Dağıtılmış Bileşik Nesne Tabanlı Ortam Üzerinde Lojistik Uygulaması

Logistics Application on Distributed Composite Object Based Environment

Jamshid Nakhchivanski
Bilgisayar Mühendisliği Bölümü
Elektrik-Elektronik Fakültesi
İstanbul Teknik Üniversitesi
nakhchivanski@itu.edu.tr

Güray Yılmaz
Bilgisayar Mühendisliği Bölümü
Hava Harp Okulu, Istanbul
g.yilmaz@hho.edu.tr

Nadia Erdoğan
Bilgisayar Mühendisliği Bölümü
Elektrik-Elektronik Fakültesi
İstanbul Teknik Üniversitesi
nerdogan@itu.edu.tr

Özet

Günümüzde dinamik veri alış-verişi sağlayan paylaşımlı sistemlere ihtiyaç artmaktadır. Veri paylaşım noktalarının çokluğu, bu noktalar arasında iletilen veri yoğunluğu nedeniyle hızlı çalışan, gerektiğinde veri gönderimine ve güncellenmesine ihtiyaç duyan paylaşımlı sistemlerin geliştirilmesine ilgiyi arttırmıştır.
Bildiride dinamik ve paylaşımlı sistemlere örnek olarak güncel bir lojistik uygulaması incelenecektir. Uygulamanın çalışma ortamı olarak Dağıtılmış Bileşik Nesne modeli [1][2] kullanılacaktır. Bu model, düğümler arasında arakatman vazifesini üstlenerek dağıtılmış uygulama için gerekli olan haberleşme, verilerin kopyalanması, parçalanma, tutarlılık, uygulamaların dinamik olarak yüklenmesi v.b. gibi temel mekanizmaları sağlar.

Abstract

Demand for shared systems providing dynamic data exchange is increasing today. Multiplicity of data sharing nodes, intensity of transmitting data between these nodes is increased interest to develop that kind of shared systems which running fast, requiring data sending and upgrade when needed. In this declaration there will be studied as an example: an actual application of logistics in a dynamic and shared systems. Distributed Composite Object model [1][2] will be used as an environment of the application. This model undertake middleware function among the nodes and provides the basic mechanism as communication, data duplication, fragmentation, consistency, dynamic load of applications, etc. which are necessary for distributed applications.

1. Giriş

Son yıllarda uygulamaların bilgi işlem alanı bireysel ortamlardan ağ ve internet ortamına taşındığı için geniş alana yayılmış bilgiyi kontrol eden uygulamaların tasarımı konusu ortaya çıkmıştır. Burada ayrıca, yalnız uygulama tasarımı değil, hem de bu uygulamaların çalıştığı dağıtılmış ortamların geliştirilmesinin de önemi artmıştır. Dağıtılmış sistemler bilgisayar ağları üzerinden sistem kaynaklarının ve bilginin paylaşımını sağlarlar. Dağıtım ile ilgili tüm ayrıntıların kullanıcılara saydam olarak gerçekleniyor olması bu sistemleri çekici yapan önemli bir tasarım konusudur.
Geniş alan uygulamaları kullanıcılarına bilgi alış-verişi ve paylaşımı için bazı kolaylıklar sağlarlar. Bu bağlamda, dağıtılmış sistemler üzerine günümüze değin bir çok proje geliştirilmiştir. Bu modeller genelde *uzaktan yordam çağırma (Remote Procedure Call-RPC), noktadan-noktaya mesaj iletimi (point-to-point message passing), dağıtılmış paylaşılan bellek (distributed shared memory)* kullanımı gibi düşük-düzeyli haberleşme ilkellerini kullanarak gerçeklenmişlerdir[3][4]. Ancak, bu nispeten düşük düzeyli haberleşme modelleri uygulamada bazı problemlerin de kaynağı olurlar. Öncelikle, bu modeller *kopyalama (replication)* ve *verinin göç ettirilmesi (data migration)* gibi, tüm dağıtılmış sistemler için geçerli olan haberleşme problemlerine çözüm sağlayamamaktadırlar. Bu noktada, geniş alan dağıtılmış sistemleri için bilgi alış-verişi ve paylaşımı konusunda karşılaşılan haberleşme problemlerini çözümleyecek bir modele ihtiyaç olduğu görülmektedir. Bizim burada inceleyeceğimiz ve daha sonra üzerinde lojistik uygulama tasarlayacağımız Dağıtılmış Bileşik Nesne modeli, basit haberleşme ilkellerine oranla daha soyut düzeyde ve aynı zamanda internet üzerinde ortaklaşa iş yürütmeyi destekleyen, farklı yapılardaki bir çok geniş alan uygulamasının geliştirilebilmesine olanak sağlayan genel ve basit bir gerçekleme ortamı sunmaktadır.
Bu model ileride anlatacağımız dağıtılmış bileşik nesne yapısını desteklemektedir. Dağıtılmış sistemler için nesne tabanlı çözümlerin daha elverişli olduğu görülmüştür[5]. Öncelikle, nesne kavramı işlevselliği gerçeklemeden ayıran doğal bir yapı sunmaktadır. Bu ayırım gereklidir. Çünkü, geniş alan uygulamaları bileşenlerin farklı özelliklerdeki alt sistemler üzerine dağıtılmalarını gerektirebilir. Bu durum, aynı işlevselliğe sahip, farklı gerçeklemelerin bulunmasını zorunlu kılar.

