Yeni Nesil Yazılım Sistemlerinde Yedekleme Tabanlı Aksaklığa Dayanıklılık

Şebnem Bora ve Oğuz Dikenelli

Bilgisayar Mühendisliği Bölümü, Ege Üniversitesi, İzmir

e-posta: sebnem.bora, oguz.dikenelli@ege.edu.tr

Özetçe

Aksaklığa dayanıklılık yöntemleri, grid ve çok-etmenli sistemler gibi yeni nesil yazılım sistemlerinde de kullanılmaktadır. Bu çalışmada çok_etmenli sistemlerde yedeklemeye dayalı aksaklığa dayanıklılık (fault tolerance) politikalarının uygulanabilmesi için çok-etmenli sistem geliştirme platformunda bulunması gereken servisler anlatılmış ve bu alanda aksaklığa dayanıklılık ile ilgili yapılan çalışmalar verilmiştir.

1. Giriş

Yedekleme, dağıtık ve dinamik ortamlarda hatanın yedeklenen kopyada maskelenerek aksaklığa dayanıklılığı sağlayan anahtar bir tekniktir. Dağıtık bir uygulama birbirleriyle işbirliği içinde olan nesnelerin kümesidir. Burada nesne bir sanal düğüm, bir süreç, bir değişken, nesne-yönelimli programlamadaki anlamıyla bir nesne ve çok-etmenli sistemlerdeki bir etmen olabilir. Bir nesne yedeklendiğinde uygulama, aynı nesnenin birkaç eş kopyasına (yedek) sahip olacaktır. Bir yedekte oluşabilecek bir aksaklık sistemdeki diğer yedeklerle maskelenecek, böylece oluşan aksaklığa rağmen kullanılabilirlik sağlanacaktır.

Uygulamada kopyalayarak yedekleme sisteme yük getiren bir yöntemdir. Yedeklenen sistem bileşenlerinin (replica) sistemin yaptığı işe dahil olması ve koordinasyonun sağlanması sırasında yapılan iletişim, sisteme çalışırken fazladan yük yüklemektedir. Bunun yanında sistemdeki bazı bileşenlerin kritikliği, sistem içindeki öneminin veya diğer bileşenlerin bu bileşenlere bağımlılığının derecesinin değişiminden dolayı, zamanla değişebilmektedir. Bu nedenle güvenilebilirlik protokolleri sisteme gerektiği yerde ve zamanda uygulanmalıdır. Başka bir deyişle, uygulamanın çalışması sırasında önem taşıyan bileşenlerin aksaklığa dayanıklı (fault-tolerant) yapılması ve bunu yaparken de kullanılan tekniğin dikkatli seçilmesi gerekmektedir.

Yedekleme her ne kadar işlem ve iletişim açısından uygulandığı sisteme fazladan yük getirse de, aksaklığa dayanıklılık açısından en çok tercih edilen yöntemlerden biridir. Aksaklığa dayanıklılık yöntemleri, grid ve çok-etmenli sistemler gibi yeni nesil yazılım sistemlerinde de kullanılmaktadır. Özellikle yedekleme yöntemi, çok-etmenli sistemlerde aksaklığa dayanıklılığı sağlamak açısından kullanılmıştır. Bu bildiride aksaklığa dayanıklılığı geliştirmek için, çok-etmenli sistemlerde yedekleme yaklaşımının kullanılması anlatılmış ve bu alanda yapılan çalışmalar incelenmiştir.

Literatürde etmenler ve çok-etmenli sistemler için pek çok tanım yapılmıştır. Bu tanımlar, farklılıklarına rağmen, çok-etmenli sistemi birbirleriyle işbirliği veya rekabet amacıyla etkileşen ve bazen özerk olarak kendi hedeflerine ulaşmaya çalışan yazılım etmenlerinin oluşturduğu ortam olarak karakterize etmektedirler. Çok-etmenli sistemler ortamın kaynaklarını ve servislerini kullanmakta ve bazı zamanlar etmenlerin başlattıkları varlıklar için sonuç üretmektedirler. Etmenler asenkron ve dağıtılmış bir şekilde etkileştikçe bu tip bir sistem, karmaşık bir sistem (complex system) haline dönüşür [1].

Merkezi bir koordinasyonun olmaması etmenlerin davranışlarını dolayısıyla etmenlerin oluşturduğu sistemin davranışını tahmin etmemizi zorlaştıracaktır. Bu durumda "Aksaklığa dayanıklılığın nasıl sağlanacağı" önemli bir sorun olarak karşımıza çıkar. Bunun için etmenlerin özerkliğini göz önünde bulundurup aksaklığa dayanıklılık için çok-etmenli sistem seviyesinde bazı kurallar tanımlamak gerekmektedir. Dinamik ve geniş ölçekli ortamlarda yedekleme ile ilgili parametreleri elle belirlemek kolay olmadığı için yedeklenecek olan etmen ve yedekleme derecesi dinamik ve otomatik olarak belirlenmelidir. Bu bildiride, bu gibi geniş-ölçekli sistemlerde aksaklığa dayanıklılık yöntemlerinden yedekleme yaklaşımının nasıl gerçekleneceği konusunda bilgi verilmektedir.

Sonraki bölümler şu şekilde organize edilmiştir: Bölüm 2'de aksaklığa dayanıklılık ile ilgili temel kavramlar verilmiştir; Bölüm 3'de aksaklığa dayanıklılık için çok-etmenli sistemlerde kullanılan temel yapılar anlatılmıştır; Bölüm 4'de ise çok etmenli sistemlerde aksaklığa dayanıklılık için yapılmış çalışmalar verilmiştir; Bölüm 5'de ise bu çalışmanın özeti verilmiştir.

