ESTUDAM Bilişim Dergisi
Cilt 4, Sayı 1 , 8-16, 2023

Journal of ESTUDAM Information
Volume 4, Issue 1, 8-16, 2023

DOI: 10.53608/estudambilisim.1225126

(Geliş Tarihi / Received Date: 27.12.2022, Kabul Tarihi/ Accepted Date: 05.01.2023)

(**Araştırma Makalesi**)

Karar Destek Sistemlerinde Çok Erkinli Mimari Kullanılması

Muammer AKÇAY[*1], Ömer SAVAŞ[2]

[1]Kütahya Dumlupınar Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Kütahya
ORCID No: https://orcid.org/0000-0003-0244-1275
[2]Kütahya Dumlupınar Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Kütahya
ORCID No: https://orcid.org/0000-0003-3662-2357

Anahtar Kelimeler:
Dağıtık sistemler,
karar destek sistemleri,
çok erkinli sistemler,
coğrafi Bilgi sistemi

Özet: Karar destek sistemleri özellikle büyük verilere sahip bilgi sistemlerinde tüm veriyi işleyerek kullanıcılarının strateji geliştirmesinden, yatırım planlamasına kadar çok geniş bir yelpazede fikir veren sistemlerdir. Karar destek sistemleri bunu yaparken farklı algoritmalar seçilebilir. En çok kullanılan algoritma Analitik Hiyerarşi Prosestir. Bu çalışma çok yüksek veri yüküne sahip özellikle coğrafi bilgi sistemlerinde hesaplama işlemlerini birden fazla bilgisayara dağıtarak kullanıcıya daha hızlı raporlar verebilmeyi amaçlamaktadır. İletişim ve veri işlemede "Çok erkinli (ajanlı) sistemler" dağıtık mimari ve metotları kullanılmıştır. Ajanlar veri yükünü paylaşarak kullanıcılara çok hızlı sonuçlar verebilmektedirler.

(**Research Article**)

Using of Multi Agent System in Decision Support System

Keywords:
Distributed systems,
decision support systems,
multi agent systems,
geographic information system

Abstract: Especially, decision support systems provide with large data to users by processing all the data in information systems from developing strategy, investment planning, and insight in a wide range. While doing this, different algorithms can be chosen. Most commonly used algorithm is Analytic Hierarchy Process. This study is aimed faster reports that calculation of very high load of data in geographic information systems by distributing the processes across multiple machines. "Multi agent systems" is used distributed architecture and methods in communications and data processing. Agents can provide very fast results to users by sharing the data load.

1. GİRİŞ (INTRODUCTION)

Gelişen internet teknolojisinin son kullanıcı için en önemli özelliği verinin daha erişilebilir olmasıdır. Günümüzde karmaşık verilere dahi erişim çok kolaydır. Burada veri yığınının büyüklüğü düşünülürse; verinin sunumu ve verinin karar vermede kullanılması bilginin verimli kullanılmasında çok önemlidir. Karar destek sistemlerinde en çok kullanılan algoritmalardan biri Analitik Hiyerarşi Prosestir [1].

Coğrafi Bilgi Sistemleri (CBS) bir konumsal veri sunma biçimidir [2] . CBS kullanarak konum bilgisinin yanında bununla ilişkili sözel veri sunumu da yapılabilir. Günümüzde araç takip sistemlerinden GPS cihazlarına; Karar destek sistemlerinden strateji planlamaya kadar birçok alanda şahıstan, şirketlere hatta devletlere kadar geniş kitleler tarafından CBS kullanılmaktadır. Bu nedenle CBS sistemi ile veri sunumunu daha verimli hale getirilmektedir.

Karar destek sistemleri de [3, 9-14] mevcut ve yoğun bilgiyi işleyerek kullanıcılarına strateji geliştirmeden, yatırım planlamasına kadar çok geniş bir yelpazede destek vermektedir. Veri ne kadar yoğun olursa sonuç o derece doğru olacaktır. Onun için bu çalışmada kaynak olarak çok yüksek veri yüküne sahip olabilen Coğrafi Bilgi Sistemleri kullanılacaktır.