Diğer sebep ise, nesnelerin verilere yalnızca önceden tanımlanmış olan işlemler üzerinden erişilmesine izin veren doğal yapısıdır[6].
Tasarlayacağımız lojistik uygulama dağıtılmış ortamda dinamik yönetilen paylaşılmış sistemlere iyi bir örnektir. Burada farklı düğümlerdeki nesneler (taşıtlar, malzemeler) ve onların özelliklerini DBN modelinin bileşik nesne yapısına uygunlaştırarak bu modelin Java ortamında tasarlanmış Dağıtılmış Bileşik Nesne Tabanlı Ortam'ı (DBNTO) üzerinde gerçekleyeceğiz.

2. Dağıtılmış Bileşik Nesne (DBN) modeli

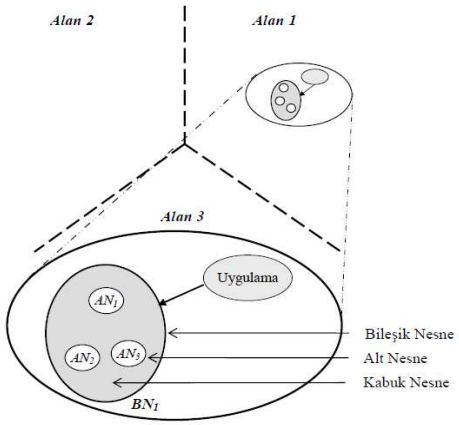
İşbirliğiyle yürütülen dağıtılmış uygulamalar genel olarak iri ya da orta taneli nesneler üzerinde çalışmalarına rağmen, bu tip uygulamaların kullanıcıları genelde nesnelerin daha küçük bir parçasıyla ilgilenmektedirler. Bundan dolayı, eğer büyük bir nesne daha küçük alt nesnelere parçalanabilirse ve istekçi alana nesnenin yalnızca kullanılacak olan ilgili parçası gönderilirse, erişilecek ve farklı alanlar arasında aktarılacak olan verinin miktarı azalacağından, çalışan uygulamanın başarımı da yükselecektir. Kopyalamaya dayalı yeni bir model olan DBN modeli bu noktadan yola çıkarak geliştirilmiştir. Bu modelinin ardında yatan temel fikir, çok sayıdaki alt nesnenin bileşiminden meydana gelen bir bileşik nesnenin farklı alanlar üzerinde fiziksel olarak dağıtılmasıdır.