2. Aksaklığa Dayanıklılık

Çok etmenli sistemlerde aksaklığa dayanıklılığın ne olduğu ve nasıl geliştirilebileceği hakkında fikir sahibi olunabilmesi için aksaklığa dayanıklılığa ilişkin bazı temel kavramların verilmesi gerekmektedir.

Aksaklığa dayanıklı olmak, güvenilebilir (dependable) sistemlerle yakından ilişkilidir. Güvenilebilirlik bir sistemin işlevsel olma ve kendisinden beklenen servisleri beklenen bir anda sağlaması olasılığıdır. Bir sistem kendisinden beklenenleri karşılayamadığında sistem arızalanmıştır denmektedir. Bu açıdan bakıldığında hata, sistemin arızalanmasına sebep olabilecek bir sistem durumudur. Hatanın nedeni ise aksaklık olarak adlandırılmaktadır.

Karşılaşabileceğimiz aksaklıklar; yazılım testleri yapılmasına rağmen program içerisindeki görülemeyen yanlışlar, önsezilemeyen sistem durumları (program belirli durumları işleyemiyor), sistemin çökmesine neden olan işlemcideki aksaklıklar veya kaynakların sınırlı olmasıdır. Bunun yanında iletişim hatlarının yavaşlığı veya iletişim hatlarındaki problemler de karşılaşılabilecek aksaklıklardan bazılarıdır.

Güvenilebilir sistemleri inşa etmek, aksaklıkları kontrol etme ile yakından ilgilidir. Aksaklıkları önleme, aksaklıkları ortadan kaldırma, aksaklıkları tahmin etme ve aksaklığa dayanıklılık, sistemleri güvenilebilir yapma açısından önemlidir; aksaklıkları kontrol altında tutmamıza yarar. Bu bildiride çok etmenli sistemlerin güvenilebilirliği aksaklığa dayanıklılık açısından ele alınmaktadır.

Aksaklığa dayanıklılık, bir sistemin aksaklıkların mevcut olması durumunda bile servisini vermeye devam etmesi demektir. Aksaklığa dayanıklı bir sistem kendisinden beklenenleri yapamadığı takdirde bunu diğer süreçlerden saklayan sistemdir. Aksaklıkları maskelemenin püf noktası ise "tekrar" kavramını kullanmaktır. Fiziksel tekrar ile sisteme, bazı elemanların yanlış çalışmasını veya kaybı tolere etmesi için fazladan ekipman veya süreç eklenmektedir. Fiziksel tekrar donanımsal veya yazılımsal yapılabilmektedir. Örneğin sisteme fazladan süreç eklendiğinde eklenen süreçlerin küçük bir kısmı çökse de sistem doğru çalışmaya devam edecektir. Diğer bir deyişle süreçlerin kopyalanarak yedeklenmesi, aksaklığa dayanıklılığı arttırmaktadır [2].

2.1. Yedekleme

Yedekleme yönteminde , bir nesnenin yedekleri grup içinde organize edilmektedir. Bir grup, grubun üyeleri olan yedeklenmiş nesnelerin oluşturduğu kümedir. Bir gruptaki yedeklerin koordinasyonu için iki temel teknik aşağıda verilmektedir:

- Aktif Yedekleme: Tüm yedeklerin tüm gelen mesajları paralel olarak işlediği bir tekniktir. Bu yöntemde yedeklerin içsel durumları senkronize durumdadır. Çıkış herhangi bir yedekten alınabilir.
- Pasif Yedekleme: Sadece bir tek yedeğin (birincil) gelen mesajları işlediği ve sonucu sağladığı bir tekniktir. Diğer yedekler bu durumda aktif olmadan beklemektedirler. Ancak içsel durumları belirli aralıklarla güncellenmektedir.

Her bir stratejinin kendine göre avantaj ve dezavantajları bulunmaktadır. Aktif yedeklemede iyileşme hızlıdır. Ancak tüm yedeklerin gelen mesajları işleme koymalarından dolayı her yedek kendi bulunduğu bilgisayardan işlemsel olarak beslenmekte ve iletişim yükü katlanmaktadır. Aktif yedekleme yaklaşımı kritik uygulamalarda ve özellikle kısa iyileşme süresinin istendiği gerçek zamanlı uygulamalarda tercih edilmektedir. Pasif yedeklemede, hatasız çalışmada maliyet düşüktür fakat iyileşme süresi yeterince kısa değildir. En uygun yaklaşımın seçimi doğrudan çevresel faktörlere; özellikle hata oranı, hatanın cinsi ve iyileşme zamanı, ek yük gibi uygulamanın gereksinimlerine bağlıdır. Aktif yedekleme, hata oranı çok yükseldiğinde veya uygulamada zaman kısıtlaması önem taşıyorsa tercih edilmeli, diğer zamanlarda pasif yedekleme tercih edilmelidir. Özellikle aktif yedeklemede uygulama belirlenimci olmayan bir kaynakta (yani verilen girdiye ve kaynağın durumuna göre çıkışından elde edilen sonuç önceden bilinen kaynak) çalışıyorsa bu durumda yedekler arasında tutarlılığın kesin sağlanacağı konusunda garanti verilemeyecektir.

Yarı-aktif yedekleme, hem aktif hem pasif yedeklemenin avantajlarını bünyesinde barındırmaktadır. Bunlar belirlenimci olmayan süreçlerin çalışması ve iyileşme zamanlarının kısaltılması şeklindeki avantajlardır. İstemciden gelen mesajlar lider tarafından diğer yedeklere dağıtılır ve bu mesajlar yedekler tarafından tek tek işleme konulur. Sonuç sadece istemciye lider tarafından döndürülür. Bu teknik iki tekniğin avantajlarını birleştirdiği için ilginç bir yaklaşımdır.

