

T.C.
GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ
MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

YAZILIM DAĞITIM SİSTEMLERİNDE
GERÇEK ZAMANLI DURUM İZLEME

72 1301

Aziz Barış DELİPİNAR
YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

TEZ DANIŞMANI
YRD. DOÇ. DR MEHMET GÖKTÜRK

T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ

GEBZE

2002

72 1301

..Aziz...Bacı...Değer... 'in tez çalışması, G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..11.03.2002... tarih ve 2002...1.10... sayılı kararıyla oluşturulan jüri tarafından ..B.İ.Şayar...Mühendisliği... Anabilim Dalında YÜKSEK LİSANS tezi olarak kabul edilmiştir.

JÜRI

ÜYE
(Tez Danışmanı)

: Yrd. Doç. Dr. Mehmet Göktürk

ÜYE

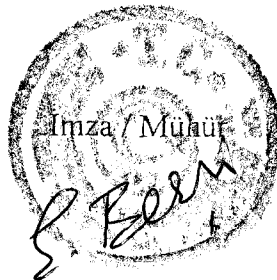
: Yrd. Doç. Dr. İlyas Kandenir

ÜYE

: Yrd. Doç. Dr. Erdi Şen Sevilgen

ONAY

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun ..06.08.2002... tarih ve 2002/1.5... sayılı kararı.



ÖZET

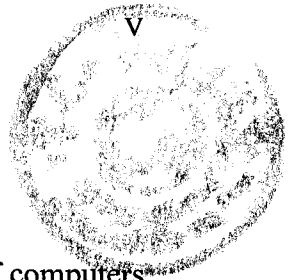
Bu tez çalışmasında dağıtık ortamlardaki yüzlerce bilgisayarı bulunan kamu yada özel kuruluşların periyodik olarak yaptığı yazılım dağıtımının canlı olarak izlenmesi amaçlanmıştır. Yapılan çalışma ile merkezdeki bir izleme bilgisayarı ile yine merkezdeki yada uzaktaki çok sayıda bilgisayar arasında iletişim kurularak uzaktaki bilgisayarların durumunu merkeze bildirmesi sağlanmıştır. Merkezdeki izleme bilgisayara gelen bilgiler kullanıcının kolayca anlayabileceği ve duruma hakim olabileceği bir grafik arayüzü ile sunulmuştur. Çalışma üç ana bölümden oluşmaktadır.

İlk bölümde, yazılım dağıtım araçları ve ortamdaki bilgisayarlara değinilmiştir. Her bilgisayarın görevi ve yazılım dağıtımının hangi aşamalarla gerçekleştiği kısaca açıklanmıştır. Ayrıca, sık yaşanan ve çözülmesi gereken problemlere temas edilerek canlı izleme sistemine olan ihtiyaç tespit edilmiştir. Dünyada yaygın kullanım alanı olan mevcut işlem izleme ürünlerine kısaca değinilmiştir. Bu mevcut yazılımların bazı özellikleri incelenerek ihtiyacı karşılayıp karşılamadığı ele alınmıştır.

İkinci bölümde, incelenen kaynaklardan da faydalanılarak ihtiyacı karşılayacak olan sistemin tasarımı yapılmış ve sistemin bileşenleri tespit edilmiştir. Bu bileşenlerin her birinin üstleneceği görevler detaylı açıklanarak sistemin işleyiş mantığı ve şekil olarak diyagramı çıkartılmıştır.

Üçüncü ve son bölümde ise yapılan tasarıma göre Microsoft Windows ortamlarında çalışan “Borland Delphi 5.0 Enterprise Edition” programlama dili kullanılarak kodlanan ve kolay bir arayüz oluşturulan programlar anlatılmıştır. Gönderici, Server (sunucu) ve İzleyici olarak nitelendirilen üç adet programın her biri detaylı olarak incelenmiştir. Bu programların çalışma şekilleri ve yer yer kodları açıklanmıştır. Sistemin üretim ortamındaki sonuçlarına ve yapılabilecek ek çalışmalara kısaca değinilmiştir.

SUMMARY



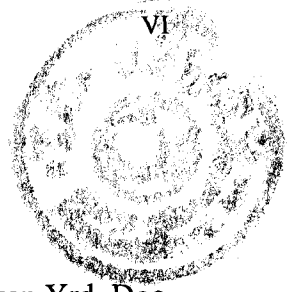
In this thesis, live observation of software distribution to hundreds of computers of government organizations or private companies, the branches of which are located both at center organizational place or away from center organizations, has been analyzed.

In section 1, the software distribution systems' general properties and function of the computers used in these systems are examined. The steps of software distribution are handled briefly. The usual problems of software distribution processes and quick solutions are examined. The needs for a live observation system have been determined and the current monitoring products that are used world wide have been examined and checked whether they fulfil our needs or not.

In section 2, with the help of the examined references, the design of the system which will fulfill our needs have been realised and the components that will form the system have been determined. The functions of each of these components have been explained in detail and the work logic and flow diagram of the system have been extracted.

In section 3, the program codes and user friendly graphical user interface that have been designed and coded with "Borland Delphi 5.0 Enterprise Edition", a visual programming language running under Microsoft Windows, are examined. The three programs that can be named as Sender, Server and Observer have been examined in detail. The work logic and important codes of these programs have been explained. Also the production results of the system and future work have been mentioned.

TEŞEKKÜR

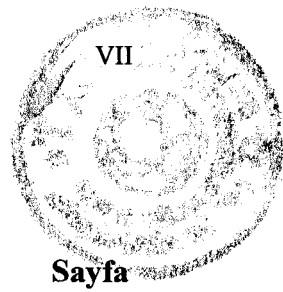


Bu sistemin tasarımımda bana destek olan ve yardımlarını esirgemeyen Yrd. Doç.
Dr. Mehmet Göktürk hocama,

Bu sistemi tasarlama aşamasında yaptığım çalışmalarda bana yardımcı olan ve
çalışmalarıma destek veren, yazılan kodları test imkanı sağlayan iş arkadaşlarıma,

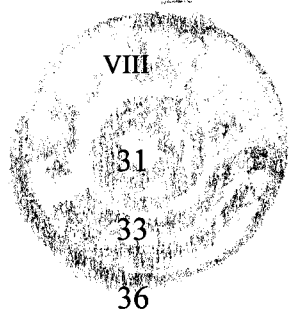
“perl” diliyle gönderici pogramın HP Unix 9000 ve IBM AIX versiyonlarını
yazan Unix grubu arkadaşlarıma,

Ve tabii, çalışmalarım sırasında sürekli bana destek olan sevgili eşim Şermin’e
en içten dileklerimle teşekkür ederim.



İÇİNDEKİLER

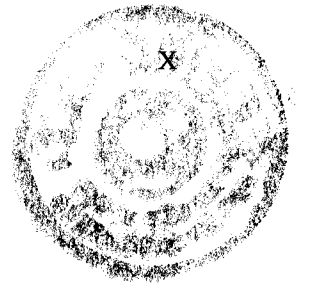
ÖZET	IV
SUMMARY	V
TEŞEKKÜR	VI
İÇİNDEKİLER	VII
SİMGELER VE KISALTMALAR DİZİNİ	IX
ŞEKİLLER DİZİNİ	X
1. GİRİŞ	1
1.1 Yazılım Dağıtım Sistemindeki Bilgisayarlar	2
1.2 Yazılım Dağıtım Aşamaları	3
1.3 Yazılım Dağıtım Sürecinde Yaşanabilen Problemler	5
1.4 Yazılım Dağıtım Sürecinin Canlı Olarak İzlenmesi	7
2. İZLEME SİSTEMİ TASARIMI	11
2.1 TCP/IP Protokolünde TCP ve UDP Metodları	11
2.2 Sistem Tasarımı	12
2.3 Server Programının Ek Görevleri	16
2.3.1 Server'a Gelen Mesajların Kayıt Edilmesi	16
2.3.2 İzleyici Bilgisayara Kaydedilen Kesajların Gönderilmesi	17
3. GELİŞTİRİLEN PROGRAM	18
3.1 Gönderici Program	18
3.2 Server Program	19
3.3 İzleyici Program	22
3.3.1 İzleyici Programın Temel Çalışma Prensibi	22
3.3.2 Detay Bilgilerin Görüntülenmesi	29
3.3.3 Kullanıcı Durumlarının Dosyaya Kaydedilmesi	31



