HATA YÖNETİMİ İÇİN ZEKİ KEŞİF VE TOPOLOJİ OLUŞTURMA YÖNTEMİ

Ertuğrul AKBAŞ[1] **Özlem SAK[2]**

[1, 2]CBR Yazılım Danışmanlık ve Bilişim Sistemleri, 41410, Gebze-Kocaeli

[1]e-posta: ertugrul@cbr.com.tr [2] e-posta: ozlem@cbr.com.tr

Anahtar sözcükler: Ağ *Topolojisi, Keşif, Hata Yönetimi, Olay İlişkisi, Dağıtık Ağ Yönetimi*

ABSTRACT

This paper will outline a complete system from automatic network model (network topology) generation to escalating events. The core parts of dynamic fault management system will be decomposed and intelligent methodologies both for detecting topology and root cause analysis will be analyzed and practiced. This work shows that proposed platform-independent, distributed and reusable fault management system architecture can be an integral part of the next generation of network management systems. Proposed architecture is applicable not only to network management system, but also to intrusion detection systems, business systems and health care information correlation systems. We present this architecture and the use java based network management platform. Finally, we demonstrate how these technological solutions have been implemented.

1. SUNUM

Ağ işlemleri karmaşıktır ve ağ operatörleri, ağ üzerinde hatalar oluştuğu zaman çok sayıda olay mesajı ile boğulmuş durumda olurlar. Bu olay mesajları sayılarının fazlalığı nedeniyle operatör tarafından işleme alınmaz. Bilgi içeriğini geliştirirken alarm seviyelerini indirgeme başarı ile yönetilmiş bir ağın belirleyici anahtarıdır.

Eğer filtreleme ve ilişki metotları görüntüleme işleminden önce uygulanırsa çok sayıda alarm ciddi biçimde indirgenebilir [1] . Hata yönetiminin en temel parçası olay ilişkileridir ve olay ilişki teknikleri kullanılmadan, olay mesajları yığınının verimli bir şekilde işlenmesi olanaksızdır. Olay ilişkilendirmenin temel işlevleri görüntüleme, filtreleme ve olay alarmlarını maskelemedir. İdeal bir hata yönetim sistemi, alarmları ya da olayları bir sebep-sonuç ilişkilendirme kabiliyeti, karmaşık filtreleme fonksiyonlarını kullanarak olayları filtreleme ve olaylar için ağ nesnelerini görüntüleme kabiliyetine sahip olmalıdır.

Günümüz ağ hata yönetimi gereklilikleri, tepkisel olay yönetimi ve zamanlama önem taşıyan unsurlardır. Bundan dolayı, dikkate alınan ve alınmayan olaylar hakkında tespit etme ve karar verme en verimli şekilde yapılmış olmalıdır. Bu araştırmada, sistem bağımsız, yeniden kullanılabilir ve dağıtık hata yönetim sistemi önerilmektedir ve bu sistem ağ topoloji bilgisini ve olaylar arası nedensel ilişkileri tutmaktadır.

Olay ilişkilendirme, hata yönetim sisteminin merkezidir. Olay ilişkilendirme, ağ ve servis yönetiminin dışında performans, test etme, servis kalitesi ve güvenlik yönetimi konularında genişletilmiş olmalıdır bunun sebebi gelecek nesil ağ yönetim sistemlerinin tamamlayıcı bir parçası olmaktır.

Şu anki, çoğu olay ilişkilendirme kabiliyetleri, yönetim sistemine gömülü bir çözüm olarak veya ağ yönetim yazılımlarının diğer bileşenleri arasında yetersiz entegrasyon ve kaynak paylaşımı ile birlikte bağımsız 3. parti sistemler olarak bulunurlar.

Başlıca farklılıklar:
İnsanoğlu, eğer model karmaşık bir yapıya sahip ise bazı detayları gözden kaçırabilir.
Sistem yöneticileri iyi yetiştirilmemiş olabilir
Her ilişki mekanizması kendi modellerini tanımlamak için kendi metodolojisine sahiptir.