Yedekleme yaklaşımları grup içinde yedekler arasında koordinasyonu sağlamaktadır. Bununla birlikte yedekleme derecesi (gruptaki yedeklerin sayısı), yedekleme yaklaşımına dayalı aksaklığa dayanıklılık politikalarının uygulanması açısından önemli bir kavramdır. Ancak çalışma zamanında yedek sayısına nasıl karar verileceği konusu, karşımıza problem olarak çıkmaktadır. Yedekleme derecesi, statik ve dinamik olarak belirlenebilmektedir. Statik aksaklığa dayanıklılık politikalarında bu sayı, uygulama başlamadan önce programcı tarafından belirlenir. Adaptif aksaklığa dayanıklılık politikalarında bir yedekleme yöneticisi, gruptaki yedek sayısına sistemdeki kaynakların durumuna göre karar vermektedir. Bunun için bir gözlem mekanizması, aksaklığa dayanıklı hale getirilecek sistemin bileşenlerinin davranışlarını, kaynakların kullanılabilirliklerini gözlemleyecek ve

sistem kaynaklarını adaptif olarak yeniden düzenleyecektir.

Yedekleme tekniklerinin kullanıldığı aksaklığa dayanıklı sistemlerde, yedeklemeyi desteklemek amacıyla küresel zaman servisi, yedekleme servisi, grup iletişimi servisi ve üyelik servisi gibi çeşitli servisler üzerine pek çok araştırma yapılmıştır. Bu servisler çok etmenli sistemlerde aktif, yarı-aktif ve pasif yedekleme stratejilerini destekleyen alt yapıyı oluşturmaktadırlar. Bir sonraki bölümde bu servislerin çok-etmenli sistemler için nasıl geliştirilebileceği anlatılacaktır. Bu servislerin destekleyeceği sistemin, asenkron bir ortamda ve mesaj ihmallerine, etmen çökmelerine ve ağ ayrılmalarına tabi olduğu ve mesaj bozulmaları ve rastlantısal aksaklıklarla karşılaşılmayacağı varsayılmaktadır.

3. Çok Etmenli Sistemlerde Aksaklığa Dayanıklılık Servislerinin Gerçeklenmesi

Bu bölümde, yedeklemeye dayalı aksaklığa dayanıklılık politikalarının uygulanması için gerekli olan alt yapılar, hedef-yönelimli çok-etmenli sistem mimarisi üzerinde anlatılacaktır.

Hedef-yönelimli bir etmenin en önemli birimleri onun hedef yöneticisi ve planlayıcısıdır. Bir hedef yöneticisi gelen mesajdan hedefi belirler ve bu hedefin planını (kısmen sıralı basit eylemler dizisi) başlatır. Planlayıcı ise zamanlamayı yapar ve bir planlama formundaki planı çalıştırır [3].

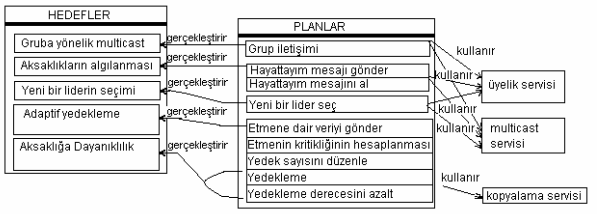
Aksaklığa dayanıklılığı arttırmak için gerekli olan servisler, hedef yönelimli çok-etmenli sistem mimarisine entegre etmek istendiğinde, aksaklığa dayanıklılık gereksinimlerinden ortaya çıkan hedefler, bu hedefleri gerçekleyecek planlar ve bu hedeflerin planları tarafından kullanılabilecek yeniden kullanılır servisler belirlenmelidir.

Çok-etmenli sistemlerde yedeklemeye dayalı aksaklığa dayanıklılık politikasını gerçekleştirebilmek için, etmenlerin bu politikaya yönelik belirlenen hedefleri gerçekleştirmeleri gerekmektedir. Bu hedefler "Gruba yönelik multicast", "Aksaklıkların algılanması", "Yeni bir liderin seçimi", "Adaptif Yedekleme ve "Aksaklığa Dayanıklılık" olmak üzere belirlenmiş ve Şekil 1.'de verilmiştir. Belirlenen bu hedeflerin gerçekleştirilmesi için, planları ve bu planlar tarafından kullanılan yeniden kullanılır servislerin gerçeklenmesi gerekmektedir.

Yeniden kullanılır servislerden biri üyelik servisidir. Bu servis gruptaki yedeklerin listesini her bir etmenin bilgi tabanında tutmaktadır. Üyelik servisi, etmenlerin grup üyelikleri konusunda bir karara varabilmek için aksaklık algılayıcı servisle birlikte çalışmaktadır. Üyelik servisince tutulan listedeki grup üyelerine, gruba gelen mesajlar güvenilir "multicast" servisi tarafından iletilir.

Aksaklık algılama, sistemdeki aksak bileşeni tanımlayabilmek açısından önem taşımaktadır. Yaklaşımımızda, "Hayattayım Mesajı Gönder" ve "Hayattayım Mesajını Al" planları "aksaklıkların algılanması" hedefine ulaşmak için çalıştırılacak olan planlardır. "Hayattayım Mesajı Gönder" planında, gruptaki her bir yedek periyodik olarak gruptaki diğer üyelere "hayattayım" mesajı göndermektedir.

Gönderilen "Hayattayım mesajını alan her bir etmen, "Hayattayım Mesajını Al" planını çalıştıracaktır. Bu planda etmen, "Hayattayım" mesajını aldığı her etmenden yeni bir "Hayattayım" mesajı bekleyecektir. Yeni bir "Hayattayım" mesajı belirli bir zaman aşımı süresince alınmadığı takdirde, mesaj alınamayan etmen "şüpheli listesine" eklenilecektir. Eğer tüm yedekler bu etmenin "Hayattayım" mesajını zaman aşımı süresince alamadılarsa, bu durumda etmenler "şüpheli listelerinde" bu durumu belirteceklerdir ve aynı durumun yinelenmesiyle bu yedeği üyelik listesinden sileceklerdir.