3.3.4 Bilgisayarların Açık / Kapalı Kontrolü	31
3.3.5 “ayarlar.ini” Dosyasının Yapısı ve Değişiklikler	33
3.3.6 Daha Fazla Aşamanın Takip Edilebilmesi	36
4. ÜRETİM ORTAMINDAKİ SONUÇLAR	39
5. SONUÇ	42
6. YAPILABİLECEK EK ÇALIŞMALAR	43
KAYNAKLAR	45
ÖZGEÇMİŞ	46
EK-1 GÖNDERİCİ BİLGİSAYARDAKİ KODLAR	47
EK-2 SERVER BİLGİSAYARDAKİ KODLAR	48
EK-3 İZLEME BİLGİSAYARINDAKİ KODLAR	55
EK-4 ÖRNEK BİR ”AYARLAR.INI” DOSYASI	62

SİMGELER VE KISALTMALAR DİZİNİ

- EDM** Enterprise Desktop Manager (Novadigm firması tarafından geliştirilen yazılım dağıtım sistem programlarının genel adı)
- LAN** Local Area Network (Yerel Alan Ağı)
- NDM** Network Distribution Manager (IBM firması tarafından geliştirilen yazılım dağıtım sistem programlarının genel adı)
- OS/2** Operating System /2 (IBM firmasının PC bazlı işletim sistemi)
- SNMP** Simple Network Management Protocol (Basit Ağ Yönetim Protokolü)
- TCP** Transmission Control Protocol
- UDP** User Datagram Protocol
- WAN** Wide Area Network (Geniş Alan Ağı)



ŞEKİLLER DİZİNİ

Şekil	Sayfa
1.1 Yazılım Dağıtım Ortamındaki Bilgisayarlar ve Bağlantıları	3
2.1 Mesaj alışverişi yapan bilgisayarlar	14
3.1 Server Programındaki Mesajlar	21
3.2 Server'a Kayıt Olma Ekranı	23
3.3 Kullanıcı Kodlarının Mesaja Göre Yerleştirilmesi	25
3.4 Aynı Konumdaki Bilgisayarların Alt Alta Listelenmesi	26
3.5 Daha Fazla Bilgisayarın Durumunun Gösterilebilmesi	27
3.6 Detayların Görüntülenmesi	30
3.7 Bilgisayarların Açık / Kapalı Kontrolünün Yapılması	32
3.8 Belli Bir Bilgisayarın Mesajlarının Görüntülenmesi	37
4.1 Üretim Ortamından Bir Görüntü	40



1. GİRİŞ

Günümüzde kamu ve özel kuruluşları geniş bir coğrafi alana yayılmış olan bütün şube ve temsilciliklerinde merkez ile uyumluluk açısından aynı yazılımları kullanmak durumundadır. Bu dağıtık sistemlerin ilk kuruluşundan sonra , yazılımlara yeni özellikler eklenmesi, zaman içinde karşılaşılan bir takım problemlerin giderilmesi, değişen mevzuata göre programda bir takım değişiklikler yapılması vs. gibi nedenlerden ötürü yazılımların yeni sürümünün gönderilmesi zorunluluğu ortaya çıkmaktadır. Bu da değişik coğrafi konumlarda olan yüzlerce ve bazen binlerce bilgisayara aynı programların gönderilmesi işiyle beraber karşımıza bu işin takip edilmesi, raporlanması gibi ek işler çıkarır. Bir kuruluştaki bazı şubelere bazı programların eksik gitmesi yada hiç gidememesi, şubelerin bazılarında programın yeni sürümünün bazılarında eski sürümünün bulunmasına yol açar, hatta bazı programların çalışmamasına sebebiyet verebilir.

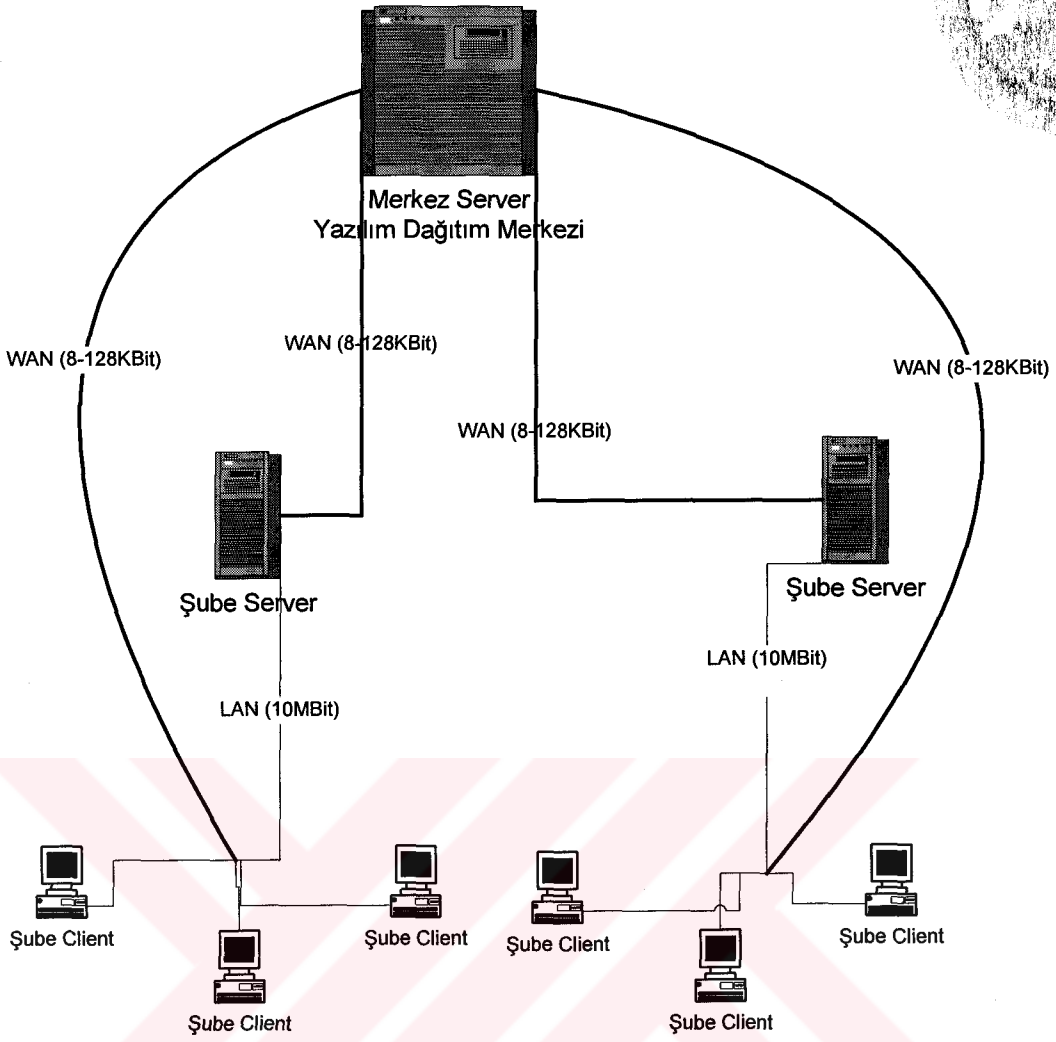
Yazılım dağıtımından sonra bütün hedef bilgisayarlarda programların aynı sürümünün bulunması yanında, dağıtım sürecinin sorunsuz devam etmesi, ertesi iş gününe kadar sonuçlanması ve bilgisayar ağı trafiğine yüklenmemesi de gereklidir. Bu tez çalışmasında, yazılım dağıtım sürecinde problemlerin erken teşhis edilerek düzeltilmesi için sürecin merkezden canlı olarak izlenmesi için gerekli araştırmalar yapılmış, problemler tespit edilmiş ve bunların ışığında bir tasarım yapılmıştır.

1.1 Yazılım Dağıtım Sistemindeki Bilgisayarlar

Yazılım dağıtımı, aynı bilgisayar ağında yüksek miktarda bilgisayarı olan kuruluşlarda genelde periyodik olarak, bazen de yazılım kaynaklı acil problemlerin (mesela bir yazıcı sürücü programının yanlış çalışması) giderilmesi için yapılır.