Bir karar destek sistemi tek başına yorumlama yapamayacaktır. Burada insan faktörü çok önemli bir kriterdir. Sistem birbirlerine göre öncelikleri belirlenmiş verileri inceler ve bu verilere göre bir sonuca ulaşır.

*Sorumlu Yazar/Corresponding Author: makcay26@gmail.com

Öncelikler ne kadar doğru belirlenmiş ise sonuç o derece doğru olacaktır. Zamanla sonuçlar incelenerek edinilen tecrübeler ve öncelikler değiştirilebilir. Farklı girişlerin eklenmesi, daha sonra onların da diğer girişlere göre öncelik değerlendirmesi sonucunda sistemin verimliliğini arttıracaktır. Karar destek sistemlerinde özellikle çok yoğun verinin kullanılması bu iterasyonlar için daima zaman maliyeti yüksek işlemlerdir.

Bu çalışmada zaman maliyetini azaltmak (daha kısa sürede sonuç almak) için karar destek sistemindeki hesaplama yükü dağıtılacaktır. Daha çok kullanılan iş gücü sonuca daha hızlı erişim imkânı verecektir. Karar destek sisteminin sonuçta bir rapor üreteceğinden rapora mümkün olan en hızlı erişimin bir mecburiyet olduğu görülebilmektedir. Özetle, ihtiyaç olan veriye sahip bir coğrafi bilgi sisteminden faydalanarak karar destek sistemi ile bir sonuca ulaşılabilir. Bu çalışmada dağıtık mimari ile hesaplama işlemi kabiliyetini artırma planlanmaktadır. Bu çalışmanın amacı dağıtık bilgisayarlar arasındaki iletişim ve işlem gücünün arttırılmasıdır. Kullanılan dağıtık sistem çok erkinli (ajanlı) bir sistemdir. Birbirleri ile merkezi bir yönetimle iletişim halindeki bilgisayarlar birbirleriyle işlenmiş veriyi paylaşan erkinlerden (ajanlardan) oluşmaktadır. Bu sistemde oluşabilecek problemler çok ajanlı sistemlerdeki mimari ve metotlarla çözülebilmektedir.

Karar destek sistemleri problemlerinin çözümünde önerilen çözümlerden biri çok ajan (erkin) mimarisidir. Bu çalışmada çok erkin mimarisi geliştirilmiş ve örnek senaryolarda uygulanmıştır.

2. MATERYAL VE METOT (MATERIALS AND METHODS)

Öncelikle sistemin donanım tasarımını yapıldı. İlk olarak 3 adet bilgisayar ile donanım tasarımı gerçeklendi (Şekil 1). Bu bilgisayarlardan bir tanesi merkezi kontrol sunucusu, diğer iki bilgisayar buna bağlı ajan bilgisayar olarak hazırlandı. Ajan sayısı istenildiği kadar artırılabilmektedir. Merkezi sunucu tüm bilgisayarların haberleşmesini senkronize edecek ve ajanlardan gelen işlenmiş sonuçları inceleyerek nihai sonuca erişecek bilgisayardır. Diğer her ajan kendisine atanan işi belirli bir plan ve zaman aralığında tamamlayarak merkezi bilgisayara bilgi gönderecektir. Gerekirse diğer ajan bilgisayarlarda hesaplanmış olan bilgiler merkezi bilgisayardan talep edilebilecek ve işlemde kullanılacaktır.

Şekil 1. Sunucu ve erkinlerin (ajanların) bağlantı şekli.