2.1. DBN modelinin haberleşme şekli ve nesne yapısı

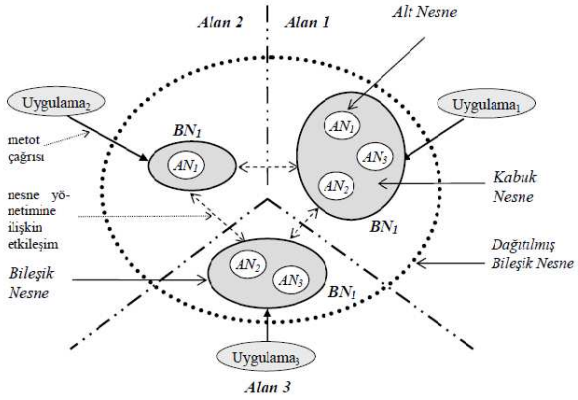
Önerilen modelde, dağıtılmış ortamdakı uygulamalar bileşik nesneler üzerinden haberleşirler ve birbirleri ile etkileşimde bulunurlar. Bu amaçla, her bir paylaşılan nesne bir arayüz üzerinden kullanılabilen bir metotlar kümesi sunar. Nesneler pasiftir. Etkinlik, nesneleri paylaşan ve onların metotları üzerinde eşzamanlı olarak çağrı yapabilen uygulama programları tarafından sağlanır. Modele göre, bir dağıtılmış bileşik nesne, Şekil 1'de sunulduğu gibi, öncelikle tekil bir adres alanı üzerinde bir veya daha fazla sayıda alt nesnenin (AN) bir kabuk nesne altında birleştirilmesi suretiyle bir bileşik nesne (BN) olarak yaratılır.

2.2 DBN modelinde nesnelerin alanlar üzerinde paylaşımı

Şekil 1'de, Alan 1 üzerinde üç alt nesnesi ile yaratılan BN_1 bileşik nesnesi diğer adres alanlarında bulunan uygulamaların yürüttükleri metot çağrıları sonucu o alanlar üzerinde, yalnızca metot çağrısının gerektirdiği alt nesneleri ile birlikte kopyalanır ve böylece, Şekil 2'de görülen nesne yapısı oluşur. Bu yapı incelendiğinde, Alan 2'de bulunan uygulamanın BN_1 üzerinde yürütmüş olduğu metot çağrıları sonucunda BN_1 bu alan üzerinde yalnızca AN_1 alt nesnesi ile ve benzer şekilde, Alan 3'de bulunan uygulamanın yürütmüş olduğu metot çağrıları sonucunda da AN_2 ve AN_3 alt nesneleri ile birlikte kopyalanmıştır. Bu suretle ortaya bir dağıtılmış bileşik nesne yapısı çıkmıştır.



Şekil 1. Tekil bir adres alanı üzerinde üç alt nesnesi ile birlikte yaratılmış bir bileşik nesne



Şekil 2. Üç adres alanı üzerine yayılmış bir dağıtılmış bileşik

2.3 DBN modelinde nesne kontrolü ve saydamlık

DBN modelinde, dağıtılmış bileşik nesneyi oluşturan alt nesnelerin farklı alanlar üzerinde rahatlıkla bağlanabilmesi amacıyla bir bağlantı nesnesi ve alt nesne kopyalarının senkronizasyon ve tutarlılığının sağlanabilmesi amacıyla da bir kontrol nesnesi her bir alt nesnenin önüne eklenmiştir. Bağlantı nesnesi ile dinamik bağlanma sağlanırken, kontrol nesnesi ile, dağıtılmış bileşik nesnenin alt nesneleri bazında düzenlenebilen çeşitli tutarlılık ve senkronizasyon mekanizmaları kurulabilmektedir. Ayrıca, dağıtılmış bileşik nesneyi oluşturan alt nesneler ve bunların yapılarını yalnızca bu nesnenin tasarımcısı bilmektedir. Tüm diğer istekçiler, nesne hakkında kendilerine sunulan istekçi arayüzünün haricinde başka bir bilgiye sahip değildirler. Bu istekçiler aynı zamanda bir metot çağrısı yürüttüklerinde bir alt nesnenin kendi alanlarına yüklenip/yüklenmediğini dahi bilmemektedirler.

2.4 DBN modelinin avantajları

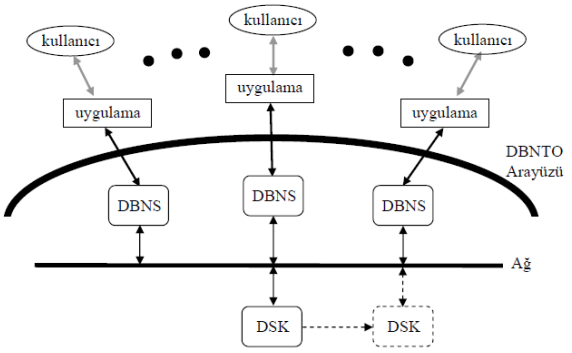
Dağıtılmış ortamdaki nesnenin alt nesnelere bölünüp, fiziksel olarak farklı adres alanlarında kopyalanmasını öngören DBN modeli ile kazanılan yararlar şu şekilde sıralanabilir:

- Kopyalama sonucu hataya hoşgörülü sistemlerin oluşturulabilmesi.
- Yalnızca alt nesnelerin kopyalanması sonucu, ağda taşınan veri miktarının azalması.
- Yalnızca hedeflenen alt nesnelerin kopyalanması sonucu, verimli bellek kullanımı.

