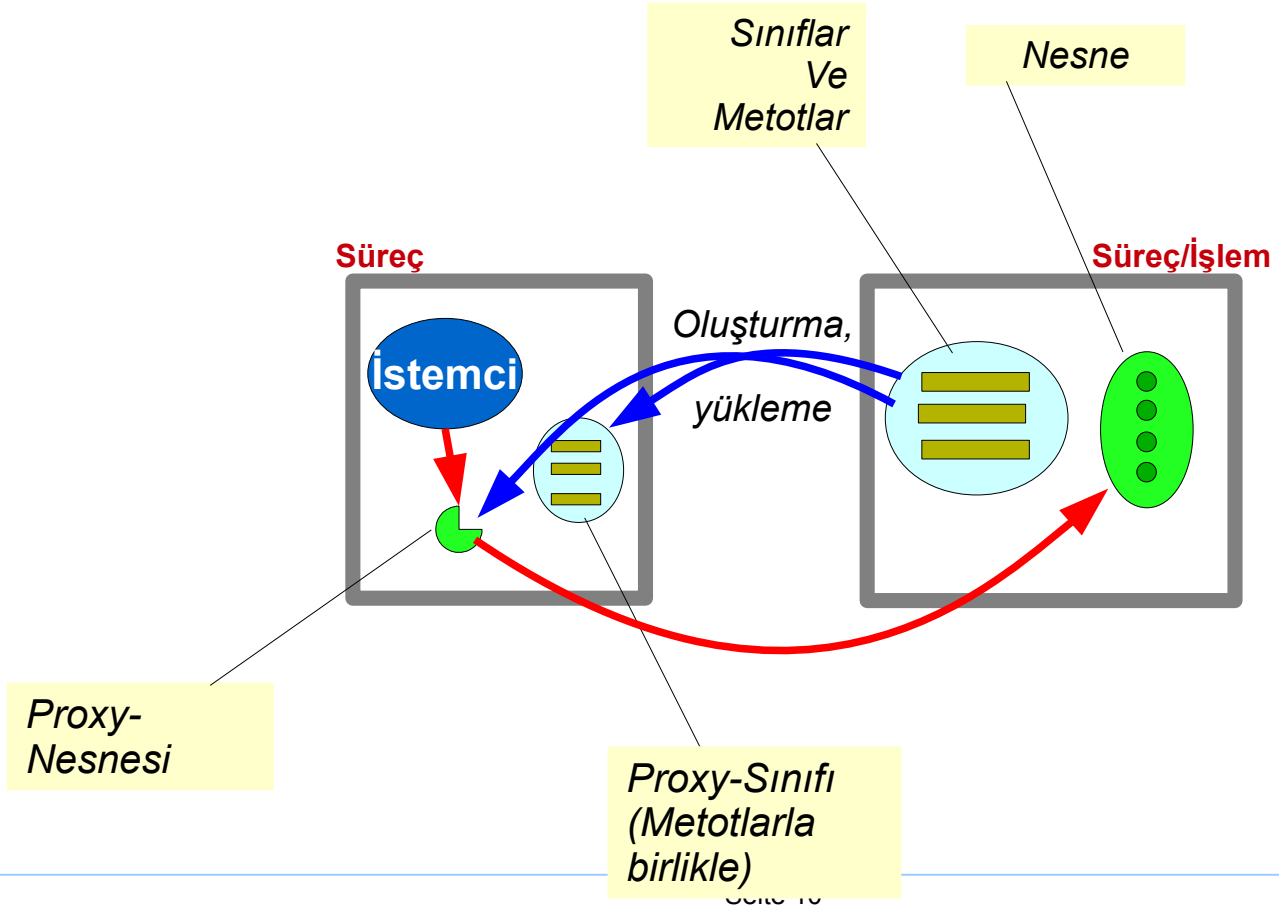
- ***İlk aşamada nesnelerin nerede oldukları ve iletişimlerinin nasıl sağlandığı konusunda endişe edilmez.***
- ***İkinci aşama nesne yerlerini ve iletişim metotlarını somutlaştırarak performasın ayarlanmasıdır.***
- ***Son aşama ise "gerçek mermi" ile test aşamasıdır(bölünmüş ağlar, çöken makineler,...)***

Waldo et al. "a note on distributed computing" (1994)

RMI RPC prensiplerinin geliştirilmiş bir halidir;
- **Dağıtık prosedürler yerine dağıtık nesneler kullanımı**
- **Bir programlama dili üzerine yoğun sınırlama**
- **bunlardan dolayı kolaylaştırma ve esnekleştirme.**

Orta Katman / RMI

- Serialization / deserialization
- oluşturulmuş(generated) sunucu işlemleri
- Gerekli sınıfların otomatik yüklenmesi (Nesneler herzaman sınıflara aittir)
- Proxy nesnelerinin otomatik oluşturumu ve yüklenmesi