Mesajlaşma için yeni bir sistem hazırlanmıştır. Bilgisayarların farklı platformlara sahip olabilecekleri düşünülürse ana bilgisayar üzerinde WSDL (Web Service Definition Language) [4] kullanan bir protokol tercih edilmelidir. Bunun için de SOAP (Simple Object Access Protocol) [5] kullanmak daha uygun olacaktır. Bu şekilde bir çözüm ile hem her işletim sisteminden erişim sağlanabilirken hem de gerekirse farklı platformlardan mesaj servisine bağlanabilecektir. Bu çalışma gerçeklenirken sunucu ve ajanlarda linux işletim sistemi üzerinde PHP dili ile uygulama geliştirilmiştir. SOAP sayesinde ileride gerekirse Mac işletim sistemine sahip bir bilgisayardan C++ ile geliştirilmiş bir uygulama mesajlaşma sistemine bağlanıp iş yapabilecektir.

Bu mesajlaşma senkronizasyonu işini yapacak bilgisayar için merkezi veri tabanı sunucusu uygun olacaktır. Mesaj servisini bu bilgisayar üzerinden yapacaktır. İşlenmiş verilere göre bir karar vermek çok iş yükü getirmeyeceğinden hem ajanların veri tabanından verileri okuması yavaşlamayacak hem de veriyi ve mesaj servisini tek kaynaktan alabileceklerdir.

İşletim sistemi olarak "ubuntu 13.04 server" ı uygun olacaktır. Hem yerleşik olarak diğer alternatiflere göre yeteri kadar hızlı, hem de GUI' sinin olmaması sisteme biraz daha hız katmaktadır. Ayrıca herhangi bir sorunda oldukça fazla kaynak bulunabilmektedir. Tüm bilgisayarlarda işletim sisteminden sonra aşağıdaki komutlar ile PHP geliştirme ortamı ve veri tabanına bağlanabilmek için eklenti kurulacaktır [6] .

```
- sudo apt-get install apache2
- sudo apt-get install php5
- sudo apt-get install libapache2-mod-
php5
- sudo apt-get install php5-pgsql
```

Komutlarıyla tüm bilgisayarlar hazır hale getirilmiştir. Mesajlaşma servisi ve istemciler yazılmalıdır. İstemci bilgisayarları IP adreslerine göre isimlendirilmektedir. Ana bilgisayar üzerinde çalışacak mesaj servisi Ek-1 de verilmiştir [7].

Bu servis PHP bir platform üzerinden SOAPSERVER fonksiyonu ile WSDL sözdizimini kullanarak servis vermektedir. Serviste `Oku`, `Yaz`, `Gönder`, `Al` isimlerinde dört adet fonksiyon bulunmaktadır. `Gönder` ve `Al` mesaj servisini kullanan bilgisayarların iletişim için kullanacağı fonksiyonlardır. `Oku` ve `Yaz` dışarıya kapalı fonksiyonlar ile gelen mesajları ilgili kullanıcılara gönderilene kadar hafızaya almaktadır.

`Oku` fonksiyonu tek parametre alan bir fonksiyon. Parametre olarak istenilen verinin ismini almaktadır. İlgili veriyi hafızadan okuyup geri döndürmektedir. Dışarıya kapalı olması sayesinde servisi kullanan bilgisayarların hafızaya direkt erişimi engellenmektedir.

`Yaz` fonksiyonu iki parametre alan yine dışarıya kapalı bir fonksiyondur. Birinci parametre daha sonra veriye

erişim için kullanılacak isimdir. İkinci parametre ise verinin kendisidir.

`Gönder` fonksiyonu servis edilen kullanıcıların birbirine mesaj göndermekte kullanacakları fonksiyondur. İki parametre almaktadır. Birinci parametre mesaj gönderilmek istenen bilgisayarın IP adresini göstermektedir. İkinci parametre gönderilmek istenen mesajdır.

Son olarak `Al` fonksiyonu da kullanıcıların kendi mesaj kutularını sorgulamakta kullanacakları servis fonksiyonudur. `Al` fonksiyonu parametre almamaktadır. Sistem mesajları talep eden kullanıcıyı tanıyıp bekleyen mesajlarını kendisine geri döndürmektedir. İstemci bilgisayardan ana bilgisayara erişim için aşağıda açıklaması yapılan fonksiyon kullanacaktır. Sınıfın kendisi Ek-2 verilmiştir [8].