2.5 Dağıtılmış bileşik nesne tabanlı ortam (DBNTO)

Dağıtılmış bileşik nesne tabanlı ortamın tasarımında, nesne bileşimi yönteminden yola çıkarak, mevcut Java nesne modelinin kopyalanarak fiziksel olarak dağıtılmış paylaşılan nesnelere genişletilmesi için gerekli alt yapının nasıl geliştirilebileceği konusu üzerinde yoğunlaşılmıştır. Özellikle, internet üzerinde dağıtılmış müşterek grup çalışmalarının dağıtılmış bileşik nesneler kullanılarak kolaylıkla gerçeklenebilmesini sağlayacak uygun mekanizmalar aranmıştır.

Dağıtılmış bileşik nesne modelini kullanan bir müşterek uygulamanın genel yapısı Şekil 3'te görülmektedir. Bu yapıda, dağıtılmış ortamda müşterek çalışmalar yürüten kullanıcılara ait uygulamalar DBNTO arayüzü üzerinden dağıtılmış bileşik nesneleri paylaşırlar.



Şekil 3. DBNTO sisteminin genel yapısı

Şekil 3'te yer alan her bir uygulama, Dağıtılmış Bileşik Nesne Sunucusu (DBNS) olarak adlandırılan bir sistem birimi aracılığıyla dağıtılmış bileşik nesne ortamına erişir. Ayrıca, DBNS'lerin ortama giriş/çıkış işlemlerini denetleyen, onlara sistem çapında tekil bir tanımlayıcı atayan ve yeni yaratılan bir DBN'yi oluşturan her bir alt nesne için tekil bir nesne tanımlayıcısı sağlayan, bir DBNTO Sistem Koordinatörü (DSK) yer alır. Sistemin hata-hoşgörü oranını arttırmak için, Şekil 3'te kesikli çizgilerle gösterilmiş olan ikinci bir DSK, yedek sistem koordinatörü olarak görev yapmaktadır. Herhangi bir şekilde asıl DSK'nın devre dışı kalması durumunda, yedek DSK üzerinden sistem problemsiz bir şekilde çalışmasına devam edebilmektedir.

3. DBNTO sistemi üzerinde lojistik uygulaması

Güncel bir lojistik uygulaması paylaşımlı ortamda dinamik yönetilebilme, dayanıklı, ölçeklenebilir, istekleri hızlı yanıtlama ve maliyeti minimize etmeğe yönelik optimum çözümler sunabilme gibi birçok özellikleri kendisinde bulunduracak şekilde tasarlanmalıdır [7].

DBNTO sistemi bu özellikleri taşıyacak bir uygulamanın yazılmasına izin veren bazı önemli imkanlar sunmaktadır.

Bunlardan en önemli olanları şunlardır:

- DBNTO sistemi bileşik nesne yönetimini desteklemektedir. Bu özellik bize lojistik uygulamada yer alan taşıtlara, malzemelere ve onların birtakım niteliklerini birer nesne olarak modelleme olanağı sunar. İstekte bulunan düğümlere yönelik hesaplamalarda sadece gerekli bileşik nesnelerin internet ortamında iletilmesinin mümkün olması veri yoğunluğunda azalma, paralel olarak ağ yükünün düşmesi ile daha az verinin hızlı şekilde gönderilebmesi avantajını sağlar.
- DBNTO sistemi *"yazma-güncelleme"* ve *"yazma-geçersiz kılma"* olarak adlandırılan iki tutarlılık protokolü desteklemektedir. Sistemde bu protokol tiplerinden *"yazma-güncelleme"* kullanıldığında, sunuculardaki herhangi bir nesnede değişiklik olursa, bu nesnenin sistemdeki tüm kopyaları bu değişikliği yansıtacak şekilde senkron olarak güncellenir. *"yazma-geçersiz kılma"* protokolu kullanıldığında ise sistemdeki herhangi bir nesnedeki değişiklik bu nesnenin kopyalarını kullanan tüm sunuculara bildirilir; sunucu ancak gerek duyulduğunda bu nesnenin en güncel şeklini uygun adresten yükler.