Yazılım dağıtım sistemlerinde yönetim ve hız açısından merkezi bir sistemin bulunması ve bütün konfigürasyon ve dağıtılacak dosyaların merkezdeki ana bilgisayarda bulunması artık bir standart haline gelmiştir. Gerek istemci (client), gerekse sunucu (server)'lara gidecek olan dosyalar, öncelikle gideceği dizine göre yada uygulama tipine göre gruplara ayrılırlar. Bu grupların ve her bir dosyanın versiyon numaraları tespit edilir. Bu dosyalar, kullanılan bir yardımcı program vasıtasıyla gidecekleri dizin ve versiyon numaraları belirtilerek ana makineye transfer edilir. Buradan yine bazı yardımcı programlar vasıtasıyla gerekli konfigürasyonlar yapılarak hangi bilgisayarlara hangi dosyaların hangi versiyonlarının gideceği ayarlanır.

Günümüzde örnek bir sistem olarak bir kuruluşun bir şubesini göz önüne aldığımızda çoğunlukla client/server (istemci/sunucu) bir mimariyle karşılaşırız. Böyle bir şubede client'lar ve server genellikle bir LAN (yerel alan ağı) oluştururlar ve birbirlerine 10 Mbit/sn ile 100 Mbit/sn arasında hızlı sayılabilecek bir hız ile bağlıdırlar. Fakat bu şubenin server ve client'larının merkezdeki ana sisteme bağlantısı WAN (Geniş Alan Ağı) ile olduğundan hızları genelde 9600 bit/sn ile 128 kbit/sn arasında değişir. Yazılım dağıtım için kullanılacak ana bilgisayar ise kuruluşun merkezindedir. Bu sistemler Şekil 1.1'de gösterilmiştir.



Şekil 1.1 Yazılım Dağıtım Ortamındaki Bilgisayarlar ve Bağlantıları

Şubede yazılım dağıtımını açısından genellikle iki değişik model bulunur; server modeli ve client modeli. Server modeli, sadece server'a gidecek olan ve genellikle client'ların server üzerinden kullandığı dosyalardır. Buna ek olarak bazı uygulamalar için server'da tutulan veritabanına da bir takım tablolar gönderilebilir. Client modeli ise, sadece client'lara gidecek dosyaları içerir. Bazı özel durumlarda birden fazla client modeli olabilir. Bu durumda her bir client için farklı modeller tasarlanır.

1.2 Yazılım Dağıtım Aşamaları

Burada incelenen adımlar Novadigm firmasının geliştirdiği ve dünyada çok yaygın olarak kullanılan EDM ürününün yazılım dağıtım aşamaları baz alınarak

4

çıkarılmıştır. Fakat yaygın olarak kullanılan diğer ürünlerde de hemen hemen aynı aşamalar mevcuttur.

EDM ürününde yazılım dağıtımı alacak olan her bir bilgisayar için kuruluş aşamasında bir Userid (kimlik adı) tanımlanır. Bilgisayarlar merkeze her bağlantılarında bu kimlik adlarından tanınırlar. Ayrıca dağıtılacak olan dosyalar, birbirleriyle olan ilişkilerine ve yazım amaçlarına uygun olarak gruplanıp servislere ayrılırlar; muhasebe ile ilgili olanların muhasebe servisi olarak, vergi ile ilgili olanların vergi servisi olarak gruplandırılması gibi. Ayrıca her servisin ve dosyanın bir versiyon numarası vardır. Dosyaların yeni versiyonu çıktığında versiyon numaraları arttırılarak eski versiyonlardan ayırt edilmeleri sağlanır. Servisin versiyonu hangi numaraya ayarlanırsa o servise bağlı olan dosyalardan o versiyon numarasına sahip olanların aktif olması sağlanır. Bütün dosyalar merkezdeki bilgisayarda sıkıştırılmış formatta saklanır. Her bir dosya için bir dizin bilgisi de tanımlanır. [SAS, 1997]

Yazılım dağıtım aşamaları genel olarak şu şekildedir :

- i) Bilgisayar kendi üzerindeki servisleri ve versiyonlarını tespit eder.
- ii) Bu versiyonlardaki dosyaların diskte mevcut olup olmadığı kontrol edilir.
- iii) Kimlik adı ile merkeze erişilir ve kendisi için tanımlanmış olan yeni versiyonları ve / veya servisleri tespit eder.
- iv) Elde edilen yeni bilgilerle kendi elinde mevcut olanlar kıyaslanıp eğer varsa aradaki fark bulunur ve getirilecek dosyalar tespit edilir.
- v) LAN'da bu dosyaları bulabileceği bir server tanımlanmışsa bu dosyaları oradan kopyalar. Eğer böyle bir tanım yoksa yada server'a ulaşamıyorsa merkezden kopyalamaya başlar.
- vi) Kopyalanan ve ortak bir dizinde tutulan dosyaları gerçek dizinlerinde aktif eder.
- vii) Merkeze işlemin bittiği ile ilgili bilgiler gönderilir.

Bütün bu aşamalarda işlemleri gerçekleştiren programlar EDM’de ayrı birer programdırlar ve rexx diliyle yazılmış bir programdan belli bir düzene göre çağrılırlar. Bu programların her birinin kendi ürettikleri log dosyaları mevcuttur fakat bu bilgiler merkezde mevcut değildir. Yapılan işlemlerin çoğu bilgisayarın kendi üzerinde yapılmaktadır.

1.3 Yazılım Dağıtım Sürecinde Yaşanabilen Problemler

Yazılım dağıtımı gönderilen yazılımın büyüklüğüne göre çok uzun sürebilen ve iletişim hatlarını oldukça meşgul edebilen bir işlemdir. Gündüzleri mesai saatlerinde iletişim hatları yoğun olduğundan ve şubelerin merkez ile iletişiminin aksamaması gerektiğinden, yazılım dağıtımı genellikle geceleyin yapılır ve bu dağıtımın sabah mesai saatinden önce bitmesi ve herşeyin düzgün çalıştığından emin olunması gerekir. Profesyonel yazılım dağıtım ürünlerinin hepsinin bir raporlama aracı vardır ve bu araçlar belli bir hedef bilgisayar için yazılım dağıtım işleminin bitip bitmediğini yada bitenlerin başarılı olup olmadığını belli dosyalara yazıp kaydetmektedirler [SAS, 1997].

Dünyanın en büyük yazılım dağıtım ürünlerinden IBM firmasının Netview Distribution Manager’ı, Novadigm firmasının Enterprise Desktop Manager’ı ve Computer Associates firmasının TNG Unicenter’ı örnek olarak ele alındığında bu tip raporlamalar görülür. Halbuki yazılım dağıtımı bu tür büyük ürünler kullanan büyük kuruluşlarda saatlerce süren ve oldukça karmaşık bir işlemdir. Sadece sonuç raporu yetersizdir. İletişim ağlarındaki problemlerden, hedef bilgisayardaki disk yetmezliğinden veya disk bozukluğundan, işletim sistemi kaynaklı bazı uyumsuzluklardan yada merkezdeki yazılım dağıtım ürünüyle haberleşen programın kendisinin bozulması gibi bir takım sebeplerden ötürü yazılım dağıtım süreci bir çok bilgisayarda yarıda kesilebilmekte ve bunların hangi aşamada kaldığı, en son ne işlemi yaptığı gibi çok önemli bilgilere sahip olunamamaktadır [DELİPINAR, 2000].

Tipik bir yazılım dağıtım aşamasında aynı dosyaların birden fazla client’a gideceği düşünüldüğünde ilk dikkat çekecek olan problem bu dosyaların her seferinde merkezden WAN üzerinden getirilmesidir. Bu sorunu aşmak için bu dosyaların sadece bir kez merkezden getirilmesi ve şubedeki server’da tutulması

gerekmektedir. Diğer client'lar bu dosyaları kopyalayacağı zaman server'dan kopyalamalı ve boşuna merkezden getirmeye çalışmamalıdır. [ZAMPROGNO, 1998] Eğer kazayla bile olsa sistem bu duruma düşerse bilgisayar ağı fazlasıyla meşgul olur ve yavaşlar. Yazılım dağıtımı geceleyin başlasa bile bu durumda client'ların bir çoğu yazılım dağıtımını sabaha kadar yetiştiremezler [DELİPINAR, 2000].