Bu sınıf `Flood`, `Al` ve `Gönder` isminde 3 adet fonksiyon içermektedir. `Flood` korumalı, `Al` ve `Gönder` serbest fonksiyonlardır. `Flood` fonksiyonu mesaj sunucusunu mesaja boğmamak için eklenmiş bir korumadır. Her kullanıcının en fazla 2 saniyede bir mesaj göndermesine izin verir. Bu şekilde hem servis yavaşlatılmamış olur hem de spam saldırılarına karşı kesin olmasa da engelleyici bir tedbir alınmış olur.

`Al` fonksiyonu serbest bir fonksiyondur. Nesne yaratıldıktan sonra kullanılabilir. Parametre almaz. Çağırıldığında mesaj servisine erişir. O bilgisayar için gönderilmiş mesajları alarak parse eder ve dizi olarak geri döndürür. Gönder fonksiyonu iki parametre alan bir fonksiyondur. Mesaj gönderilecek bilgisayarın IP adresi ve mesajdır. Servise bağlanır. Sisteme ilgili mesajı iletir. Geri değer döndürmez.

Bilgisayarlar arası iletişim bu şekilde sağlanmıştır. İlk olarak örnek bir sorun seçilsin. Sınırları, karayolları ve nüfusu bilinen belirli bir alandaki yerleşim yerleri için mevcut durumu inceleyecek ve yol ihtiyacı en yüksek olan yerleşim yerinin belirlenmesi istenmektedir. Bu çok önemli bir stratejik plandır.

1. erkin (ajan) önce TÜİK (Türkiye İstatistik Kurumu)'ten alınan nüfus verisini işleyip yerleşim yerlerinin erkek, kadın ve çocuk nüfusu olarak belirlemektedir. Taşra yerleşim yerlerinde yol ihtiyacı için önce erkek, ardından da çocuk nüfusu önemlidir. Buna göre nüfusları ağırlıklandırıp (katsayılandırıp) toplayarak ana bilgisayara sonucu bildirmektedir. Burada algoritma geliştirmek yine kullanıcının tercihidir. Ancak veri çekmek ve göndermek ile ilgili 1. ajan için örnek kod aşağıdadır.

```
//Veritabanı bağlantısı
if(! $db = @pg_connect("host=".$mesaj->kontrol_IP." dbname=kubis port=5432 user=postgres password=********"))
{
      echo 'Veri Tabanına Erişilemedi!';
      exit(0);
}

//Nufus verisini getir
$sql = 'select erkek_nufus, kadin_nufus, cocuk_nufus, ilce_id from g_koy_yerlesim y JOIN g_ilce_sinir i ON y.ilce_id = i."MI_PRINX"';
$sorgu = pg_query($db, $sql);

$nufus = 0;

//Tüm kayıtları gezerek yerleşim yerleri için ağırlıklı nüfusu hesapla
while($kayit = pg_fetch_array($sorgu))
      $nufus += $kayit['erkek_nufus'] * 0.35 + $kayit['kadin_nufus'] * 0.25 + $kayit['cocuk_nufus'] * 0.4;

//Sonuçları ana makineye bildir
$mesaj->gonder($mesaj->kontrol_IP, 'nufus*****'.$i.'|'.$nufus);
$mesaj->yaz('i', $i);

//Sayfayı monitör eden varsa bilgi yazdır
echo 'Veri İşlendi ve Gönderildi! (İlçe No: '.$i.')';

//Veritabanı bağlantısını kapat
if (pg_close($db) != 1) echo "<br>Hata: Bağlantı Sonlandırılamadı!";

/* End of file index.php */
/* Location: ./index.php *
```

Bu program parçası sırası ile merkezi veri tabanına bağlanıp ilgili veriyi almaktadır. Tüm kayıtları **bir**er **birer gezerek** ağırlıklı **nüfus** bilgisini hesaplamaktadır. Son olarak da işlenmiş sonuç verisini ana bilgisayara mesaj servisini kullanarak bildirmektedir. Sayfayı izleyen kullanıcılar için bilgi yazdırılmaktadır. Son olarak da veri tabanı bağlantısı kapatılmaktadır.