Bu özellikler bize uygulamada optimum değerleri hesaplarken düğümlerdeki (sunuculardaki) verilerin en son hallerine hızlı bir şekilde ulaşmamızı sağlar. Anlatılanlardan DBNTO sistemi üzerinde tasarlanmış uygulamamızın bize hız artımında, sistem kaynaklarının daha az kullanımında önemli katkılar sağlayacağı kanaatindeyiz.

3.1. Lojistik uygulamanın genel görünümü

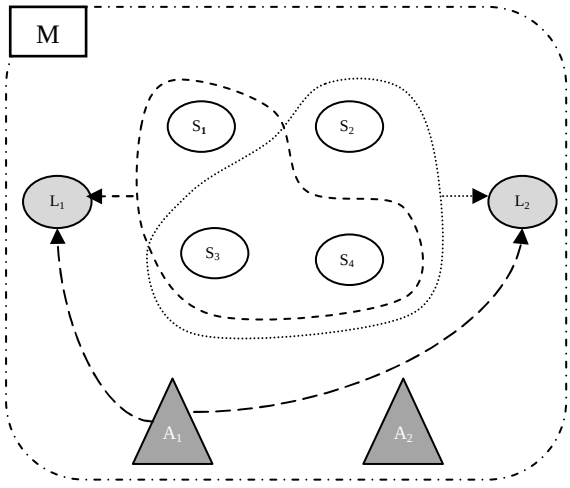
Lojistik uygulamasının DBNTO ortamı üzerinde tasarlanması için bu ortam yapısına uygun ve uygulama performansını en yüksek dereceye taşıyacak bir yol izlenmesi gerekir.
DBNTO sisteminin yapısından anlaşılacağı gibi merkezi bir düğüm üzerinde DBNTO Sistem Koordinatörü (DSK) ve diğer düğümler üzerinde DSK'nın koordine

ettiği Dağıtılmış Bileşik Nesne Sunucuları (DBNS) vardır. DBNS'ları düğümlerde çalışan farklı kullanıcı uygulamalarının DSK üzerinden haberleşmesini, veri iletimini, güncellemesini v.b. sağlar. Mutlak gereksinim olmamakla beraber DSK'nü kullanıcı programlarından her hangi birinin çalıştığı düğüm üzerinde yerleştirmemiz mümkündür (biz farklı düğümde çalışan versiyonunu inceleyeceğiz).

Düğümler arasında mesafe ölçüsünden yollar vardır. Genelde, bu yolların maliyeti iki düğüm arasında farklı yönlerde farklı değerlerde olabilir. Bu noktada ayırt edilmesi gereken bir konu da düğümler arasında haberleşme yolu (veri iletim yolu) ile mesafe ölçüsünden yol anlayışı arasındakı farktır.

Sistemde Merkez yönetici düğümden başka kullanıcı uygulamaların çalıştığı düğümler olduğunu söylemiştik. Bu düğümleri tiplerine göre aşağıdaki gibi sıralayabiliriz.

- Satıcı firmaların yerleştiği düğümler. Bu düğümlerde satışa sunulan farklı malzemeler ve onları taşıyacak farklı taşıt araçları yer alır.
- Lojistik firmalarının yerleştiği düğümler. Bu düğümlerdeki firmaların farklı satıcı firmalar ile ikili anlaşmaları vardır. Bu firmalar satıcı ve alıcı firmalar arasında yönetici/aracı rolünü üstlenirler. Anlaşmada oldukları satıcı firmaların taşıt ve malzeme kapasitesine, alıcı ile satıcı arasındaki yol mesafesine ve maliyetine göre optimal gönderim planı hazırlamak ve bunu alıcı tarafa sunmakla görevlidirler.
- Alıcı firmaların yerleştiği düğümler. Bu firmalar anlaşmak istedikleri lojistik firmalara bu yöndeki tekliflerini göndererek onlar üzerinden iş anlaşması yapabilirler.