Şekil 1. Aksaklığa Dayanıklı Çok-etmenli sistemdeki hedef ler

Dağıtık sistemlerde, bozulmuş bir süreci tolere etmek için en iyi yaklaşım bir kaç eş sürecin bir grup içinde organize edilmesidir. Tüm grupların en önemli özelliği ise bir gruba mesaj gönderildiğinde tüm grup üyelerinin bu mesajı alabilmeleridir. Bu yolla gruptaki bir sürecin çalışması aksarsa diğer süreçler aksayan sürecin kaldığı yerden işleme devam edebilirler. Yani yedekler arasındaki koordinasyon ve tutarlılık, "multicast" yapılabilmesini sağlayan grup iletişim servisi tarafından desteklenmektedir. Biz çok-etmenli sistem geliştirme çerçevesinin doğal olarak "multicast" sağladığını varsaymaktayız. Bu özellik grup iletişiminin gerçeklenmesinde kullanılmaktadır. Ancak "multicast" mekanizması sadece gelen mesajların keyfi bir sırada dağıtılmasını sağlamaktadır.

"Multicast"daki sıralama, aksaklığa dayanıklı sistemler geliştirebilmek için desteklenmelidir; çünkü yedekler arasındaki koordinasyon ve tutarlılık, gelen isteklerin sırayla gerçekleştirilmesiyle sağlanmaktadır. Sıralama "Gruba yönelik multicast" hedefinin "grup iletişimi" planında gerçeklenmektedir. Bu planda, başka bir etmenden gelen her mesaja gruba özel, artan bir sayı verilmektedir ve daha sonra bu mesajlar üyelik listesindeki tüm yedeklere gönderilmektedir.

Yedekleme servisi yedekleme tabanlı aksaklığa dayanıklılık yöntemlerinin en önemli parçası olmaktadır. Bu durumda yedekleme servisinin yardımıyla, her etmen bir çok kez ve farklı yedekleme stratejileri ile yedeklenebilir. Her bir grubun yedek grubunu koordine eden ve diğer etmenlerle iletişim kuran bir lideri vardır. Bir lider arızalandığında yedek grubundaki bir yedek, yeni lider olarak seçilir. Bir liderin arızalanmasına yönelik önlem almayı sağlayan "yeni bir liderin seçimi" hedefi ve "yeni bir lider seç" planı bu amaç için gerçeklenmektedir.

İster statik aksaklığa dayanıklılık ister adaptif aksaklığa dayanıklılık politikası tercih edilsin, bir etmenin aksaklığa dayanıklılık politikasını uygulayabilmesi için aksaklık algılayıcı servisinden bir aksaklık raporu alması gerekmektedir. Aksaklık algılayıcı aracılığıyla şüphelenilen bir etmen üyelikten çıkarıldıktan sonra "Aksaklığa dayanıklılık" içerikli mesaj grup içinde "multicast" edilir. Bu aksaklık mesajı alındığında, lider etmenin hedef yöneticisi tarafından aksaklığa dayanıklılık hedefiyle eşleştirilir.

Statik aksaklığa dayanıklılık politikasının tercih edilmesi söz konusuysa yedekleme derecesi ve yedekleme stratejisi etmen ilk yaratıldığında belirlenmektedir ya da adaptif aksaklığa dayanıklılık politikası için yedek sayısı bilinen herhangi bir değer olacaktır. Fakat bir grubun yedekleme derecesi ve stratejisi, içeriğinde yeni yedekleme derecesi ve stratejisi olan bir istek ile değiştirilebilmektedir. "Aksaklığa Dayanıklılık" içerikli bir mesaj alındığında "aksaklığa dayanıklılık" hedefi eşleşecek ve bu durumda lider konumundaki etmen grubun eski yedekleme derecesini korumak isteyecek "Yedek sayısını düzenle" planını işleyecektir. Bu planın görevlerinde, lider üyelik listesini kontrol ederek kaç yedeğin yaratılacağını anlamaya çalışır. Olması gereken yedek sayısı ile o anki üyelik veri yapısının büyüklüğü arasındaki fark, yaratılması gereken yedek sayısı olacaktır ve başlangıçta belirlenen yedek sayısına ulaşana kadar yeni yedeklerin yaratılacağı "yedekleme" planı lider etmen tarafından çalıştırılacaktır. Doğal olarak yedekler ve liderler aynı planları işlemeyeceklerdir. Bir etmen başlatıldığında etmenin tipi lider, yedek veya normal olarak belirlenmektedir. Böylece etmen belirlenen tipe göre planlarını işleyecektir.