Benzer olarak ajan 2 ise yerleşim yerlerinin kullandığı yol uzunluklarını asfalt, stablize ve özniteliklerine göre katsayılandırarak hesaplayarak -örneğin asfalt yol uzunluğunu 3 **ile** stablize yol uzunluğunu 2 ile çarparak- ana bilgisayara göndermektedir.

```
//Veritabanı bağlantısı
if(! $db = @pg_connect("host=".$mesaj->kontrol_IP." dbname=kubis port=5432 user=postgres password=********"))
{
      echo 'Veri Tabanına Erişilemedi!';
      exit(0);
}
```

```
//Yerleşim yerleri ile coğrafi eşleştirmeleri yaparak yolları getir
$sql = 'select m.islem_id, m.yol_id, a.ilce_id, ST_Length(m."SP_GEOMETRY") as uzunluk from g_koy_yolu_mevcut m JOIN g_koy_yolu_ag a ON m.yol_id = a."MI_PRINX" where a.ilce_id = ' . $i;
$sorgu = pg_query($db, $sql);

//Her kayıt için ağırlıklı uzunlukları hesapla
while($kayit = pg_fetch_array($sorgu))
{
      if(!isset($dizi['islem'.$kayit['islem_id']]))
$dizi['islem'.$kayit['islem_id']]     = 0.0;
      $dizi['islem'.$kayit['islem_id']] += floatval($kayit['uzunluk']);
}

$veri = '';


//Hesaplanmış verileri okunabilir formata getir
foreach($dizi as $k => $v) $veri .= $k .     '-'     .str_replace(',',     '', number_format($v, 2)).'*';

//Ana makineye sonuçları bildir
$mesaj->gonder($mesaj->kontrol_IP, 'yol*******'.$i.'|'.$veri);
$mesaj->yaz('i', $i);

//Monitör eden var ise bilgi göster
echo 'Veri İşlendi ve Gönderildi! (İlçe No: '.$i.')';

//Veri tabanı bağlantısını kapat
if (pg_close($db) != 1) echo "<br>Hata: Bağlantı Sonlandırılamadı!";

/* End of file index.php */
/* Location: ./index.php */
```

İkinci ajan veri tabanına bağlanıp veriyi almaktadır. Tüm veri tabanı kayıtları teker teker işlenerek ağırlıklı uzunluklar hesaplanmaktadır. Daha sonra bu veriler anlaşılabilir bir bilgiye dönüştürülmektedir. Son olarak ana bilgisayara sonuçlar gönderilmektedir. Web sayfasını takip eden kullanıcılar için bilgi verilip veri tabanı bağlantısı kapatılmaktadır.

Ana bilgisayar diğer bilgisayarlardan veri bekleme esnasında yerleşim yerlerinin alanlarını hesaplamaktadır. Verilerin tamamı oluşturulduktan sonra alan, yol uzunluğu ve nüfus kriterlerinin birbirlerine göre öncelikleri belirlenmiştir. Bu belirlemelere göre Analitik Hiyerarşi Proses algoritması uygulayarak bir sonuca ulaşılmaktadır.

3. BULGULAR (RESULTS)

Birinci ajanın hesapladığı sonuçlar aşağıdaki gibidir. Tüm sonuçlar çok uzun olduğu için ilk birkaç sonuç örnek olarak verilmiştir. Sütun isimleri yerlesim_yeri_id ve ağırlıklı nüfus bilgisidir. Sonuçlar ,Tablo 1-5 verilmiştir.

Tablo 1. Ağırlıklı Sonuç Değerleri.
Array

Yeleşim yeri ID	Ağırlıklı Nüfus (float)
3	3439.5
4	522.05
12	16161
1	3736.25
6	9170
9	685.95
7	2575.05
10	12850.15
8	6895.45

2. ajanın hesapladığı sonuçlar aşağıdaki gibidir. Tüm sonuçlar çok uzun olduğu için ilk birkaç sonuç örnek olarak paylaşılmıştır. Sadece Dizinin ilk boyutu `yerlesim_yeri_id` ikinci boyut ise `tür_id` (asfalt, stabilize gibi). `Değerler` ise uzunluk olarak verilmiştir.