İster ticari olsun isterse kuruluşların kendi geliştirdikleri olsun, bütün yazılım dağıtım programlarında bu engel aşılmalıdır. Bunun için ilk önce server rolünde çalışan bilgisayar, hem kendi modelinde yer alan hem de client modelinde yer alan dosyaları merkezden almalıdır. Bu aşama bazen saatler sürebilir. Bu aşama tamamlanıp bütün dosyalar geldikten sonra client'ların yazılım dağıtımı başlatılmalıdır. Bu çalışma mantığı her ne kadar programlarla otomatik olarak gerçekleştirilse de pratikte bazı sorunlara rastlanabilmektedir. Bu sorunları daha iyi tespit edebilmek için şubedeki bir bilgisayarla merkezdeki yazılım dağıtım ana bilgisayarı arasında yazılım dağıtımı boyunca yapılan konuşmayı ve çalışan programları adım adım incelemek ve izlemek gerekir.

Pratikte sık yaşanan bir sorun da merkezdeki bilgisayara aşırı yüklenme ve dolayısıyla bu bilgisayarda hafıza veya işlemci kullanımının % 100'e dayanması ve bu yüzden bu ana bilgisayarın neredeyse durma noktasına gelmesidir. Bu problem de kendisini bütün bilgisayarların en son bulundukları adımda takılı kalmaları ve uzun bir süre hiçbir ilerleme olmaması şeklinde göstermektedir [DELİPINAR, 2000].

Diğer bir sorun ise bilgisayarların kendileri için tanımlanmış Server'ı, tanım eksikliğinden / bozukluğundan yada başka bazı problemlerden dolayı bulamamaları ve her birinin dosyaları merkezden getirmeye çalışmasıdır [ACAR, 1998]. Tahmin edileceği gibi bu durum, yazılım dağıtım sürecinin zamanını client sayısı kadar katlar.

1.4 Yazılım Dağıtım Sürecinin Canlı Olarak İzlenmesi

Yazılım dağıtımını canlı olarak izlendiği takdirde, üretim ortamında sık rastlanan, client'larda ve server'larda yanlış konfigürasyon yada merkezden yanlış / eksik servis atanması, versiyonların yükseltilmesinin unutulması gibi problemler henüz yazılım dağıtım aşamasında tespit edilebilir. Mesela izleme bilgisayarında bir server'ın yazılım dağıtım esnasında bütün aşamaları yaptığı halde merkezden dosyaların getirilmesi aşamasını yapmaması bize server'ın eksik dosyası olmadığını göstermiş olur. Fakat aslında bu imkansızdır. Çünkü client'lara gidecek dosyaların mutlaka server'a da gitmesi gerekir ve yazılım dağıtımını zaten yeni dosyalar göndermek için yapılmaktadır. Dolayısıyla bu durum bize server'a yanlış servis yada yanlış versiyon atandığını yada dosyaların yanlış isimlendirmelerle ana makineye yüklendiğini gösterir. Eğer bir izleme sistemi olmazsa, bu hata ancak ertesi sabah kullanıcıların yeni dosyaların gelmediğini farketmeleri sebebiyle ortaya çıkar ve iş işten geçmiş olur. Bu hatanın düzeltilmesi çok uzun zaman alabilir ve bazen, kritik uygulamalarda şubedeki bilgisayarların uzun zaman hizmet vermesini engelleyebilir. Bu tip problemlerin önceden yani yazılım dağıtım esnasında farkedilmesi ve gerekli değişikliklerin yapılıp yazılım dağıtımının tekrar başlatılması büyük bir kuruluş için çok önemlidir [DELİPINAR, 2000]. Bunun gibi örnekler çoğaltılabilir.

Günümüzde işletim sistemleri içinde anlık CPU kullanımı, anlık hafıza kullanımı, harddisk'ten okunan byte sayısı, çalışan toplam işlem sayısı, bilgisayarın ağ kartına gelen giden TCP paket sayısı vs. gibi birçok değişik bilgiyi izleme ve istendiğinde bir veri tabanına kaydetme imkanı veren yardımcı programlar mevcuttur. İşletim sistemiyle birlikte gelenler dışında ayrıca bu işlemleri kapsayan ve çok daha gelişmiş ve yüzlerce bilgisayarı tek bir ekrandan merkezi olarak izleme imkanları sunan büyük ticari yazılımlar da mevcuttur. Bu yazılımlar sayesinde izlenen değişkenlere belli sınırlar atanabilir ve bu sınırlar belli bir süre aşıldığında uyarı mesajları çıkabilir. Bu yazılımların yaptığı, genel olarak sistem bileşenlerini izlemektir. Bu konu hakkında yapılan çalışmaları incelediğimizde karşımıza "process monitoring – işlem izleme" ile ilgili bazı temel sorunlar ortaya çıkar :

i) Neyin ve nasıl izleneceğinin kesin olarak tespit edilmesi gereklidir

[GRANT and HIGGINS, 1996].

- ii) İzlenecek olan sistemlerin hangi aşamada seyrettiğini anlamak için çalışan programların davranışına müdahale etmek gerekir. Fakat genelde bu programlar birden fazladır ve her işletim sistemi için ayrı ayrı parçaları vardır. Bu da müdahaleyi oldukça zorlaştırır. [KAMINKA, 2001, BARGHOUTI and KRISCHNAMURTHY, 1995]
- iii) İzlenecek olan sistemlerde haberleşme için kullanılacak olan bilgi izlenecek olan işleme göre değişiklik arzedebilir [KAMINKA, 2001].
- iv) İzleme işlemi, hem izlenen bilgisayarlara işlemci ve hafıza olarak ek yük getirecektir hem de bilgisayar ağının meşguliyetini arttıracaktır. Özellikle hatların yoğunluğu direk olarak bilgisayar sayısı ile doğru orantılıdır [KAMINKA, 2001].
- v) Bilgisayar ağındaki gecikmeler göz önüne alınmalıdır [JOYCE et al, 1987].
- vi) İzlenecek olan bilgisayarlarla iletişimin güvenli ve sağlam olması gereklidir [KAMINKA, 2001].

Şirketlerin kendi bilgisayar ağları kullanıldığı için güvenlik problemi ortadan kalkmıştır. Bu tez çalışmasında, çalışan programların kodlarına müdahale edilmeden, sadece üretilen log dosyaları incelenerek yada yazılım dağıtım prosesinin kendi sağladığı bazı adımlardan faydalanarak sistemin hangi aşamada olduğu tespit edilmiştir.

Tez çalışmasında örnek olarak kullanılan EDM’de bütün yazılım dağıtım aşamalarındaki işlemleri gerçekleştiren süreçler ayrı birer programdır ve rexx diliyle yazılmış bir programdan belli bir düzene göre çağrılırlar. Bu programların her birinin kendi ürettikleri log dosyaları mevcuttur. Bu “.exe“ programlarına kısaca değinelim :

EDMCON00.exe - Client ve merkezdeki yazılım servislerini karşılaştırır.

EDMCON20.exe - Client ve merkezdeki dosyaları karşılaştırır.

EDMCON30.exe - Karşılaştırma sonrası hangi dosyaların getirileceğini belirler.

EDMSTGRQ.exe - LAN'daki Server'da olan dosyaları oradan getirir.

EDMGETRS.exe - LAN'daki Server'da olmayan dosyaları merkezden getirir.

EDMCON40.exe - Getirilen dosyaları kendi dizinlerine koyar.

EDMSELRS.exe - Birden fazla versiyonu olan dosyaların hangisinin aktif olacağına karar verir.

EDMACTRS.exe – Versiyonu seçilen dosyayı aktif eder. O dosya için metod tanımlıysa EDMCON50.exe' yi çağırır.

EDMCON50.exe – Metod çalıştırır. Mesela bir .C dosyası kopyalandıktan sonra derlenecekse, yada bir .ZIP dosyası açılacaksa bu dosyanın tanımında belirtilen komut satırıyla bu işlem gerçekleştirilir.

EDMSTATE.exe – Sürecin dosya ve servis özeti hazırlanır.

Yazılım dağıtımı sırasında server'ların ve client'ların hangi aşamada olduğunu anlamamanın tek yolu bu süreci canlı olarak izlemektir. Canlı olarak izlemek için ise yazılım gönderilen hedef bilgisayardan merkezdeki bir veya birkaç bilgisayara belli aralıklarla ya da belli adımlarda bilgi gönderilmesi ve merkezde bu bilgilerin değerlendirilmesi gereklidir. Bir yazılım dağıtım ürünü gibi oldukça özel bir sistemi incelemek için ise bu araçların özel olarak geliştirilmesi gerekmektedir.