M – merkez düğüm
L_1,L_2 – lojistik firmaları
S_1,S_2,S_3,S_4 – satıcı firmalar
A_1,A_2 – alıcı firmalar

Şekil 4. Lojistik uygulamanın haberleşme yapısı

Şekil 4'ü inceleyecek olursak, DSK görevini üstlenmiş olan M merkez düğümü görürüz. Diğer tüm düğümlerde kullanıcı uygulamalar çalışmakta ve ağ (internet) üzerinden M düğümü ile bağlantıları bulunmaktadır. Şekildeki kesik-noktalı çizgi tüm düğümlerin M düğümünün kontrolü altında olduğunu gösterir. Burada L_1,L_2 alıcı ile satıcı arasında lojistik anlaşmaları kontrol eden firmaları, S_1,S_2,S_3,S_4 satış işlemlerini yürüten firmaları, A_1,A_2 alıcı firmaları göstermektedir. L_1 lojistik firması S_1,S_3,S_4, L_2 lojistik firması ise S_2,S_3,S_4 satıcı firmaları ile anlaşmıştır.

Alıcı A_1 firması isteklerini hem L_1, hem de L_2 firmalarına gönderir. L_1 ve L_2 firmaları kendi değerlendirme ve hesaplamalarına uygun olarak ürettikleri sonucu istekte bulunan A_1 firmasına iletirler. A_1 firması ise maliyet, zaman v.b. gibi değerleri kontrol ederek kendisi için en uygun seçeneği tercih eder.

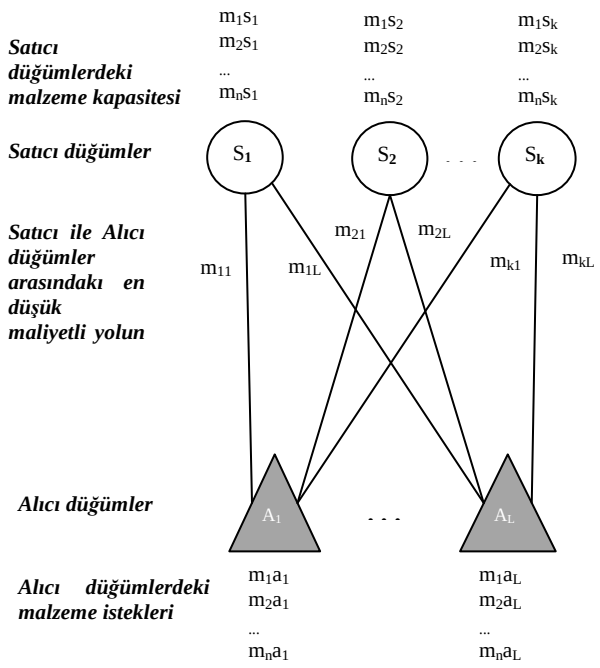
3.2. Lojistik uygulamanın tasarımında matematiksel yöntemler

Uygulamamızda lojistik firmaların optimal gönderim planı hazırlamak ve alıcılara sunmakla görevli olduğunu söylemiştik. Bu planı hazırlamak için bazı formüller ve matemetiksel yöntemlere gereksinim duymaktayız.

Uygulamanın fiziksel olarak dağıtılmış yapısı dikkate alındığında, merkez düğüm, farklı kullanıcı uygulamaların çalıştığı düğümler ve onlar arasında farklı uzunluklu yollar bulunur ve bu yapı bir ağırlıklı graf olarak temsil edilir. İlgilendiğimiz ana konulardan biri alıcının isteklerinin maliyeti en düşük olan yollarla gönderiminin yapılarak karşılanmasıdır. Alıcı isteklerinin tümünün bir tek satıcı firma tarafından karşılanamaz olması da mümkündür. Bu durumda, alıcının isteklerini değerlendiren lojistik firması, anlaşmalı olduğu satıcı firmalardan birkaçı ile işbirliği yaparak isteğin tümünü, maliyeti de en aza indirerek karşılama yoluna gidebilir. Ayrıca, iki veya daha fazla sayıda alıcı istekte bulunmuş ise, hepsinin istekleri aynı zamanda karşılanabilmelidir.

Yukarıda anlattılan koşullara uygun hesaplamaları yapmak için aşağıdaki yöntemler kullanılmıştır.