Tablo 2. Yerleşim yerlerinin yol uzunlukları.
3=> Array

Yeleşim yeri ID	Ağırlıklı Nüfus (float)
3	212174.49
4	25900.74

Tablo 3. Yerleşim yerlerinin yol uzunlukları.
5 => Array

Yeleşim yeri ID	Ağırlıklı Nüfus (float)
8	851112.73
1	15847.6
2	23442.14
11	11600.66
4	2096.35

Tablo 4. Yerleşim yerlerinin yol uzunlukları.
10=> Array

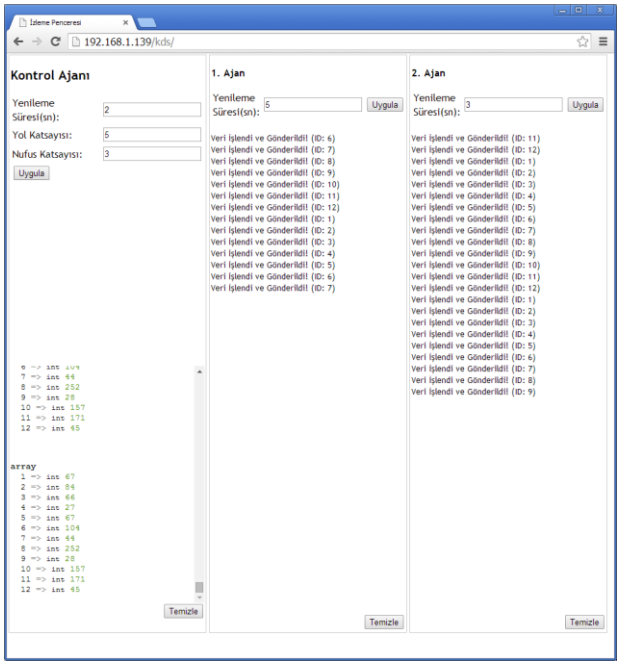
Yeleşim yeri ID	Ağırlıklı Nüfus (float)
10	551655.67
11	1071.53
9	5068.67

Ana bilgisayar üzerinde toplanan bu değerler ağırlıklarına göre hesaplanmış ve normalize edildikten sonra aşağıdaki gibi bir sonuca ulaşılmıştır.

Tablo 5. Önceliklerine göre hesaplanmış yardım alma değerleri. Array

Yeleşim yeri ID	Ağırlıklı Nüfus (float)
1	67
2	84
3	66
4	27
5	67
6	104
7	44
8	251
9	28
10	157
11	171
12	45

Tüm ajanların ve kontrol ajanını monitör eden bir test sayfası hazırlanarak aynı anda tüm bilgisayarlar izlenebilmektedir. Mesajlaşma servisinin verdiği esneklikle bu her platformdan yapılabilmektedir. HTML, javascript ile geliştirilmiş benzer bir uygulamanın ekran çıktısı aşağıdadır (Şekil 2).



Şekil 2. Bir izleme uygulaması ekran görüntüsü

4. TARTIŞMA VE SONUÇ (DISCUSSION AND CONCLUSION)

Sonuç olarak ajanlar veri yükünü paylaşarak kullanıcılara çok hızlı sonuçlar verebilmektedirler. Normal şartlarda çok yoğun çalışmayan sunucular böyle bir iş için kullanılarak kullanım verimi arttırılmaktadır.

Buna ilaveten platform bağımsız mesajlaşma servisi sayesinde istenirse mobil cihazlara ve hemen hemen her işletim sahibine sahip cihaz bir ajan gibi kullanılıp hesaplattırma yaptırılabilir. Hatta bu cihazların donanım özellikleri sisteme bir sensör kaynak olarak eklenebilir. Bu da bilgi sistemini, statik, kapalı sistem bir uyguluma olmaktan çıkararak dinamik hem iş yükünü paylaşabilen hemde farklı kaynaklardan farklı veriler alabilen bir sistem haline getirmektedir.