Bu problemler yüzünden, otomatik ağ oluşturmak ve olay ilişki modeli önemlidir.

Önerilen dağıtık ve dinamik hata yönetimi mimarisi Java Dağıtık Hata Yönetimi Mekanizması

(JADFAME) tabanlıdır ve bu problemlere çözüm bulacaktır[4].

Bununla birlikte uygulama ve prototip çalışması bir ağ yönetimi çözümü üzerinde gerçekleştirilmiştir, önerilen hata yönetimi mimarisi saldırı algılama, stok market, ev güvenlik, sağlık hizmetleri gibi hata yönetiminden fayda sağlayan alanlar üzerinde kullanılabilir.

Bu makale aşağıda gösterildiği gibi düzenlenmiştir:
2. kısım sistem mimarisini tanımlar.
3. kısım uygulama detaylarını sunar.
4. kısım sonucumuzu içerir.

2. MİMARİ

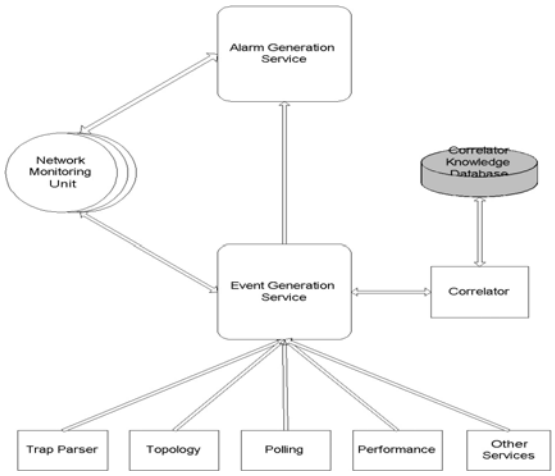
JADFAME- Java Dağıtık Hata Yönetimi Mekanizması – dağıtık bir hata yönetimi servisidir. JADFAME ' in ilk uygulaması dinamik olarak isteğe uyarlanabilen olay ilişkilendirme ve dağıtık sistem mimarisi üzerine odaklanır. Üç ana bileşen:

Olay işleme birimi: Bu bileşen, dağıtık bir ortamda olay kayıt birimi olarak kullanılmıştır. İlişkilendirilmeye ihtiyacı olan olaylar, olay – işleme birimini kullanarak olay veritabanına eklenmiş olmalıdır. Bu birim performans, güvenlik servisleri topoloji olayları ve SNMP trap' ları gibi çeşitli kaynaklardan gelen olayları kaydeder.

İlişkilendirme Birimi: Bu bileşen, dağıtık bir ortamda olay ilişkilendirme mekanizması olarak kullanılmıştır. Bu mekanizma olay ilişkilendirme modelini kullanarak bileşik olayların ilişkilendirilmesi, olay nedeni ve olay kaynaklarını kullanarak basit anlamda ilişkilendirme yapar.

Uyarı İşleme Birimi: Bu bileşen dağıtık bir ortamda, uyarı oluşturma birimi olarak kullanılmıştır. Bu birim temel olayları görüntülemek ve aynı zamanda hareket oluşturmak için kullanılmıştır, örneğin; elektronik posta gönderme, olayları bırakma, kısa mesaj gönderme ve otomatik hesap açma.

Bu mimari, esnek bir hata yönetim uygulama altyapısıdır (computing framework) ve geliştiriciler ve araştırmacılar kendi hata yönetimi uygulamalarını geliştirebilir ve çalıştırabilirler. Çünkü programlama desteği, açık ve esnek bir altyapıya sahip mekanizma ile desteklenmiştir.