İşletim sistemlerinin hemen hepsi TCP/IP protokolünü desteklemektedir. Bu işletim sistemlerinde Cobol, C, Java, Rexx, Visual Basic, Pascal, Perl vs. gibi programlama dillerinden uygun olan herhangi biri kullanılarak TCP socket programlama yapılabilir ve merkezdeki bilgisayara istediğimiz formatta mesajlar göndermek mümkün olur. Hatta internetten araştırıldığında yada bu ortamların kaynak kitaplarına bakıldığında ihtiyaç duyulan işlevi yerine getiren örnek program kodları da bulunmaktadır.

Hem mesajları gönderen programın hem de gönderilen mesajları karşılayacak olan programın sıfırdan yazılması çok büyük bir hareket alanı ve esneklik sağlar. Bu durumda geliştirilecek olan yazılım ihtiyaca göre istendiği gibi değiştirilebilir. Mesajları karşılayacak programın kullanıcı arayüzü ve işlevselliği de istenilen şekilde değiştirilebilir.



2. İZLEME SİSTEMİ TASARIMI

2.1 TCP/IP Protokolünde TCP ve UDP Metodları

Merkezden uzaktaki çok sayıdaki bilgisayardaki işlemleri canlı olarak izlemenin yaygın olarak kullanılan ve tercih edilen yolu bilgisayar ağlarının getirdiği haberleşme protokollerini kullanan programlama teknikleriyle iletişim kurmak ve bilgisayarları karşılıklı mesajlaştırmaktır. Şu anda bütün dünyada en yaygın olarak kullanılan haberleşme protokolü TCP/IP (Transmission Control Protocol / Internet Protocol) dir. Piyasadaki güncel olan programların hemen hepsi bu protokolü kullanmaktadır. Bu protokol diğerlerine nazaran hızlı, güvenilir ve yönlendirilebilir (routable) olması sebebiyle tercih edilmiştir. TCP / IP protokolünde temel olarak kullanılan iki tür metod vardır :

i) Çift taraflı bağlantı üzerine dayalı TCP

ii) Bağlantı gerektirmeyen UDP

Temel olarak her iki metod da aynı işi yapmaktadır. Aralarındaki en büyük fark ise şöyle açıklanabilir; TCP'de iletişim sağlamak ve veri transferi yapmak için iki bilgisayarın karşılıklı olarak bağlantı kurması ve bağlantı kurulduğuna dair karşılıklı onay gönderilmesi (Acknowledge) gerekir. Bu bağlantıdan sonra veri transferi yapılabilir. UDP (User Datagram Protocol)'de ise iletişim için karşılıklı onay gerektiren bir bağlantı gerekmez. Mesaj gönderilecek olan bilgisayarın IP adresine yada Host ismine belli bir port numarasından mesaj yada veri yollanır. Yollanan bu verinin karşı tarafa ulaşmış olup olmadığı kontrol edilmez. Bu metod daha güvensiz gibi gözükse de TCP'ye göre daha hızlıdır. TCP yada UDP kullanmaya karar vermek biraz da yapılacak uygulamanın çalışma prensibine bağlıdır. Eğer aynı anda sadece iki bilgisayar bir bağlantı kurup veri transferi yapacaksa (mesela ftp gibi bir dosya transferi işi) yada **Telnet** gibi sürekli komutlar gönderilip cevapları alınacaksa bağlantıya dayalı olan TCP tercih edilir. Fakat yüzlerce ve hatta binlerce bilgisayardan merkezdeki bir bilgisayara aynı anda mesajlar gelecekse bu durumda

merkezdeki tek bir bilgisayara aynı anda yüzlerce yada binlerce bağlantı kurulması gerekir ki bu sorunlara sebep olur. Böyle bir durumda bağlantı gerektirmeyen UDP tercih edilir.

Günümüzde bilgisayar ağlarında yaygın olarak kullanılan SNMP (Simple Network Management Protocol) mesajlaşmasında UDP protokolü kullanılmaktadır. Yine IBM'in Netfinity adlı uzaktan erişim ve mesajlaşma programının mesajları UDP ile yollanmaktadır.

Windows işletim sistemlerinde TCP ile bağlantı kullanıldığında her bağlantı için karşı tarafta hangi bilgisayara bağlı olduğunun program tarafından takip edilmesi gereklidir çünkü bir kere bağlantı kurulduktan sonra bu bağlantı üzerinden veriler gider ve gelir. Yeni nesil, Windows tabanlı görsel bir programlama dili (örneğin Visual Basic, Delphi vs) kullanıldığında aynı anda bağlantı yapılacak her bir bilgisayar için ayrı bir TCP objesi kullanılması gereklidir. Çok sayıda bağlantı kurulup veri transferi yapıp bağlantının koparılması ve bunun tek bir TCP objesi ile yapılması denenmiş ve problem yaşanmıştır. UDP'de ise UDP objesi belli bir port'tan mesajları dinler. Bir bilgisayar bu port'tan mesaj gönderdiğinde, gönderilen mesaj tampon belleğe (buffer) gelir ve ne kadar çok mesaj gelirse gelsin bilgisayar bunları bu tampon bellekten okuyup gerekli işlemleri yapabilir.

Bu tez konusunda da yüzlerce bilgisayarın üzerinde çalışan işlemlerin durumu izleneceğinden bu bilgisayarlardan aynı zamanda yüzlerle ifade edilebilecek sayıda mesaj beklendiği için UDP metodu tercih edilmiştir.

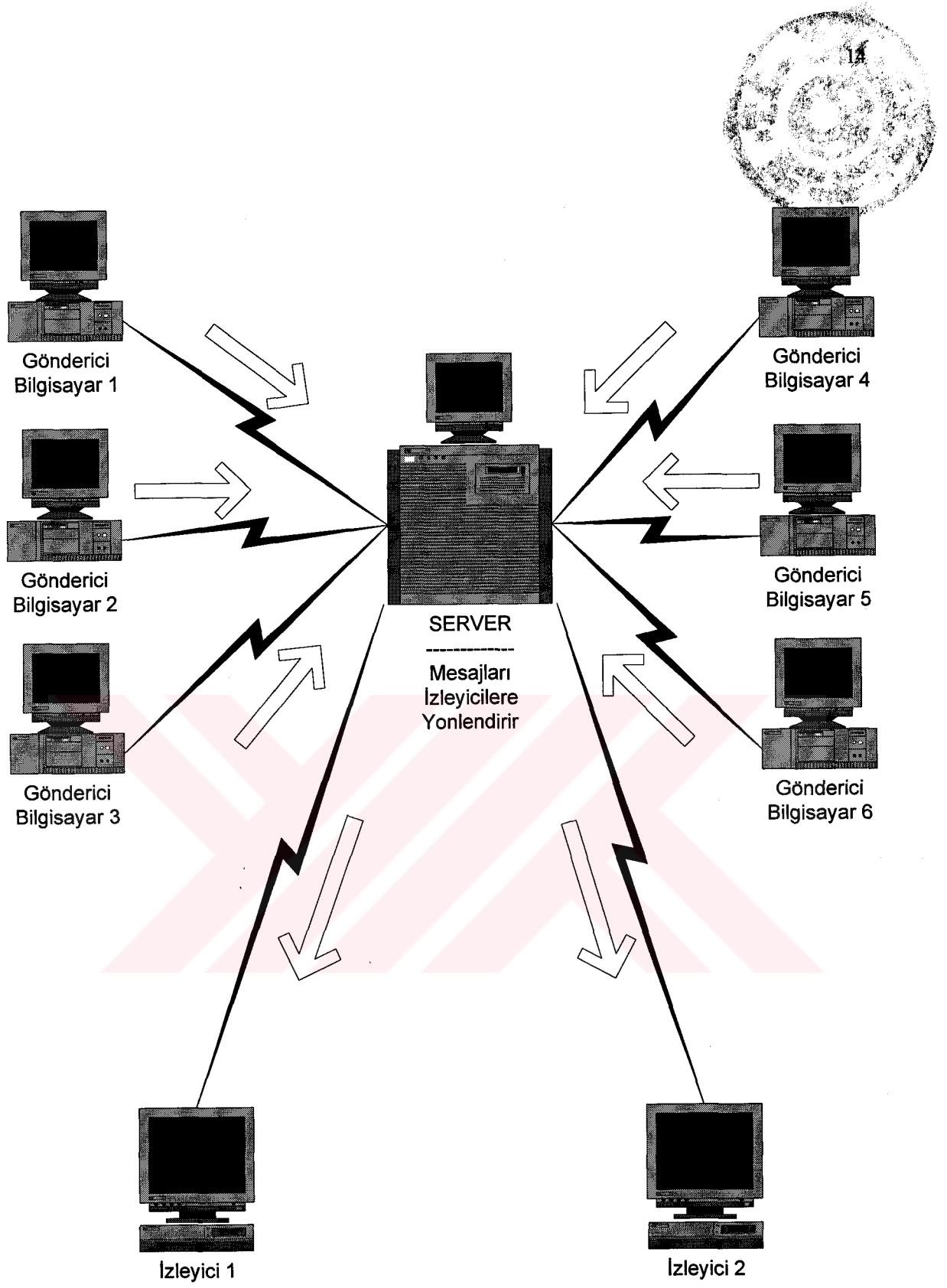
2.2 Sistem Tasarımı

Bilgisayarlardaki herhangi bir işlemi izleyebilmek için o bilgisayarların merkeze işlemin neresinde olduğunu belirten bir mesaj (veri) göndermesi gereklidir. Bunun sağlanması için UDP kullanan ve belirtilen IP adresine belirtilen port numarasından verilen mesajı gönderen bir gönderici program gereklidir. Merkezde de bu programın gönderdiği mesajları alıp yorumlayan ve gerekli değerlendirmeyi yaparak