Etik Hususlar (Ethical Considerations)

Etik kurallara uyum (Compliance with ethical guidelines)

Yazarlar etik görev ve sorumluluklara riayet edildiğini beyan ederler.

Finansman (Funding)

Yazarlar kamu, ticari veya kar amacı gütmeyen sektörlerdeki fon kuruluşlarından özel bir hibe alınmadığını beyan ederler.

Çıkar çatışması (Conflict of interest)

Yazarlar herhangi bir kişi veya kurumla çıkar çatışmasının bulunmadığını beyan ederler.

Teşekkür (Acknowledgment)

Projenin hazırlanması esnasında bizlere sabır gösteren değerli eşlerimiz ve çocuklarımıza teşekkür ederiz.

KAYNAKÇA (REFERENCES)

[1] Ishizaka, A. and Labib, A., 1999. Analytic Hierarchy Process and Expert Choice, Benefits and Limitations, Portsmouth Business School, University of Portsmouth..

[2] Coğrafi bilgi sistem, http://tr.wikipedia.org/wiki/Co%C4%9Frafi_bilgi_sistemi, (Erişim Tarihi: 30.10.2021).

[3] Arslan V, Yılmaz G., 2010. Karar Destek Sistemlerinin Uygulanması İçin Uygun Bir Model Geliştirilmesi, Havacılık Ve Uzay Bilimleri Dergisi, Cilt 4 Sayı 4 (75-82).

[4] WSDL, http://www.w3.org/TR/wsdl, (Erişim Tarihi: 30.10.2021).

[5] SOAP, http://tr.wikipedia.org/wiki/SOAP, (Erişim Tarihi: 30.10.2021).

[6] Ubuntu, http://www.howtogeek.com/howto/ubuntu/installing-php5-and-apache-on-ubuntu/, (Erişim Tarihi: 30.10.2021).

[7] SOAPSERVER, http://www.php.net/manual/en/class.soapserver.php, (Erişim Tarihi: 30.10.2021).

[8] SOAPCLIENT, http://tr2.php.net/manual/en/class.soapclient.php, (Erişim Tarihi: 30.10.2021).

[9] Dong, C.S.J., Srinivasan, A., 2013. Agent-enabled Service Oriented Decision Support Systems, Decision Support Systems, 55, 364-373.

[10] Ozceylan, E., Erbas, M., Tolon, M., Kabak, M., Durgut, T., 2016. Evaluation of Fright Villages: A

GIS-based Multi-criteria Decision Analysis, Computers in Industry, 76,. 38-52.

[11] Saarloos, D. J. M., Arentze, T. A., Borgers, A. W. J., Timmermans, H. J. P. 2008. A Multi-agent Paradigm as Structruing Principle for Planning Support Systems, Computers, Environment and Urban Systems, 32, 29-40.İnternet: Millî Eğitim Bakanlığı Hayat Boyu Öğrenme Kurumları Yönetmeliği, http://hbogm.meb.gov.tr/meb_iys_dosyalar/2018_04/11093946_MEB_HBO_KURUM LARI_YYNETMELYYY.pdf, 11 Nisan 2018.

[12] Laandgren, P., Srivastava, V. Leonard, N. E., 2021. Distributed Cooperative Decision Making in Multi-agent Multi-armed Bandits Automatica, Vol 125, https://doi.org/10.1016/j.automatica.2020.109445

[13] Radulescu, R., Mannion, P., Roijer, D. M., Nowe, A. (2020). Multi-objective Multi-agent Decision Making: a Utility-based Analysis and Survey. Autonomous Agents and Multi-Agent Systems (2020) 34:10 https://doi.org/10.1007/s10458-019-09433-x