Şekil 2. Dağıtık Bileşen Mimarisi

2.1 Dağıtık Mimari

Dağıtık mimari, genelde ağ yönetim sistemlerini ve özelde olay ilişkilendirme sistemlerini yapılandırmada genel kabul görmüş ve kullanılan bir tekniktir[2]. İyi tanımlanmış fonksiyonlarla birlikte standart servislerin kullanımı ve standart iletişim protokolleri; açık, ölçeklendirilebilir ve isteğe uyarlanabilir sistemlerin yapılandırılmasına izin verir. Çeşitli teknolojiler CORBA, DCOM ve RMI dağıtık ağ yönetim sistemlerinin altyapısını yapılandırmak için kullanılabilir[5]. Bu makalenin takip eden kısımlarında, RMI altyapısını kullanarak mimariyi kendiliğinden web tabanlı hata yöneticisine dönüştüren dağıtık olay ilişkilendirme sistemlerinin dizaynını açıklayacağız.

2.2 İlişkilendirici Mekanizma

JADFAME bileşen ilişkilendirici, yüksek dinamizmi olan ve kolay modellenebilir bir olay ilişkilendirme modeline sahiptir ve model ilişkilendirici bilgi veritabanına eklenir.

Diğer ağ yönetimi platformlarına uyarlanabilen ilişkilendirici bilgi veritabanı içinde birbiriyle alakalı topoloji bilgilerini bir araya getiren bir topoloji ilişki modelleyici modül (şekil 7) vardır. Olay ilişkilendirme sisteminin en önemli parçası başlangıç ilişkilendirmeyi oluşturmaktır. Eğer ilişkiler doğru değilse, sistem tarafından üretilen herhangi bir bilgi de yanlış olacaktır. Topoloji keşfi tipik olarak olay ilişkilendirmesi için çok kullanışlı bir araç olarak kabul edilmiştir[6].

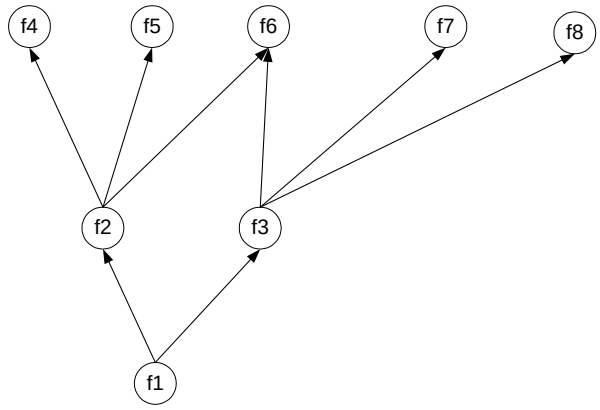
Olay ilişkilendirme modeli, topoloji nesnelerini ve topoloji olaylarını ilişkilendirilip birleştirildiği birimdir.

Topoloji birliği modelleyici bileşeni, yönetilen nesneleri ve onların ilişkilerini otomatik olarak belirler. Bu model aynı zamanda kullanıcı tarafından ayarlanabilir. Topoloji ilişkilendirme modelleyicisi, keşif modüllerinden yönetilen

nesnelerin örneklerini oluşturur ve görüntüleme modülleri, servisleri tarama ya da olay veritabanından olay örneklerini oluşturur bundan sonra olay ilişkilendirme modeli örnekleri oluşturulmuş olur.

Olay ilişkilendirme genel model şablonu, kurallar kullanılarak oluşturulur:
"Eğer olay varsa hareket gerçekleşir",
"Eğer olay1 ve olay2 ve olay3 varsa hareket gerçekleşir yoksa olaylar durdurulur,
"Eğer olay eşik değeri geçilirse hareket gerçekleşir",
"Eğer olay eşiğinin t2 zamanlama sınırlaması varsa olay gerçekleşir yoksa olaylar durdurulur",
"Olayı durdur",
"Bütün olayları durdur",
"Kritere göre filtrele",

Bu kurallar içinde; elektronik posta gönderme, midi çalma, program çalıştırma ya da olay(lar) oluşturma hareket olarak değerlendirilebilir. AND, OR, NOT gibi mantık işleçleri kuralın bir parçası gibi kullanılmış olabilir.