Eğer aksaklığa dayanıklılık çok-etmenli sisteme adaptif bir şekilde uygulanacaksa bu sistemdeki kritik etmenler, bu etmenlerin yedeklenebilmeleri için belirlenmelidir. Kritik etmenleri belirleyebilmek için ortamdan etmenin diğer etmenler ile iletişimi ve etkileşimi ile ilgili bilgi toplamak gerekmektedir. Daha sonra toplanan bu bilgi etmen kritikliğini değerlendirmek amacıyla kullanılacaktır. Etmen kritikliği, diğer etmenlerin belirli bir etmene olan bağımlılıklarını ve o etmenin çok-etmenli sistemdeki önemini göstermektedir. Etmenin kritikliğine ilişkin bilgi toplamak ve değerlendirmek amacıyla "Adaptif yedekleme" hedefi belirlenir ve "Etmene dair veriyi gönder", "Etmenin kritikliğinin hesaplanması", "Yedek sayısının hesaplanması", "Yedek sayısını düzenle" ve "Yedekleme" planları bu hedefin ulaşılmasını sağlar. "Etmene dair veriyi gönder" planında etmen sistemindeki diğer etmenler ve etmen grubunun lideri bir "gözlemci" etmene kritikliklerinin değerlendirilmesine ilişkin veri göndermektedir. Bu "gözlemci" etmen sistemdeki tüm etmenlerden gelen veriyi değerlendirir ve topladığı verilerden elde ettiği bilgiyi bir mesaj içerisinde tüm etmenlere gönderir. Bu mesajı alan lider etmen "Etmenin kritikliğinin hesaplanması" planında kritikliğini hesaplar ve "Yedek sayısının hesaplanması" planıyla mevcut sayıdaki kaynaklara ve etmenin kritikliğine göre etmen grubunun yeni yedekleme derecesini belirler. Son olarak "Yedek sayısını düzenle" ve "Yedekleme" planları ile bu hedefe ulaşılması tamamlanır. Eğer hesaplanan yedekleme derecesine göre yeni yedek yaratmak gerekiyorsa kopyalama servisi kullanılarak yedekleme işlemi yapılır. Bazen yedekleme derecesi o anki etmen kritikliğine göre fazla çıkabilir ve varolan bazı etmenleri gruptan çıkarmak ve bitirmek gerekebilir. Bu durumda "yedekleme derecesini azalt" planı işletilmelidir.

"Yedekleme" planında, yeni yedeğin yerleştirileceği bilgisayardaki kopyalama sunucusu ile RMI (Remote Message Invocation) aracılığıyla ile ilişki kurulmaktadır. Uzaktaki bilgisayarda yeni bir yedeğin oluşturulabilmesi için öncelikle kopyalama sunucusu ve etmen geliştirme platformunun hazır olarak beklemeleri gerekmektedir. Kopyalama sunucusuna RMI aracılığıyla gönderilmeden önce, etmenin içsel durumu serileştirilmiş nesne halinde bir dosyaya yazdırılır. Daha sonra yedeklemeyi gerçekleştirmek için gerekli olan dosyalar halindeki etmen bilgisi (etmenin kodu, etmenin içsel durumunu oluşturan veri yapıları, planları) byte dizinini transfer eden RMI "mesaj" metodu kullanılarak uzaktaki bilgisayara transfer edilir. Özetle RMI "mesaj" metodu kullanılarak eylem ve planların bulunduğu kütüphaneler, bu kütüphanelerin kopyalanacağı dizinlerin yolu (path), ve byte dizini halindeki etmen iç durumu gönderilmektedirler.

Bir kopyalama sunucusuna orijinal etmenden RMI mesaj metoduyla ulaşıldığında kopyalama sunucusu kütüphaneleri, etmenin kaynak kodunu belirtilen dizinlere kopyalar ve etmenin kaynak kodunu çalıştırır. Yedeklenmiş etmen çalışmaya başladığında, yedek olarak amaçlarını gerçekleştirmek için hazır hale gelir. Yedeklenen etmene ilişkin tek kısıtlama çalıştırma modunun yedek olmasından dolayı kendini kopyalayamamasıdır. Ancak lider etmen çöktüğünde yedek etmen yeni lider seçildiği takdirde kopyalama ve diğer etmenleri yanıtlama yeteneğine sahip olacaktır.

4. Çok Etmenli Sistemlerde Aksaklığa Dayanıklılık Konusunda Yapılan Çalışmalar

Aksaklığa dayanıklılık, çok etmenli sistemlerde yeni sayılabilecek bir alan olarak karşımıza çıkmaktadır. Bazı çok-etmenli sistem geliştirme platformlarında aksaklığa yönelik çözümler önerilse de, bunların çoğu probleme yönelik olmaktadır. Örneğin bazı projelerde etmen işbirliğini sürdürmeye ilişkin problemler [4], bazılarında ise mobil etmeninin yer değiştirmesine ilişkin problemler [5] ele alınmıştır.

Kumar ve arkadaşları komisyoncu (broker) aksaklıklarını tolere edebilecek güvenilir bir komisyon mimarisini tanımlayan bir yöntem sunmuşlardır [4]. Bu yöntem "teamwork" kavramına dayalıdır ve bir takımda yeralan komisyoncular hiyerarşik bir şekilde dizilmektedirler. Komisyoncular aralarında bilgi değişimi yaparken aynı zamanda etmenler arasında iletişimi de sağlamaktadırlar. Onların bu yöntemi etmenler için değil komisyoncular dan oluşan aksaklıkları gidermek için önerilen bir yöntemdir.

Klein etmen sistemlerine uyumlu ek olabilen (plug-in) ve paylaşılabilen bir kural dışı durum işleme (exception handling) servisine dayalı bir yaklaşım sunmuştur [6]. Bu servis çok etmenli bir sistemde, sistemdeki gelişmeleri takip etme amacıyla kullanılmaktadır. Yeni bir etmen yaratıldığında "yeni etmen kayıtlama" etmeni bu yaratılan etmenin normal davranışlarına ait tanımı kaydeder ve bu etmenin normal olmayan davranışlarını bulup çıkaracak gözcüler (sentinel) yaratır. Gözcü, aksaklığa ait bir semptomu sezdiği takdirde bu bilgiyi teşhis (diagnosis) etmenine gönderir. Teşhis etmeni bir kaç muhtemel teşhis listesini, bu listeye göre yapılması gereken doğru eylemleri belirleyecek olan çözüm etmenine gönderir.

Hagg çalışmasında bir takım özel fonksiyonları veya etmen topluluğundaki bazı özel durumları koruyacak gözcü (sentinel) etmenleri kullanmıştır [7]. Bu gözcüler semantik adresleme kullanarak diğer etmenlerle etkileşirler. Böylece etmen iletişimini ve etkileşimini kontrol ederek diğer etmenlerin modellerini oluştururlar. Bunun yanında zamanlayıcı kullanarak etmen ve iletişim aksaklıklarını algılarlar.