[14] Bulling, N. (2014). A Survey of Multi-Agent Decision Making. Ku¤nstl Intell (2014) 28:147–158. DOI 10.1007/s13218-014-0314-3

EKLER

Ek-1

```
class Mesaj
{
 /**
 * Oku
 * İstenilen isimdeki veriyi oku
 *
 * @access private
 * @param string İstenilen verinin adı
 * @return string Okunan değer
 */
 private function oku($ad)
 {
      return apc_fetch($ad);
 }

//---------------------------------------------------------------------
 /**
 * Yaz
 * Verilen isimdeki veriyi yaz
 *
 * @access private
 * @param string Verinin adı
 * @param string Verinin değeri
 */
 private function yaz($ad, $veri)
 {
 try
 {
   if(apc_fetch($ad)) apc_delete($ad);
   apc_store($ad, $veri);
 }
 catch (Exception $ex)
 {
      return $ex;
 }
 return TRUE;
 }

//---------------------------------------------------------------------
/**
 * Gönder
 * Belirtilen bir kullanıcıya mesaj gönder
 *
 * @access public
 * @param string Erişim Kodu
 * @param string Mesaj Gönderilecek IP
 * @param string Mesaj
 * @return string Erişim Kodu
 */
 public function gonder($kime, $mesaj)
 {
 self::yaz('mesaj-'.$kime,                                    self::oku('mesaj-
'.$kime).$_SERVER['REMOTE_ADDR'].'::'.$mesaj.'<>');
 }

//---------------------------------------------------------------------
/**
 * Al
 * Belirtilen bir kullanıcıya ait mesajları al
 *
```

```
 * @access public
 * @param string Erişim Kodu
 * @return string Mesajlar
 */
 public function al()
 {
 $mesaj = self::oku('mesaj-'.$_SERVER['REMOTE_ADDR']);
 self::yaz('mesaj-'.$_SERVER['REMOTE_ADDR'], '');
 return $mesaj;
 }

//--------------------------------------------------------------------
}

$sunucu=new                                                    SoapServer(null,
array('uri'=>'http://192.168.1.136/kds/mesaj.php'));

$sunucu->setClass('Mesaj');
$sunucu->handle();

/* End of file mesaj.php */
/* Location: ./kds/mesaj.php */
```

Ek-2

```
class Mesaj
{
 private $istemci = NULL;
 public $kontrol_IP = '192.168.1.139';
 private $uri = 'http://192.168.1.139/';
 private $location;

 public function __construct()
 {
 $this->location = 'http://'.$this->kontrol_IP . '/kds/mesaj.php';
 $this->istemci = new SoapClient(null, array
 (
 'uri' => $this->uri,
 'location' => $this->location
 ));
 }

 /**
 * Flood Koruması
 * Mesaj sunucusunu mesaja boğmamak için flood kontrolü
 *
 * @access private
 */
 private function flood()
 {
 if (self::oku('sec') > time() - 2)
 {
 usleep(100);
 self::flood();
 }

 self::yaz('sec', time());
 }

//--------------------------------------------------------------------
 /**
 * Al
 * Mesaj Al
```

```
 *
 * @access public
 * @return string Gelen Mesajlar
 */
 public function al($str = FALSE)
 {
 self::flood();
 $string = $this->istemci->al();
 if($str) return $string;
 $i = 0;
 foreach(explode('<>', $string) as $satir)
 {
 $d = explode('::', $satir);
 if(count($d) == 1) continue;
 $dizi[$i]['IP'] = $d[0];
 $dizi[$i]['mesaj'] = $d[1];
 $i++;
 }
 return @$dizi;
 }

//--------------------------------------------------------------------
/**
 * Gönder
 * Mesaj Gönder
 *
 * @access public
 * @param string Mesaj Gönderilecek IP
 * @param string Mesaj
 */
 public function gonder($IP, $mesaj)
 {
 self::flood();
 $this->istemci->gonder($IP, $mesaj);
 }

//--------------------------------------------------------------------
}
 
$mesaj = new Mesaj();
$veri = $mesaj->al();
```