Şekil 3. Uyarı İlişkilendirme Kullanan Ağ Tanılayıcı

Önerilen sistemin olay ilişkilendirme modelleyicisi aşağıdaki gibi çalışır:
Her olay kaynağı ve bu olaylar arasındaki ilişkiler modellenmiştir.
Hata f4, hata f2 nedeniyle ya da hata f6 hem hata f2 hem hata f3 nedeniyle gerçekleşiyorken hata f3, hata f1 tarafından gerçekleşmiştir. Basit mantık kuralları içerisinde uyarıları ilişkilendirerek, biri aşağıdaki hata kurallarını yapılandırılabilir:

Kural 1: if f4 and f5 and f6 then f2
Kural 2: if f4 or f5 or f6 and f2 then f1
Kural 3: if f2 and f3 then f1

Bu ilişkilendirme modelinin başka bir avantajı; yönetilen nesne ve olay model ilişkilendirici modelden ayrıştırılabilir ve java' nın yardımıyla, bütün sistemi kapatmaksızın ve yeniden derlemeksizin her iki model değiştirilebilir.

3.ZEKİ TOPOLOJİ OLUŞTURULMASI

3.Katman topolojisi, bir ağ içerisinde varolan düğümler ve yönlendiriciler ile ilişkilendirilmiştir ve bu yönlendiriciler arası yollar birbirlerine bağlanmışlardır. Ağ topolojisinin genel bir görünüşü, farklı alanlarda ağ yönetimine yardımcı olur. Ağ topolojisini bilmek, ağ yöneticilerine bir ağın daha iyi şartlarda çalışması için gerekli bilgiyi verir. Topoloji bilgisi hali hazırda var ise muhtemel tıkanıklıkları bulmak daha kolay olur. Tıkanıklık probleminin halletmek için ağ içerisinde belli başlı bir yerde ekstra bir yönlendiriciye ihtiyaç olabilir. Pek çok ticari ürünlerde 3.katman keşfi gerçekleştirilmiştir fakat 2.katman zor ve yönetilen cihaza bağımlı olduğu için genel bir çözüm verilmemiştir. Bir ip ağının güncel fiziksel topoloji bilgisi; reaktif ve proaktif kaynak yönetimi, olay ilişkilendirilmesi, ve sebep-sonuç analizi dahil pek çok ağ yönetimi görevleri için önemlidir. 2. katman keşfi, 2. katman ağ cihazlarını bulur ve otomatik olarak topoloji haritalarını oluşturur. Predriyne [9] ya da openview [6] gibi bazı ticari programlar mevcuttur fakat bunlar ticari olarak korunmuş protokoller üzerinde kullanılır ve heterojen ağlar için desteklenmez.

3.1 III. Katman Keşfi

İlk olarak, yönlendiricilerden SNMP [10] ile mevcut bilgiyi kullanır. İkinci olarak, ağ üzerinde MIB tablolarını toplar ve sonrasında trafik oluşturmaz. Üçüncü olarak, her bir düğüm üzerinde çalışan diğer SNMP ajan programlarından başka hiç bir yazılıma güvenmez.

3.1.1 Algoritma

Ağ topolojisinin keşfi için kullanılan algoritma, bir yönlendiricinin yönlendirici tablosunda ve ARP tablosunda depoladığı bilgiler kullanılarak oluşturulur. MIB-II IP grubu içerisinde "ipRouteTable" olarak ifade edilen yönlendirici tablosu mevcut internet yönlendirme bilgilerini içerir ve ağ topolojisini çıkarmak için kullanılabilir. Bu tablo içindeki girişler hedef ip tarafından indekslendirilmiştir. Her giriş, hedefe giden bir yönlenmeyi gösterir. Bu yönlenme içinde, her bir giriş için ipRouteNextHop alanı sonraki yönlendiricinin ip adresini belirler. Böylece yönlendiricinin direkt komşuları, ipRouteTable' dan ipRouteNextHop alanları elde edinilerek öğrenilebilir.