Yukarıda anlatılan çalışmaların dışında yedekleme mekanizmaları kullanarak çok etmenli sistemlerdeki aksaklığa dayanıklılığı sağlayacak yöntemler de bulunmaktadır. Örneğin, Fedoruk ve Deters çok etmenli sistemlerde aksaklığa dayanıklılığı geliştirmek ve kullanılabilirliği ve güvenilirliği arttırmak için "proxy"i kullanarak şeffaf yedeklemeyi gerçekleştirmişlerdir [8]. "Proxy" bir arayüz olarak çok etmenli sistemlerde, yedekler ve diğer etmenler arasında tüm iletişimi sağlamaktan sorumludur. "Proxy " aynı zamanda yedeklerin oluşturduğu grupta işlemleri yürütmekten ve durum yönetiminden (state management) sorumludurlar. Bu çalışmada "proxy"ler tüm aksaklığa dayanıklılık işlerinden sorumlu olsalarda kendileri aslında aksaklığa açıktırlar. Bu çalışmada "proxy"nin arızalanması durumunda aksaklığa dayanıklılığa yönelik herhangi bir yöntem gerçekleştirilmemiştir. Fedoruk ve Deters önerdikleri yöntemi FIPA-OS platformunu kullanarak gerçekleştirmeye çalışmışlardır. Bu yöntemde yedekleme çalışma zamanından önce yapılmış olup aksaklığa dayanıklılık politikasını, yedekleme stratejisini veya yedek sayısını sonradan değiştirmek ve aksaklığa dayanıklılık ile ilgili servislerde değişiklik yapmak mümkün olamayacağı için önerdikleri yöntem yeniden kullanılabilirlik ve esneklik açısından fakirdir.

Guessoum ve arkadaşları etmen ve organizasyon seviyesinde adaptasyon sağlayan adaptif çok etmenli bir sistem gerçekleştirmişlerdir [9], [10]. Bu organizasyonda sistem, sürekli kontrol edilerek çok etmenli sitemdeki kaynaklar ve aksaklığa ilişkin bilgi edinilmiştir. Bu mimari DIMA [11] ve DarX "middleware" [12] kullanılarak gerçekleştirilmiştir. DarX ile yazılım bileşenleri gerek yedeklenebilir gerekse yedeklenmeyebilir, çalışma zamanında yedekleme mekanizması değiştirilebilir.

Yukarıdaki çalışmalar şüphesiz çok etmenli sistemlerde aksaklığa dayanıklılığı geliştirmek için faydalı yöntemlerdir. Ancak kullanılan yazılım varlıkları çok etmenli sistemlerde aksaklığa dayanıklılık problemini çözmek için esneklik ve yeniden kullanılabilirlik açısından insan kullanıcılarına kolaylık sağlamaktan uzak bulunmaktadırlar. Örneğin, DarX "middleware"i veya Fedoruk'un çalışması bir sistemi aksaklığa dayanıklı hale getirmek için önceden bu sistemin bulunduğu ortama yüklenmelidir. Bu durumda DarX'ı veya Fedoruk'un çalışmasını sistemin başlangıcında yüklemeden bir etmenin aksaklığa dayanıklı olması söz konusu olamamaktadır. Bu şu anlama gelmektedir; bir etmen eğer aksaklığa dayanıklı olsun isteniyorsa aksaklığa dayanıklı bir şekilde başlatılmalıdır, sonradan bu etmenin aksaklığa dayanıklı olması mümkün değildir.

DimaX, DIMA çok etmenli sistem geliştirme platformu ile yukarıda anlatılan DarX "middleware"'inin entegrasyonu sonucu oluşan aksaklığa dayanıklı çok etmenli bir sistem geliştirme platformudur [13]. DimaX, sistem (DarX), uygulama (etmenler) ve gözlemleme olmak üzere 3 seviyeden oluşmaktadır. Uygulama düzeyinde DIMA, çok etmenli uygulamalar için gerekli kütüphaneleri sağlamaktadır. Bunun yanında DarX, etmenleri dağıtmak ve kopyalamak için gerekli mekanizmaları sunmaktadır. Gözlemlemeyi yapan gözlem mekanizması ise hem uygulama hem de "middleware" düzeyinde çalışmaktadır ve yedeklemenin otomatik olarak yapılmasından sorumludur. DimaX

isimlendirme (white pages), aksaklık bulma, gözlemleme ve yedekleme servislerini sağlamaktadır.

DimaX aksaklık modeli olarak çökme modelini benimsemiş olup, etmenlerin belirlenimci olduğunu varsayarak aktif veya pasif yedeklemeyle yedek gruplarını organize etmek gibi iki seçeneği kullanıcıya sunmaktadır.

Gözlemleme mekanizması sistem ve uygulama düzeyinde bilgi toplayarak yedeklemeyi kontrol etmektedir. Bunun için bilgisayar gözlemcisi isimli karşıt-eylemli etmenleri kullanılarak yerel bilgiyi değerlendirmek için gözlem verisi toplanmakta ve işlenmektedir. Aynı zamanda küresel bilgiyi oluşturmak için bilgisayar gözlemcileri aralarında yerel bilgilerini değiştirmektedirler. Daha sonra toplanan bu veriler, her bir etmenle ilişkilendirilen gözlemleme etmenine belirli aralıklarla gönderilmektedir. Her bir gözlemleme etmeni sadece bir bilgisayar gözlemcisi ile iletişim kurabilmektedir. Gözlemleme etmeni belirli aralıklarla kendisine gelen veriyi kullanarak ilişkilendirildiği etmenin kritikliğini hesaplamaktadır. Gözlemleme etmeninin kullandığı veriler etmenin rolü ve iletişim yükü ve aynı zamanda etmen organizasyonunun toplam iletişim yüküdür. Bu verilerin kullanılması ile hesaplanan etmen kritikliği, daha sonraki adımda etmenin kullanılabilir kaynaklardan payını belirlemekte kullanılmaktadır.