- Düğümler arasında en küçük kapsayan ağacı bulma algoritması[8]. Bu algoritma bir düğümden başka düğümlere giden en düşük maliyetli yolu bulmada kullanılacaktır.
- Satıcılar ve alıcılar arasında maliyeti minimize etmeğe yönelik Simplex algoritması ve değiştirilmiş şekli[9][10]. Bu algoritma satıcı ve alıcı düğümler arasında malzeme gönderiminin en az maliyetle çözümün belirlenmesi için kullanılacaktır.



Şekil 5. Lojistik uygulamada alıcı satıcı ilişkisi

Optimal sonucun nasıl bulunduğunu anlamak için Şekil 5'i inceleyelim. Burada alıcı ve satıcı düğümler arasındakı yol kapsayan ağaç algoritması ile bulunmuş, birçok aradüğümden geçen, maliyeti en düşük olan yoldur. Şekil 5'te karmaşıklığı azaltmak için ara düğümler gösterilmemiştir. Optimum çözümü bulmak için sistemin Simplex algoritması ile çözümlenebilecek şekle getirilmesi gerekir. Bu amaç için gerekli olan tanımlamalar ve formüller şunlardır:

- m_{ij} , $i \in (1,k)$; $j \in (1,L)$; (1)

i satıcı ve j alıcı düğümü arasındakı en düşük maliyetli yolun maliyeti.

- $m_i s_j$, $i \in (1,n)$; $j \in (1,k)$; (2)

s_j satıcı düğümündeki m_i malzemesinin kapasitesi.

- $m_i a_j$, $i \in (1,n)$; $j \in (1,L)$; (3)

a_j alıcı düğümünde m_i malzeme isteği.

Yukarıdaki tanımlamalar göz önüne alınırsa, minimize edilmesi gereken formüller ve koşulları şu şekilde ifade edilebilir:

$$\sum_{i=1}^{k} m_{ip} \sum_{j=1}^{n} m_j s_i \rightarrow \min, p \in (1, L) \qquad (4)$$

$$m_i a_p = \sum_{j=1}^{k} m_i s_j, i \in (1, n), p \in (1, L) \qquad (5)$$

(4) lineer formülü (5) koşulları kapsamında alıcıların isteklerini karşılayan minimum maliyetli yolları n bulunmasını sağlar. (5) ile ifade edilen koşullar ise alıcı isteklerinin farklı satıcı düğümlerden karşılanabileceğini ve onların toplamının alıcı isteğine eşit olması gerektiğini gösterir.

3.3. Lojistik uygulamanın çalışma adımları

Uygulamamızın DBNTO sistemi üzerinde çalışması için ilk önce dağıtılmış bileşik nesne ortamının hizmete hazır duruma getirilmesi gerekir. Bunun için ilk olarak çalıştırılacak olan birim DBNTO Sistem Koordinatörü (DSK)'dür. Koordinatör ilk olarak daha sonra yaratılacak olan sunucuların da önceden bileceği belirli bir düğümde ve belirli bir *iskele (port)* numarası ile başlatılır. Daha önce de anlatıldığı gibi, sistemin hata-hoşgörü oranını arttırmak üzere Yedek Sistem Koordinatörü de aynı anda belirli bir port numarası ile çalıştırılır.

Bir sonraki adımda sisteme katılacak olan her bir kullanıcı uygulama DBNTO sistemine giriş için *new DBNTO_Server();* komutunu yürütür. Eger sistem daha önce çalıştırılmış ve bazı işlemler yürütülmüşse, gerekli veriler kullanıcı veritabanından bileşik nesne yapısına uygun şeklinde DBNTO sistemine yüklenir.

Sistemdeki kullanıcı uygulamalar kendilerine has bir kullanıcı arayüzü üzerinden çalışırlar. Satıcı firmalarda çalışan kullanıcı uygulamalar o firmada bulunan taşıtları, malzemeleri kontrol etmek, yenilerini eklemek, lojistik firmalardan gelen istekleri incelemek, kabul/ret etmek, yeni lojistik firmaya üye olmak, üyelikten ayrılmak gibi işlemler yürütürler. Alıcı firmalardaki kullanıcı uygulamalar ise, sistemde yer alan lojistik firmaları görür, onlara isteklerini göndererek en uygun olanları ile anlaşma yapabilirler. Lojistik firmalardaki kullanıcı uygulamalar alıcı tarafın isteklerini kabul etmek, optimal sonucu bulmak, bu sonuca uygun satıcı firmalara istek göndermek, yeni satıcı ve alıcı firmalarla anlaşma yapmak v.b. gibi işlemler gerçekleştirirler.