Bir yönlendirici ile başlama, keşif algoritmasının çalıştığı makinenin varsayılan ağ geçidi bilgisine bakılarak veya bir giriş argümanı belirlenerek öğrenilebilir, yönlendiricinin her bir komşusu için

her bir tekrarda tekrarlamalı algoritma çağırılarak bu işlem yapılabilir, böylece "yönlendirici bağlantılı" olarak keşif işlemi tamamlanır. Döngünün bitiş şartı ya domain kriteri ya da ilk yönlendiriciden olan ve önceden belirlenen maksimum uzaklıktır.

// Bu yönlendiricinin IP yön tablosu ele alınır ve onun direkt komşuları bu tabloya eklenir.

```
void getIPRouteTable(Router router) {

    check ( hopnumber );   // Check # of hops.

    ipRouteTable ' ı bu yönlendiriciden al;
    Her geçerli yönlendirme üzerindeki tüm ilişkileri süz;
    Bütün bu ilişkileri (komşuları) bu yönlendiricinin ilişki listesine ekle
    hopnumber ++;
    Router theRouter = router.getNeighbor();

    while(theRouter != null ){
        getIPRouteTable(theRouter);
        // Recursion
        theRouter=router.getNextNeighbor();
    }
    hopnumber --;
 }
```

Yukarıda bulunan algoritma yalnızca yönlendiriciler arası topolojiyi yapılandırır. Diğer hostların bağlanılabilirliklerini keşfetmek için, bir yönlendirici ARP tablosu kullanılmıştır. Biz her tekrarlamada ARP tablosunu basitçe okuyabiliriz ve hostları onun host listesine ekleyebiliriz.

Bu aşama, sadece yönlendiriciler ve anahtarlar (2.katman ve 3. katman) gibi ağ cihazlarını bundan başka son hostlar gibi 3. katman cihazlarını keşfetmez. Şekil 6. ve şekil 7. de (pek çoğu marka ve versiyonu ile) gösterilen araçların tam listesini almak amacıyla "end host" analizlerini yapmak ve yukarıda gösterilen algoritmayı kullanarak ağ cihazları arasındaki bağlantılar yapıldıktan ve başlangıç keşif işleminden sonra SysOID kullanarak keşfi tamamlayacağız.

3.2 II. Katman Keşfi

II. katman topoloji keşfi, MIB II ve BRIDGE MIB gibi yalnızca SNMP tabloları kullanarak gerçekleştirilmiştir. Topoloji MIB'i gibi özel mib'ler heterojenliği bozmamak amacıyla kullanılmaz. Biz BRIDGE MIB 'i kullandık ve bu algoritma (spanning tree analysis) tabanlıdır.
Anahtarlamalı bir domain olan N 'in uçları keşfediliyor.
S = Anahtarlamalı domain' in switch ve yönlendiricilerin mac adresi listesi olarak tanımlanmış.
Sij = Si switch 'inin J. interface' i Sij olarak tanımlansın.
Aij = O switch portunun tüm mac adres listesi de Aij olarak tanımlansın.
D = SNMP si olan bir araçtır.

Adım 1.

$A_{ij} \bigcup A_{kl} = U$ ve $A_{ij} \bigcap A_{kl} = \phi$ ise S_{ij} ve S_{kl} direkt olarak birbirine bağlıdır. İspat burada dikkate alınmaz. İlgilenen okuyucular [1-11-12] nin gösterdiği kaynaklara bakabilirler.

Adım 2.

$S_i \in A_{kl}$ fakat $S_k \notin A_{ij}$ değil ise S_i ve S_k birbirine bağlı değildir.

Adım 3.

$A_{ij} \bigcap A_{kl} = \{D_m\}$ ise "spanning tree analysis" yapar.

Eğer (S_i ' nin "root port" kolonundaki değeri ve S_k nın root değeri , S_i nin root değerine eşitse) S_i and S_k, SNMP si olmayan S_m aracı ile birbirlerine bağlıdırlar.

Adım 4.

$A_{ij} \bigcap A_{kl} = \{D_m\}$ ise "spanning tree analysis" yapar.