kullanıcıya bilgi mesajı gönderen bilgisayardaki işlem durumu hakkında anlamlı bilgiler veren bir izleme programı gereklidir [WYBRANIETZ and HABAN, 1988].

Burada karşımıza şöyle bir problem çıkıyor : Gönderici programın her zaman merkezdeki bir veya daha fazla sayıdaki bilgisayara ve dolayısıyla bunların belli olan IP adreslerine mesaj göndermesi gerekir. Eğer merkezdeki bu izleyici bilgisayarların IP adresi değişirse veya yeni bir bilgisayardan izlemek gerekirse mesaj gönderici programın hedef adres parametrelerinin yeni bilgisayarı (IP adresini) da içerecek şekilde değiştirilmesi ve bu değişikliğin dağıtık ortamdaki yüzlerce, binlerce bilgisayara gönderilmesi gerekecektir ki bu fazladan bir külfet olur.

Çözüm olarak şöyle bir metod geliştirilebilir : Gönderici bilgisayarlar ile izleyici bilgisayarlar arasına merkezde server (sunucu bilgisayar) olarak görev yapan, IP adresi sabit olan üçüncü bir bilgisayar konur, bkz. Şekil 2.1. Bütün gönderici bilgisayarlar bu server bilgisayara mesaj gönderirler, bu bilgisayar üzerinde çalışan bir program ise gelen mesajları izleyici bilgisayarlara gönderir. Bu programa “reporter” (raporlayıcı) da denir [TOKUDA et al, 1988].



Şekil 2.1 Mesaj alışverişi yapan bilgisayarlar

İzleyici bilgisayarın IP adresinin değişmesi yada yeni izleyici bilgisayarlar eklenmesi durumunda merkezde bulunan server bilgisayardaki programda değişiklik yapmak yeterlidir. Eğer bu işlemin dinamik olarak yapılması istenirse, yani herhangi bir bilgisayara izleyici program yüklenir ve server bilgisayarda manuel bir değişiklik yapılmasına gerek duyulmadan otomatik olarak mesajların artık bu yeni izleyici bilgisayara da gönderilmesi gerekirse bu durumda izleyici program çalışmaya başlarken server program ile bağlantı kurmalı ve artık gelen mesajları kendisine de göndermesini istemeli ve server program da kendi üzerinde buna uygun değişiklikleri yapmalıdır. Bu bağlantı ise yine TCP/IP ile kurulur ve veri transferi yapılır. İzleyici program ile Server program'da önceden tanımlanmış bir takım anlamlı mesajlar olmalıdır [BARGHOUTI and KRISCHNAMURTHY, 1995].

Böylece Server programı hangi mesaj geldiğinde bu izleyiciyi mesajın yönlendirileceği bilgisayarlar grubuna dahil edeceğini bilmelidir. İzleyici program, Server programa ilgili mesajı iletir ve talebinin gerçekleştiğinden emin olmak için bir cevap bekler. Server program da kendisine gelen bu anlamlı mesajı gönderenin IP adresini alır ve bu IP adresini mesajın yönlendirileceği IP adresleri listesine ekler. İzleyici bilgisayara işlemin başarıyla tamamlandığına dair anlamlı bir mesaj gönderilir. İzleyici program da beklediği bu onay mesajını alarak işlemin başarıyla tamamlandığını anlar ve gerekirse bunu ekrana bir mesajla bildirir. Böylece aynı Server bilgisayarın bulunduğu bilgisayar ağına bağlanabilen herhangi bir bilgisayar izleyici olarak kullanılabilir. O halde sistem için üç çeşit bilgisayar (yazılım) gereklidir :

- i) Gönderici Bilgisayar (UDP mesaj gönderir)
- ii) SERVER Bilgisayar (Gönderici'den gelen mesajları İzleyici'lere yönlendirir.)
- iii) İzleyici Bilgisayar (Gelen mesajları yorumlayıp kullanıcıya anlamlı hale getirir ve gerekirse grafik olarak ekranda gösterir.)

Bu üç tip bilgisayar (Gönderici, Server ve İzleyici) ve yaptığı işler Şekil 2.1'de şematik olarak gösterilmiştir. Bu yapının avantajları ise gönderici bilgisayarın her zaman sadece bir tek IP adresine (Server'a) mesaj göndermesi ve merkezdeki izleyicilerin sayısının ve IP adreslerinin değişmesinin mesajı gönderecek olan

yüzlerce bilgisayarı hiç etkilememesi ve böylece herhangi bir değişiklik yapmaya gerek kalmamasıdır.

2.3 Server Programının Ek Görevleri

Server programının çalışacağı Server bilgisayarın üzerine yüklenmiş olan hizmeti verebilmesi için her gün 24 saat çalışması gerekir. Ayrıca bu bilgisayar Server olduğu için oldukça güçlü ve disk alanı yeterli olması gereklidir. Böyle sürekli açık olan ve güçlü olan bir makinanın kurulacak olan bu sistemde sadece gelen mesajları yönlendirmesiyle yetinmek dışında başka yararlı görevler de üstlenmesi sisteme ek avantajlar getirir. Ek avantajlar iki ana grupta incelenebilir :

- i) Server'a gelen mesajların belli isimli dosyalarda kayıt edilmesi.
- ii) İstenildiğinde yeni açılan bir İzleyici Bilgisayara kaydedilen mesajların gönderilmesi

2.3.1 Server'a Gelen Mesajların Kayıt Edilmesi

Şu ana kadar yapılan sistem tasarımına göre, mesajlar Server tarafından İzleyici bilgisayarlara gönderilir ve Server üzerinde herhangi bir arşiv yoktur. Server bilgisayar üzerinde gelen mesajların kayıt edilmesi hem daha sonra geçmişe yönelik olarak mesajların incelenebilmesini hem de kayıt altına alınan mesajların istenildiğinde başka izleyici bilgisayarlara da aktarılmasını olanak sağlar. Bir yazılım dağıtım süreci bazan akşamdan başlayıp sabaha kadar sürer. Bu yüzden yazılım dağıtımını izleyecek olan kişiler işyerinden değil de evden de izlemek isteyebilirler. Bu durumda herşeyi izleyebilmek ve olası problemlerden haberdar olabilmek için evdeki bilgisayarın yazılım dağıtım süreci boyunca bazan sabaha kadar açık olması ve SERVER ile aynı bilgisayar ağında olması gerekir. Aksi takdirde gelen mesajlar kaybolmuş olur. Sonradan durum izlenemez. Fakat Server Bilgisayar gelen mesajları kaydederse bu mesajlar arşivlenmiş olur.