Yukarıdaki çalışmada, DarX aksaklığa dayanıklılık "middleware"i ve Dima çok etmenli sistem geliştirme platformu birleştirilerek aksaklığa dayanıklı çok etmenli sistem geliştirme platformu DimaX elde edilmiştir. Aksaklığa dayanıklılık için gerekli servisler DarX aracılığıyla DimaX kullanıcılarına sunulmuştur. DimaX aynı zamanda sunduğu hiyerarşik gözlemleme mekanizması ile yedeklemeyi otomatik hale getirerek etmen yedek gruplarının yedekleme derecesini belirleyebilmektedir. Yedek gruplarının yedekleme derecesini belirlemekte kullanılan etmen iletişim yükü ve etmenin rolü gibi bilgiler gözlemleme mekanizması tarafından toplanmaktadır. Bunun yanında toplanan verilerin işlenmesinde "birbirine bağımlılık çizgesi" [10] kullanılarak etmen kritikliği bilimsel bir modele oturtulmak istenmiştir.

Yukarıda çok etmenli sistemlerde aksaklığa dayanıklılık konusunda yapılan çalışmalardan bahsedilmiştir. Bu yapılan çalışmalar, kullandıkları etmen geliştirme çerçevesine bağlı olarak esneklik ve yeniden kullanılabilirlikten uzak çalışmalardır. [14]' deki çalışmada çok etmenli organizasyonun farklı bölümlerinde aynı anda hem adaptif hem statik aksaklığa dayanıklılık politikalarının uygulanmasını sağlayacak bir yaklaşım sunulmaktadır. Bu yaklaşımı gerçekleştirmek için aksaklığa dayanıklılık politikaları, HTN formunda yeniden kullanılabilir planlar şeklinde gerçekleştirilmiştir [16]. Böylece öteki etmenler veya yöneticiler bir etmenden aksaklığa dayanıklı bir şekilde davranmalarını istedikleri anda gönderdikleri bir istek ile etmenin aksaklığa dayanıklı olmasını sağlayabilirler veya etmen aksaklığa dayanıklı bir etmense etmenin varolan aksaklığa dayanıklılık politikasını değiştirebilirler. Çünkü aksaklığa dayanıklılığı sağlamak için sunulan servisler çok etmenli sistem geliştirme platformunun içine gömülmüştür ve bu durumda bir sistemi aksaklığa dayanıklı yapmak için sisteme aksaklığa dayanıklılık özellikleri kazandıran yazılımların sonradan sisteme eklenmesine gerek kalmamaktadır. Buna ek olarak bir etmen organizasyonunun yedek gruplarında aynı anda hem adaptif hem statik aksaklığa dayanıklılık politikalarının uygulanabilmesini sağlayacak planlar da organizasyonun kullanımına sunulmuştur.

[15]'de yine dinamik ve geniş ölçekli ortamlarda yedekleme ile ilgili parametreleri manuel olarak belirlemek kolay olmadığı için yedeklenecek olan etmeni ve yedekleme derecesini dinamik ve otomatik olarak belirleyen adaptif yedekleme mekanizması anlatılmıştır. Bu mekanizma SEAGENT [17] çok-etmenli sistem platformuna gömülmüş olup kullanıcı veya etmen kullanıcı istediği takdirde bu mekanizmayı sistem çalışmasına dahil edip adaptif aksaklığa dayanıklılık politikasını uygulayabilmektedir. Çok-etmenli sistemler için gerçekleştirilen bu mekanizma adaptif aksaklığa dayanıklılık politikalarını uygulayan diğer sistemlerdeki mekanizmalardan farklı olarak klasik kontrol teoriye dayalı geri besleme kontrolü tekniği kullanmıştır. Bu mekanizma otomatik ve dinamik olarak yedeklenecek etmeni ve yedekleme derecesini belirlemektedir ve geri besleme kontrolü tekniği daha önceden Stankovic tarafından gerçek-zamanlı zamanlayıcılarda (real-time scheduler) kullanılmıştı [18].

5. Sonuçlar

Bu çalışmada, çok etmenli sistemlere yedekleme tabanlı aksaklığa dayanıklılık politikalarını uygulayabilmek için gerekli olan servislerin nasıl gerçekleştirileceği anlatılmış ve mevcut çalışmalar incelenmiştir.

Etmen yedekleme bir veya daha fazla yedek etmenin yaratılması işlemidir. Yedek sayısı yani yedekleme derecesi önemlidir ve her şey bir etmenin görevlerini yürütürken ne kadar kritik olduğu ile ilişkilidir. Bu açıdan bakıldığında etmen kritikliğinin statik ve dinamik olması gibi iki durum karşımıza çıkmaktadır. İlk durumda, çok-etmenli organizasyon, durağan bir yapıda ve az sayıda durağan davranışlı etmenlerden oluşmuştur. Böyle bir sistemde kritik etmenler tasarımcı tarafından belirlenebilmekte ve

çalışma zamanından önce yedeklenebilmektedir. Bu durumda bu sisteme statik aksaklık politikası olarak isimlendirilen, sisteme uyguladığı yedekleme stratejisi ve yedekleme derecesini değiştirmeyen aksaklığa dayanıklılık politikası uygulanır.

İkinci durumda, çok-etmenli organizasyon, değişken bir yapıda ve çok sayıda değişken davranışlı etmenlerden oluşmuş olduğu için etmen kritikliği çalışma zamanından önce belirlenemez. Bu durumda etmen kritikliği dinamik olarak değerlendirilmelidir ve sisteme uygulanan yedekleme stratejisinin ve yedekleme derecesinin değişebileceği adaptif aksaklığa dayanıklılık politikası uygulanmalıdır.