Eğer (S_i ' nin "root port" kolonundaki değeri ve S_k nın root değeri , S_i nin root değerine eşitse) "graphical spanning tree analysis" yap.

"Graphical spanning tree analysis : "
Eğer S_i switch 'inin tablosunun "enable forwarding" alanında D_m aracı için bir giriş varsa ve D_m aracı root' tan uzakta ise D_m konum olarak S_i switch ' inin altındadır.

Eğer S_i switch 'inin tablosunun "enable forwarding" alanında D_m aracı için bir giriş varsa ve D_m aracı root' tan önünde ise D_m konum olarak S_i switch ' inin üstündedir (D_m konum olarak S_i switch 'i ile root arasındadır).

4. PROTOTİP OLUŞTURMA

Önerilen sistemin prototipi, yönetim platformu olarak CBR Big:Eye [13], veritabanı olarak mysql [14] ve programlama dili olarak java kullanılarak oluşturulmuştur. Sistem olay yapısı şekil 4 de tarif edilmiştir. Olay işleme birimi aşağıdaki formatta gösterildiği gibi olayları kabul eder.

Event Catagory	Topology Info	Reason	Time

Şekil 4. Olay Yapısı

Başlıca Alanlar:

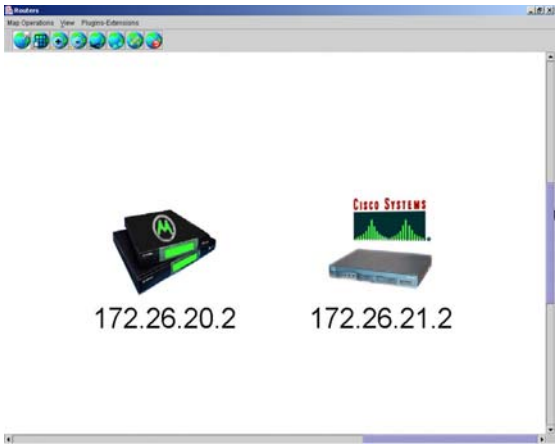
Olay Kategorisi: Bu alan olayın hangi kaynağa ait olduğunu belirtir, örneğin bir durum olayı, bir güvenlik olayı, bir topoloji olayı ve bir performans olayı.

Topoloji Bilgi(si): Bu alan, olay ve fiziksel ağ arasındaki ilişkileri belirleyen ağ, eleman adı, nesne adı gibi pek çok alt-alana sahiptir.

Sebep: Bu alan olayın sebebini belirtir. Düğüm hatası , (cihazlar arası bağlantıların) bağ kopması, cihazlar arası bağlantıların gerçekleştirilmesi ve eşik değerleri olay nedenlerine bazı örneklerdir.

Zaman: Bu alan olayın oluştuğu zamanı belirtir.

Şekil 5. Topoloji (Ağ Yapısı)



Şekil 6. "End Host" Analiz Sonuçları

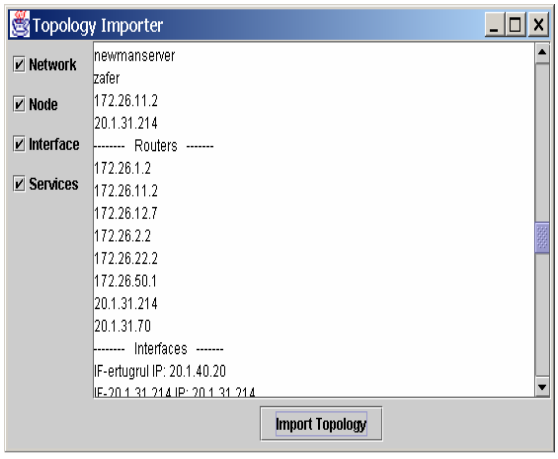
4.1. Gelişmiş Olay Tarama Özellikleri

JADFAME hata yönetim sistemi, olay tarayıcısı olarak adlandırılan çok büyük bir kapasiteye sahip olay canlandırma aracıdır. Bu tarayıcı, RMI altyapısı ile alarm işleme bileşenlerini ve olay sorgulamasını kullanır

Olay tarayıcısı zaman, alarm seviyesi ve olay kaynağı tabanlı filtreler tanımlama kabiliyetine sahiptir.