2.3.2 İzleyici Bilgisayara Kaydedilen Kesajların Gönderilmesi

Server'da kaydedilen mesajların istenildiğinde İzleyici Bilgisayarlara gönderilebilir. Fakat mesajlar Server'da bir tek dosyaya kaydedilmişse bu durumda çok eski yazılım dağıtımlarının mesajları bile gönderilir. Bunun önüne geçmek ve sadece belli bir yazılım dağıtım sürecinin mesajlarını almak için her yazılım dağıtımının başlangıcında İzleyici Bilgisayar'dan Server'a bağlantı kurulur ve yeni bir yazılım dağıtım sürecinin başladığına dair mesajlaşmalar yapılır. Bu yeni süreçte günün tarihini de içeren bir isim verilebilir. Böylece Server Bilgisayarın o andan itibaren gelen mesajları belirtilen isimli bir dosyaya yazması sağlanır. Yazılım dağıtım süreci başladıktan sonra bile sisteme yeni giren bir İzleyici Bilgisayar istendiğinde Server Bilgisayar ile haberleşerek kayıtlı olan mevcut yazılım dağıtım süreçlerinin listesini alabilir ve seçtiği bir isme ait olan Yazılım Dağıtımının başından beri gelen mesajları kendisine isteyebilir. Bu durumda kendisi sistemde değilken Server'a gelmiş olan mesajları kendisine almış ve kendisini güncellemiş olur.

3. GELİŞTİRİLEN PROGRAM

Sistemin Teorisi bölümünde yapılan tasarımda ortaya birbiriyle haberleşen 3 tip bilgisayar programı çıkmaktadır. Bu programların her biri Delphi 5.0 Enterprise Edition'da tasarlanmış ve derlenmiştir.

3.1 Gönderici Program

Gönderici program, sistemde yer alan 3 tip bilgisayar programı arasında en basit yapıda olanıdır. Bu program gönderici bilgisayarlarda izlenen işlemlerin durumunu mesajla Server programa göndereceğinden her durum değişikliğinde belli parametrelerle çağırılması gerekmektedir. Bu yüzden bu program komut satırından parametre girilerek çağırılır. Bu parametreler; mesajın gönderileceği bilgisayarın IP adresi, kullanılacak UDP port numarası ve gönderilecek mesajdır. İzleyici bilgisayarın mesajın kimden geldiğini anlayabilmesi için bütün gönderici bilgisayarlara bir kullanıcı kodu atanır ve bu bilgisayarlar kullanıcı kodlarından tanınırlar. Gönderici bilgisayarlar da gönderdikleri mesajın başına kullanıcı kodunu koyarlar, sonra ' : ' koyup mesajı eklerler. Kullanıcı kodunun kolay girilebilmesi için bunu komut satırından ayrı bir parametre olarak vermek uygun olur. Böylece gönderici programın 4 tane parametresi olur. Mesaj parametresi, kullanıcı kodu ve gönderilecek mesaj olarak ikiye ayrılmıştır.

Gönderici programın adı "sendudp.exe" olsun. Bu durumda komut satırında şöyle yazılacaktır :

Sendudp.exe 10.8.8.150 6666 TEST_001 25

Gönderici program kullanıcı kodu ile gönderilecek mesajı araya ' : ' koyarak birleştirir ve "10.8.8.150" Ip adresli bilgisayara 6666 nolu porttan "TEST_001 : 25" mesajını UDP ile gönderir. TEST_001 yerine 255 karakter uzunluğa kadar herhangi bir kullanıcı kodu ve yine 25 yerine de 255 karakter uzunluğa kadar herhangi bir mesaj yazılabilir. Gönderici programda bir UDP objesi kullanılmıştır. Bu objenin adı *NMUDPI* dir. Bu objenin Ip adresi, port numarası gibi alanlarına gerekli değerler

atanarak mesaj gönderilebilir. Komut satırından girilen parametrelerin sayısı *ParamCount* değişkeninde, kendileri ise *ParamStr(1)* değişkeninde bulunur. Bu Mesajı gönderen kod parçası EK-1'dedir.

3.2 Server Program

Server program Sistem tasarımında gönderici program ile izleyici program arasında kalan ve köprü görevi gören programdır. Bu programın IP adresi sabit olan bir server bilgisayara kurulması ve orada 24 saat çalışması gerekmektedir. Server'ın en temel görevi kendisine gelen mesajları izleyici bilgisayarlara yönlendirmektir. Bunu yapabilmek için mesajları göndereceği izleyici bilgisayarların IP Adreslerine ihtiyacı vardır. Teori kısmında bahsedilen ve otomatik olması tercih edilen işlemler şöyle gerçekleşir :

İzleyici Bilgisayar Programı ilk açıldığında ya da istendiği zaman menü adımından istenilen bir Server'a (IP adresine) kendini kayıt ettirebilir. Bunun için Server'ın IP adresi girilir ve onaylanır. İzleyici bilgisayar bu IP adresine 'BANA DA YOLLA' UDP mesajını gönderir.

Server, kendisine gelen 'BANA DA YOLLA' mesajının hangi IP adresinden geldiğini tespit eder. (Delphi 5.0'daki UDP objesi mesajı gönderen IP adresini bir değişkende tutar ve böylece bu adres bilgisine ulaşılabilir.) Bu IP adresini bir string dizisinde tutulan gönderilecekler listesine ekler. Bundan sonra kendisine gelen ve özel anlamı olan mesajlar dışındaki her mesajı bu dizide tutulan IP adreslerine yine UDP objeleri aracılığıyla gönderir. İşlemin başarıyla yapıldığının onayını da 'BANA DA YOLLA' mesajıyla kendisini kayıt ettiren izleyici bilgisayara gönderir.

İzleyici Bilgisayar, kayıt olma mesajını Server'a gönderdikten sonra 5 saniye bekler. Bu 5 saniye içinde Server'dan onay mesajı gelirse ekrana kayıt işleminin başarıyla yapıldığını belirten bir mesaj çıkar. Eğer 5 saniye içinde onay mesajı gelmezse ekrana Server ile iletişim kurulamadığına dair bir mesaj çıkar. Bu süreye "timeout" (zaman aşımı) denir ve bu değer ayarlanabilir. Eğer Server'a gelen mesajların kayıt altına alınması isteniyorsa İzleyici Bilgisayardan 'Yeni bir Yazılım Dağıtımını Başlat '

menü adımı seçilir ve Server'a yeni Yazılım Dağıtım adını da içeren ilgili mesaj gönderilir. Server bu mesajı alınca mesajın içinden yeni Yazılım Dağıtım adını alır ve bundan sonra gelen mesajları bu isimli bir dosyaya text modunda kaydeder.

Yazılım Dağıtım süreci bittiğinde İzleyici Bilgisayardan 'Yazılım Dağıtımını Bitir' menü adımı seçilir ve program Server'a ilgili mesajı gönderir. Server, gelen mesajı algılar ve eğer varsa o anda kayıt edilen Yazılım Dağıtımının ismini İzleyici Bilgisayara ilgili mesaj ile gönderir. İzleyici Bilgisayarda Yazılım Dağıtım ismi ekranda görüntülenerek bu Dağıtımın bitirilmesi için onay istenir. Onay verilirse Server'a gerekli mesaj gönderilir. Server bu onay mesajını alınca artık mesajları kayıt altına almaz. Fakat kendisinde tanımlı olan İzleyici Bilgisayarlara göndermeye devam eder. Yazılım Dağıtımını devam ederken ya da bittikten sonra herhangi bir izleyici bilgisayar kendisini Server'a kayıt ettirdikten sonra Server'dan kayıt altına alınan Yazılım Dağıtım mesajlarının kendisine gönderilmesini isteyebilir. Bunun için 'Yazılım Dağıtım Verilerini Server dan Getir' menü adımını seçer ve böylece Server'a ilgili mesaj gönderilir.