Yapılan çalışmalarda adaptif aksaklığa dayanıklılık politikası uygulayan grupların sayısı arttığında tüm sistem performansında bir düşüş olduğu gözlenmektedir [14]. Kuşkusuz kaynakların sınırlı olduğu ortamlarda adaptif aksaklığa dayanıklılık çok etmenli sistemlere çok büyük avantajlar sunmaktadır. Ancak bu avantajların yanında sistem ve uygulamalar belirli aralıklarla gözlenecek ve elde edilen bilgilere göre organizasyona yeni yedekler eklenecek veya mevcut olan yedekler yok edileceklerdir. Bu organizasyona fazladan bir yük getirecek ve organizasyonun performansının yanıt zamanı açısından düşürecektir. Bu açıdan bakıldığında sistem performansındaki düşüş ile bu avantaj arasındaki bir noktada sistem parametrelerini belirlemek akıllıca olacaktır. Bu da bir etmen sisteminde bütün grupların adaptif aksaklığa dayanıklılık politikasının uygulanmasından kaçınılarak hem adaptif hem statik aksaklığa dayanıklılık politikalarının uygulanmasının avantajını göstermektedir. Buna göre kritikliği yüksek olan etmenlere adaptif aksaklığa dayanıklılık politikası uygulanırken, kritikliğinin değişmesi beklenilmeyen veya kritikliği yüksek olamayan etmenlere statik aksaklığa dayanıklılık politikası uygulanmalıdır. Böylece çok etmenli bir organizasyonda bazı etmenler statik aksaklığa dayanıklılık politikasını uygularken bir kısmı da adaptif aksaklığa dayanıklılık politikasını uygulayabilmektedir. Hem tek nokta aksaklığı problemini hem de adaptif yedekleme mekanizmasından kaynaklanan ek maliyeti hafifletmek açısından, gerçekleştirilecek olan adaptif yedekleme mekanizmasının dağıtık bir şekilde gerçekleştirilmesi önerilmektedir.

Kaynakça

[1] http://agtivity.com/agdef.htm, güncelleme 2006.

[2] Tanenbaum A. S. and van Steen M., Distributed Systems: Principles and Paradigms, Prentice-Hall, 2002.

[3] Paolucci M.et al., A planning component for RETSINA Agents. Intelligent Agents VI, LNAI 1757, N. R. Jennings and Y. Lesperance, eds., Springer Verlag 2000.

[4] Kumar S., Cohen P. R, and Levesque H. J., The adaptive agent architecture: Achieving fault-tolerance using persistent broker teams. In Proceedings, Fourth International Conference on Multi-Agent Systems, 2000.

[5] Stefan Pleish and Andre Schiper., Modeling Fault-Tolerant Mobile Agent Execution as a Sequence ofAgreement Problems. Proceedings of the 19th IEEE Symposium on Reliable Distributed System, 2000.

[6] Klein M. and Dallarocas C., Exception handling in agent systems. Etzioni O., Muller J. P.. and Bradshaw J. M. editors, Proceedings of the Third International Conference on Agents (Agents'99) pages 62-68, Seattle,WA,1999.

[7] H¨agg. S., 1996, A sentinel approach to fault handling in multi-agent systems, In Proceedings of the second Australian Workshop on Distributed AI, in conjunction with the Fourth Pacific Rim International Conference on Artificial Intelligence (PRICAI'96), Cairns, Australia, 1996.

[8] Fedoruk A. and Deters R., Improving fault-tolerance by replicating agents, In Proceedings of 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems, Bologna, Italy, 2002.

[9] Guessoum Z., Briot J.-P., Charpentier Z., Aknine S., Marin O. and Sens P., Dynamic and Adaptative Replication for Large-Scale Reliable Multi-Agent Systems, Proc. ICSE'02 First International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'02), ACM, Orlando FL, U.S.A, 2002.

[10] Guessoum Z., Ziane M., Faci N., Monitoring and organizational-level adaptation of multi-agent systems. AAMAS'04, ACM, pp. 514-522, New York City, 2004.

[11] Guessoum Z. and. Briot J. P., From active objects to autonomous agents, IEEE Concurrency, 7(3):68-76, 1999.

[12] Guessoum Z., Briot J. P., Sens P., and Marin O., 2001, Toward fault-tolerant multi-agent systems, In MAAMAW'2001, Annecy. France.

[13] Faci, N., Guessoum Z., Marin O., DimaX: A Fault Tolerant Multi-Agent Platform, Proc. ICSE'06 Fifth International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'06), ACM. Shangai, China, 2006.

[14] Bora, S., Dikeneli, O., Implementing A Multi Agent Organization That Changes Its Fault Tolerance Policy at Run-time. In: Proceedings of ESAW'05. Lecture Notes in Computer Science, Berlin, Germany, Department of Computer Engineering Ege University, Springer Verlag (2005).

[15] Bora, S., Dikeneli, O., Applying Feedback Control in Adaptive Replication Mechanisms in Fault-Tolerant Multi-Agent Organizations, Proc. ICSE'06 Fifth International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'06), ACM. Shangai, China, 2006.

[16] Erol K., et al., A Critical Look at Critics in HTN Planning. In IJCAI-95, 1995.

[17] Dikenelli, O., Erdur R. C., Gumus O., Ekinci E. E., Gurcan O., Kardas G., Seylan I., Tiryaki A. M., SEAGENT: A Platform for Developing Semantic Web Based Multi Agent Systems, In AAMAS'05, 2005.

[18] Stankovic, J. A., Lu, C., Tao, G. ve Son, S. H., Feedback Control Real-Time Scheduling: Framework, Modelling,

and Algorithms, Real-Time Systems Journal, vol. 23, pp.
85-126, 2002.