4.2 Topoloji İlişki Modelleyicisi

Topoloji ilişki modelleyicisi, topoloji alakalı bilgileri ilişkisel bilgi veritabanına eklemek amacıyla prototip içinde kullanılan bir araçtır. Bu çalışmada biz bir ağ yönetim altyapısı olarak Big:Eye topoloji veritabanını ve olay veritabanını kullandık. Topoloji nesneleri ve olay ilişkilerinin, ilişkisel bilgi veritabanına eklendikten sonraki durumu Şekil 7. de gösterilmiştir. Burada ilişkileri gösteren bir ağaç modeli vardır ve bütün yaprak objeler kendi sahiplerini bilirler ve bütün kök nesneler kendi child' larını bilirler. Bu model daha sonra kural tabanlı ilişki modeline dönüştürülmüştür.



Şekil 7. Topoloji İlişki Modelleyicisi

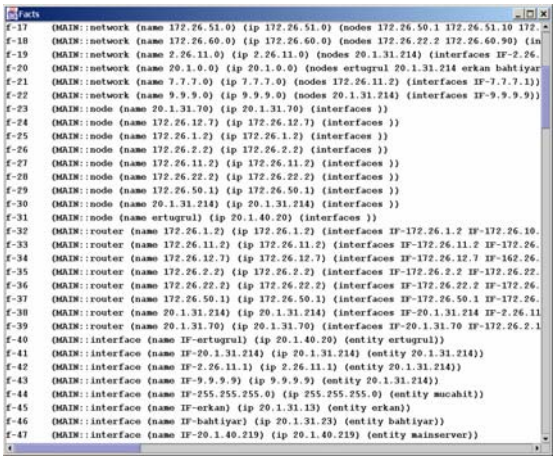
Şekil 5. te gösterilen topoloji bir test yatağı olarak kullanılmıştır. Bu ağ Big:Eye kullanılarak keşfedilmiştir ve topoloji dahil edici modül , bilgi veritabanı modeli için kullanılmıştır.[Topoloji ilişki modelleyicisi otomatik ilişki(sel) bilgi tabanı modelleme olarak kullanılmıştır]. Otomatik topoloji keşfi, olay ilişkilendirici modelleme için çok kullanışlı bir araçtır.

Topoloji ilişki modelleyicisi, önceden tanımlanmış şablon bilgi modelinin yardımıyla, ilişki mekanizması bilgi tabanının otomatik modellenmesi için kullanılır. Bu bileşen otomatik olarak bir model oluşturur, daha sonra (eğer istenirse) modelin değiştirileceği düşünülerek topoloji ilişki modelleyicisi tarafından oluşturulan model tam olarak bileşene bağlanmaz. Topoloji ilişki modelleyicisi yalnızca ağ yönetimi alanına tam olarak ilişkilendirilmiş olan bir bileşendir. Eğer JADFAME, saldırı algılama sistemleri, ticari sistemler ve sağlık hizmetleri gibi sistemlerde kullanılacak olursa, topoloji ilişki modelleyicisi gibi başka bir araç ortaya çıkacak, bu araç bahsedilen domain' lerden JADFAME ' e verileri aktaracaktır.

4.3. Gelişmiş İlişki Özellikleri

Bu makalede hata yönetiminin mimari fonksiyonları üzerinde yoğunlaşıldı ve aynı zamanda geliştirilen prototip hata yöneticisi için tam özellikli bir olay ilişkilendirici geliştirildi. İlişkilendirici mekanizma benzer kaynaktan gelen olaylar için özel bir çeşit mantık değerlendiricisine sahiptir ve bu değerlendirmede motor olarak jess[15] kullanılmıştır.