Örnek Buffer : Lokal (dağıtık olmayan) Çözüm

```
public class Main {

static enum Token { PING, PONG };

static Buffer<Token> buffer = new Buffer<>();

public static void main(String[] args) {

  new Thread(new Runnable(){ // Producer
    public void run() {
      while (true) {
        try {
           buffer.put(Token.PING);
           Thread.sleep(1000);
           System.out.println(buffer.get());
        } catch (InterruptedException e) {}
      }
    }
  }).start();

  new Thread(new Runnable(){ // Consumer
    public void run() {
      while (true) {
        try {
           buffer.put(Token.PONG);
           Thread.sleep(1000);
           System.out.println(buffer.get());
        } catch (InterruptedException e) {}
      }
    }
  }).start();
}

}
```

```
public class Buffer<TokenType> {
  TokenType place;
  boolean empty = true;

  synchronized void put(TokenType t)
      throws InterruptedException {
    while ( ! empty ) wait();
    place = t;
    empty = false;
    notify();
  }

  synchronized TokenType get()
      throws InterruptedException {
    while ( empty ) wait();
    empty = true;
    notify();
    return place;
  }

}
```

Dağıtım: Producer / Consumer ve Buffer farklı süreçlerde.

Buffer : RMI ile gerçekleştirim (1)

```
import java.io.Serializable;
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface BufferI<TokenType extends Serializable> extends Remote {
      public void put(TokenType t) throws RemoteException, InterruptedException;
      public TokenType get() throws RemoteException, InterruptedException;
}
```

Servis katmanları
Bu servis istemcilerine nesneyi sunar..

```
import java.io.Serializable;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;

public class RMIBuffer<TokenType extends Serializable> implements BufferI<TokenType> {

  private TokenType place;
  private boolean empty = true;

  @Override
  public synchronized void put(TokenType t) throws RemoteException, InterruptedException {
      while (! empty) wait();
      place = t;
      empty = false;
      notify();
  }

  @Override
  public synchronized TokenType get() throws RemoteException, InterruptedException {
      while ( empty ) wait();
      empty = true;
      notify();
      return place;
  }
```

Service gerçekleştirimi

Puffer: RMI ile gerçekleştirim (2)

```
  public static void main(String[] args) throws RemoteException {

      RMIBuffer<Token> obj = new RMIBuffer<>();
      BufferI<Token> stub = (BufferI<Token>) UnicastRemoteObject.exportObject(obj, 0);

      Registry registry = LocateRegistry.createRegistry(Registry.REGISTRY_PORT);

      registry.rebind("RMIBuffer", stub);

      System.err.println("Server ready");
    }

}
```

Servis-Kayıt işlemi

Puffer : RMI ile gerçekleştirim (3)

Producer

```
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;

public class Pinger {

   public static void main(String[] args)
      throws MalformedURLException, RemoteException, NotBoundException, InterruptedException {

      BufferI<Token> buffer =
      (BufferI<Token>) Naming.lookup("rmi://127.0.0.1/RMIBuffer");

      while (true) {
            buffer.put(Token.Ping);
      }
  }
}
```

Consumer

```
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;

public class Consumer {

  public static void main(String[] args)
      throws MalformedURLException, RemoteException, NotBoundException, InterruptedException {

      BufferI<Token> buffer =
      (BufferI<Token>) Naming.lookup("rmi://127.0.0.1/RMIBuffer");

      while (true) {
            System.out.println(buffer.get());
      }
  }
}
```

Günümüzde Uygulamalar ve Orta Katman

– **„Dağıtıklığın Şeffaflığı“ hedefi ilişkilendirilir.**

Dağıtık programların kullanım durumu dağıtıklığın problemlerine dayanmaktadır.

Bu problemler soyutlaştırılamaz ve soyutlaştırılmamalı, bunun yerine – uygulama geliştiriciler tarafından – çözülmelidirler.

– **Bununla birlikte kendilerini dağıtık bilgisayarların sekiz yanılgısından(Peter Deutsch, James Gosling) uzak tutabilirler.**

1. ***The network is reliable.***
2. ***Latency is zero.***
3. ***Bandwidth is infinite.***
4. ***The network is secure.***
5. ***Topology doesn't change.***
6. ***There is one administrator.***
7. ***Transport cost is zero.***
8. ***The network is homogeneous.***

http://en.wikipedia.org/wiki/Fallacies_of_Distributed_Computing

– **RMI – örnek olarak – hiç bir hata ile karşılaşmadığı sürece, sistemde herşey yolunda gittiği sürece mükemmel.**

– **Infrastukturprobleme sind weitgehend gelöst, eine weitere absolute Transparenz wird nicht angestebt. ??**

Soru:
RMI asenkron modele ne ölçüde uymaktadır?

Web Teknolojileri ile Dağıtık Uygulamalar

- **Taşıma Protokolü olarak HTTP üzerinde indirgeme**
  Bütün iletişimler HTTP kullanır
- **Request-Response Uygulama Protokollerinde İndirgeme**
  Bütün uygulama (oturum) protokolleri request-response protokolleridir.
  Web-Soketleri: bu sınırlamanın gecikmiş aşımıdır
- **Serileştirme konseptlerinde indirgeme**
  XML / JSON / HTML