Sisteme eklenecek herbir yeni kullanıcı uygulama DSK üzerinde kayıt yapmak zorundadır. Sisteme kayıt yapan ve sistemden ayrılan her kullanıcı bilgisi DSK tarafından tüm sistem kullanıcılarına bildirilir.

4. Sonuç

Bu çalışmada, dağıtılmış nesneye-yönelik sistemler için kopyalamaya dayalı bir model olan "dağıtılmış bileşik nesne" yöntemi ve bu yöntemi destekleyecek bir ara katman yazılımı olan "dağıtılmış bileşik nesne tabanlı ortam" sunulmuştur. Önerilen sistem yaklaşımı, farklı alanlarda bulunan çok sayıdaki kullanıcı arasında bilgi paylaşımına ve müşterek iş yürütmeye imkan sağlayan uygulamalar için paylaşılan nesneler kümesi olarak düzenlenmiş, dağıtılmış haberleşme ortamı sunmaktadır. Daha sonra bu ortam üzerinde dağıtılmış paylaşımlı sistemlere örnek olarak güncel bir lojistik uygulaması ve onun tasarım aşamaları anlatılmıştır. DBNTO sisteminin bileşik yapısı ve kopyalamaya dayalı nesne tutarlılık yönetimi, yürütme zamanını azaltmak üzere istekçiye

bileşik nesnenin yalnızca kullanılacak olan ilgili kısmının gönderilmesi sonucu çalışan uygulamanın başarımını yükseltmeğe yönelik konular incelenmiştir. Ayrıca, lojistik uygulamanın genel yapısı, farklı düğümlerde çalışan kullanıcı uygulamaların tipleri ve özellikleri açıklanmıştır. Sistemde maliyeti optimize etmek için kullanılacak olan tanımlamalar ve formüller belirlenmiş, uygulamanın çalışma adımları, birimler arasındaki etkileşim tasarlanmıştır. Söz konusu olan lojistik sistem gerçeklenme aşamasında olup, ilerki çalışmalarda sistemin yanıt verme süresi, ölçeklenebilirliği gibi konularda başarım değerlendirilmesi yapılacaktır. Dağıtılmış bileşik nesne ortamına uygun olarak tasarlanan uygulamamızın, yapısal özellikleri ve ortamın sağladığı tüm avantajlara da sahip olması nedeniyle, bu niteliklerin başarıma olumlu şekilde yansıyacağı öngörülmektedir.

5. Kaynaklar

[1] DCOBE: Distributed Composite Object-Based Environment, Yilmaz and Erdogan *The Computer Journal.*2005; 48: 273-291

[2] Dağıtılmış nesneye dayalı sistemler için dağıtılmış bileşik nesne modeli ,*itüdergisi/d mühendislik ,Güray YILMAZ, Nadia ERDOĞAN, Cilt:1 Sayı:1 Ağustos 2002*

[3] Remote Procedure Calls (RPC) - A tutorial on ONC RPC,Cardiff University, Dave Marshall 1999

[4] *Parallel Programming Techniques & Applications Using Networked Workstations & Parallel Computers 2nd ed.*, by B. Wilkinson & M. Allen, 2004 Pearson Education Inc.

[5] An Object-Based Approach to Programming Distributed Systems, Andrew S. Tanenbaum,Henri E. Bal,Saniya Ben Hassen Dept. of Mathematics and Computer Science, Vrije Universiteit Amsterdam, The Netherlands,M. Frans Kaashoek Laboratory for Computer Science, M.I.T. Cambridge, MA

[6] Advantages of using the object-oriented paradigm for designing and developing software, Applied Computing, Mathematics and Statistics group, Applied Management and Computing Division, Box 84, Lincoln University,Lincoln, Canterbury, New Zealand 2001

[7] Features of Logistics planning software, http://www.logistics-management-software-guide.com/

[8] Dijkstra's algorithm revisited: the dynamic programming connexion, Sniedovich. M, 2006

[9] Simplex Algorithm Advanced Algorithms, Tamal K. Dey April 30, 2009

[10] Smoothed Analysis of Algorithms: Why the Simplex Algorithm Usually Takes Polynomial Time, Shang-Hua Teng, Journal of the ACM, Vol. 51, No. 3, May 2004, pp. 385–463.