Topoloji yönetici yardımıyla, yönetilen nesneler arası ilişkiler ve ağ topolojisi hakkında doğru bilgiler bir araya getirilir. Her yönetilen nesne, yönetilen nesneler arası ilişkilere göre veritabanında gösterilmiştir. Şekil 8, otomatik olarak modellenmiş ilişki bilgi veritabanı içinde bulunan doğru bilgilerin bir ekran görüntüsünü sunar.



Şekil 8.Olgu (Çarpınım) Bilgi Tabanı

5. SONUÇ

Bu makalede dağıtık bileşen tabanlı mimari kullanılarak, hata yönetimine yeni bir yaklaşım sunulmuştur. Bu fikir, saldırı algılama gibi olay ilişkilendirme desteğine ihtiyaç duyan diğer sistemler gibi gelecek jenerasyon yönetim sistemlerinin gerekliliklerini bir araya getirir. Bununla birlikte bilgi tabanı modelleme için metodoloji oluşturma ve otomatik kural algılama sunulmuştur. Olay ilişkilendirme modeli topoloji bağımlılıkları kullanılarak oluşturulmuştur ve gerçek bir ağ üzerinde uygulanmıştır.

Önerilen mimari diğer sistemlere adapte edilebilir özelliğe sahiptir. Bu şu anlama gelir, dağıtık hata yönetim sistemini kullanmak için bir ön gereklilik yoktur. Bununla birlikte bu mimari web tabanlı yönetim sistemlerine uyarlanabilir ve dağıtık API' lerin yardımıyla üçüncü parti araçların ayrılmaz bir parçası olabilir.

Önerilen ilişkilendirici mekanizma bulanık mantık ve sinir ağları gibi yapay zeka teknikleri ile entegre olabilme yeteneğine sahiptir.
Önerilen mimari aynı zamanda esnek bir hata yönetimi (computing framework) altyapısıdır ve bu esnek hata yönetimi altyapısının (computing framework) yardımı ile araştırmacılar bu teknikleri kendi sistemlerine uyarlayabilirler.

KAYNAKLAR

[1] Jakobson, G., Weissman, M., "Alarm Correlation," IEEE Network, Vol. 7, No. 6, pp. 52-59, Nov. 1993.

[2] Hasan, M., Sugla, B., Viswanathan, R., "A Conceptual Framework for Network Management Event Correlation and Filtering Systems," Sixth IFIP/IEEE International Symposium, pp. 233-246, 1999.

[3] Jakobson, G., Weissman, M., Brenner, L., Lafond, C., Matheus, C., "GRACE: Building Next Generation Event Correlation Services," Proceedings of IEEE/IFIP Network Operations and Management Symposium, pp. 701-714, April 2000.

[4] E. Akbaş, "System Independent And Distributed Fault Management System", *The Eighth IEEE Symposium on Computers and Communications,* 30 June-3 July 2003, Antalya, Turkey

[5] Downing, T. B., Java RMI: Remote Method Invocation, IDG Books Worldwide, 1998.
[6] Yui Breitbart, Mios Garafalakis, Cliff Martin, Rajeev Rastogi, S. Seshadri and Avi Silberschats, "Topology discovery in Heteregoneus IP Networks", Proceedings of IEEE INFOCOM,2000

[7] www.hpopenview.com
[8] www.cisco.com
[9] www.predryne.com

[10] J. Case, M. Fedor, M. Schoffstall, and J. Davin. “A Simple Network Management Protocol (SNMP)”. Internet RFC-1157, May 1990.

[11] Y. Bejerano, Y. Breitbart, M. Garofalakis, and R. Rastogi. “Physical Topology Discovery for Large Multi-Subnet Networks”. In *Proc. of IEEE INFOCOM*, 2003.

[12] Y. Breitbart, M. Garofalakis, C. Martin, R. Rastogi, S. Seshadri, and A. Silberschatz. “Topology Discovery in Heterogeneous IP Networks”. In *Proc. of IEEE INFOCOM*, 2000.

[13] www.cbr.com.tr

[14] www.mysql.org

[15] www.sandia.gov/~jess