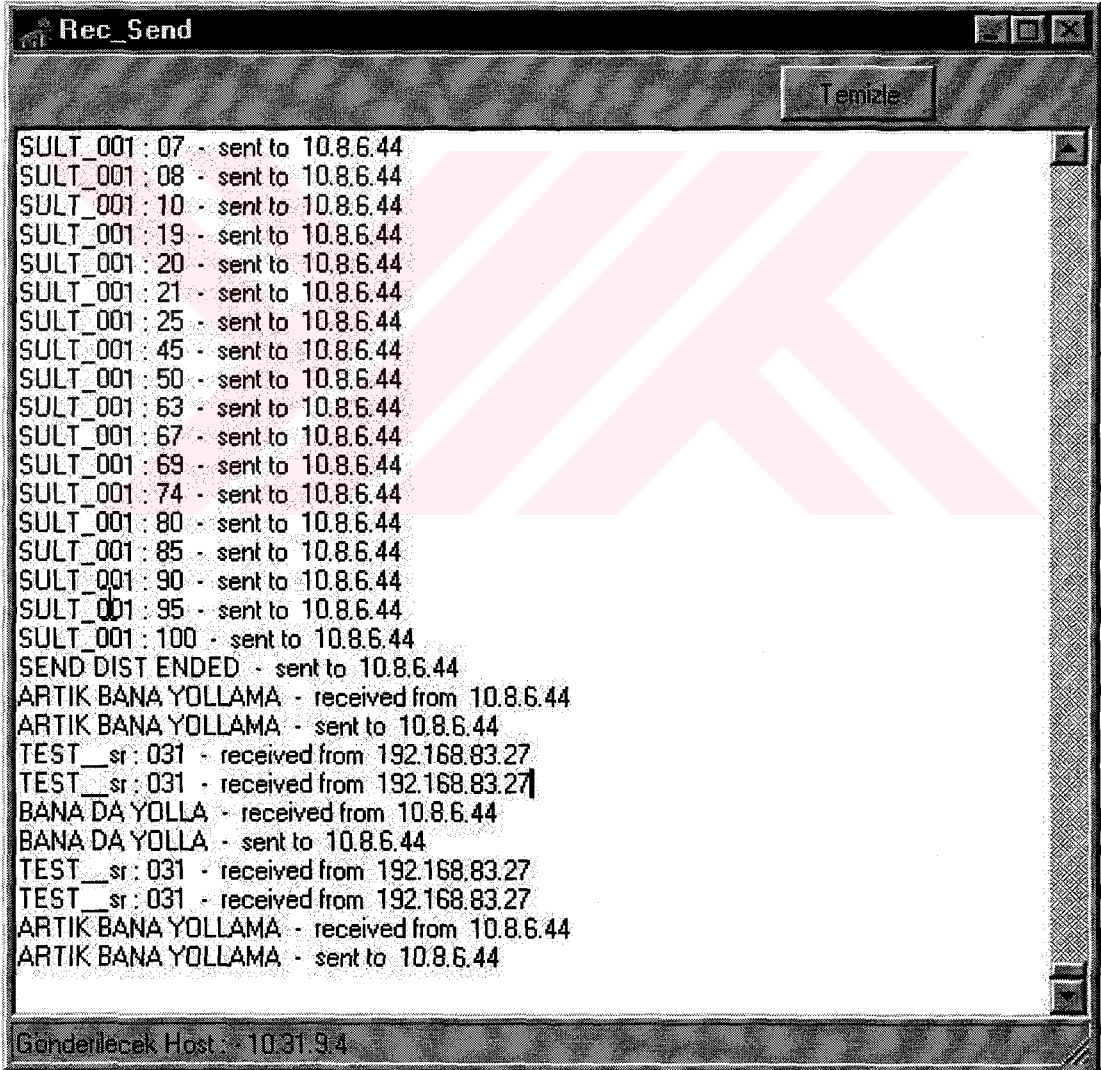
Server, gelen mesajı algılar ve mesajların kaydedildiği dizindeki bütün dosyaların isim listesini alıp bunları ilgili mesajlara ekleyerek İzleyici Bilgisayara gönderir. İzleyici Bilgisayarda bu isimler listelenir. Kullanıcı ister şimdi devam eden Yazılım Dağıtımının ismini isterse de eski bir Yazılım Dağıtımının ismini seçer ve onaylar. Bu seçilen isim ilgili bir mesajla Server'a gönderilir. Server, gelen mesajdan seçilen ismi bulur ve o dosyayı açıp içindeki bütün mesajları kendisinden isteyen İzleyici Bilgisayarlara gönderir. Server üzerinde kayıtlı birden fazla İzleyici bilgisayar olsa da Server bu mesajları sadece kendisinden isteyen İzleyici Bilgisayara gönderir, diğerlerine göndermez. Diğerlerine sadece gönderici bilgisayarlardan gelen mesajları eskisi gibi yönlendirmeye devam eder.

İzleyici Bilgisayardan istendiğinde yada programdan çıkarken otomatik olarak Server'dan Kayıt Silme ekranı gelir. Bu ekranda kayıt edilirken kullanılan Server'ın IP adresi otomatik olarak yazılır. Onaylandığında Server'a bu izleyici bilgisayarın listeden silinmesine ilişkin mesaj gönderilir.

Server bu mesajı alınca mesajı gönderen bilgisayarın IP adresini listeden çıkarır ve İzleyici Bilgisayara onay mesajı gönderir.

İzleyici Bilgisayar mesajı Server'a gönderdikten sonra 5 saniye bekler. Bu 5 saniye içinde Server'dan onay mesajı gelirse ekrana kayıt silme işleminin başarıyla yapıldığını belirten bir mesaj çıkar. Eğer 5 saniye içinde onay mesajı gelmezse ekrana yine Server ile iletişim kurulamadığına dair bir mesaj çıkar.

Server programı ayrıca gelen giden bütün mesajları ekrana da yazar. Örnek bir görüntü Şekil 3.1'de görülebilir.



Şekil 3.1 Server Programındaki Mesajlar

Server Programın İzleyici programla olan iletişimini sağlayan temel prosedür UDP kanalıyla önceden belirlenen port numarasından bir mesaj geldiğinde çalışan program parçasıdır. Bu prosedür yukarıda açıklanan işlevlerin Server programdaki kısmıdır. İzleyici programla arasındaki özel mesajlar geldiğinde o mesaja göre gerekli işi yapar. Bu mesajlar dışında bir mesaj geldiğinde ise gelen mesajı üzerinde tanımlı olan izleyici bilgisayarların IP adreslerine yollar. Bu işlemleri yapan prosedürün kaynak kodu küçük açıklamalarla birlikte EK-2'dedir.

3.3 İzleyici Program

Tasarlanan bu sistemin en karmaşık ve görsel olarak kullanıcıya en fazla bilgi veren parçası İzleyici Programdır. İzleyici programın temel görevi gönderici programın gönderdiği ve Server programın da yönlendirdiği mesajları yorumlamaktır. İzleyici programın yazılım dağıtımına başlamadan önce kullanılabilecek ikinci bir işlevi daha vardır : Bilgisayarların Açık / Kapalı kontrolü. Bu işlev yazılım dağıtımının yapılacağı ve izleyici programda tanımlı olan bütün bilgisayarların açık ve bilgisayar ağında olup olmadığını yani merkezden o bilgisayarlara erişimin olup olmadığını kontrol eden kısımdır. Bu işlev ilgili konu başlığında detaylı anlatılmaktadır.

3.3.1 İzleyici Programın Temel Çalışma Prensipleri

İzleyici program çalıştırıldığında ekrana ilk önce herhangi bir server bilgisayara kayıt olunabilmesi için "Server'a kayıt olma" ekranı gelir. Burada kutucukta kayıt olunacak server'ın ayırd edilebilmesi için verilen kısa bir isim, ardından ":" karakteri ve server'ın IP adresi yazılı olarak gelir. Bu isim ve adres "ayarlar.ini" dosyasında belirtilmiştir. Bu dosyada alt alta birden fazla isim ve adres belirtilebilir. Bu durumda dosyadakiler liste olarak bu kutucukta alt alta gösterilir. Listedeki istenilen server seçilir. Baş tarafında kısaca isim belirtilmesi böyle bir liste olduğunda Server'ları isimlerinden kolaylıkla tanıyabilmek içindir. Dosyada en üstte yazılan server ismi ve adresi bu kutucukta görünen server ve adres olur.

Kayıt olma işleminin bir örneği Şekil 3.2’de görülmektedir. İlgili server listeden seçilerek yada elle yazılarak girildiğinde program, server’la bağlantı kurup kayıt olma isteğini bildirir. 5 saniye içinde olumlu cevap geldiği takdirde başarılı bir şekilde kayıt olunduğunu gösteren bir mesaj ekrana çıkar. Bu süre zarfında cevap gelmezse işlemin başarısız olduğunu belirten bir mesaj çıkar.

Server’a kayıt olma işlemi zorunlu değildir. Kayıt olunmazsa gönderici bilgisayarların direk olarak mesajı izleyici bilgisayarın IP adresine ve izleyici bilgisayarın dinlediği port’tan göndermeleri gerekir. Örneğin merkezde bir takım testler yapılırken bu kayıt olma işlemi kullanılmayabilir. Kayıt olmamak için program başlarken Şekil 3.2’deki ekran geldiğinde “iptal” butonuna tıklamak yeterlidir. Kayıt olunmadığı zaman Server bilgisayarın getirmiş olduğu kolaylık, esneklik ve veri depolaması özelliklerinden faydalanılmamış olur.



Şekil 3.2 Server’a Kayıt Olma Ekranı

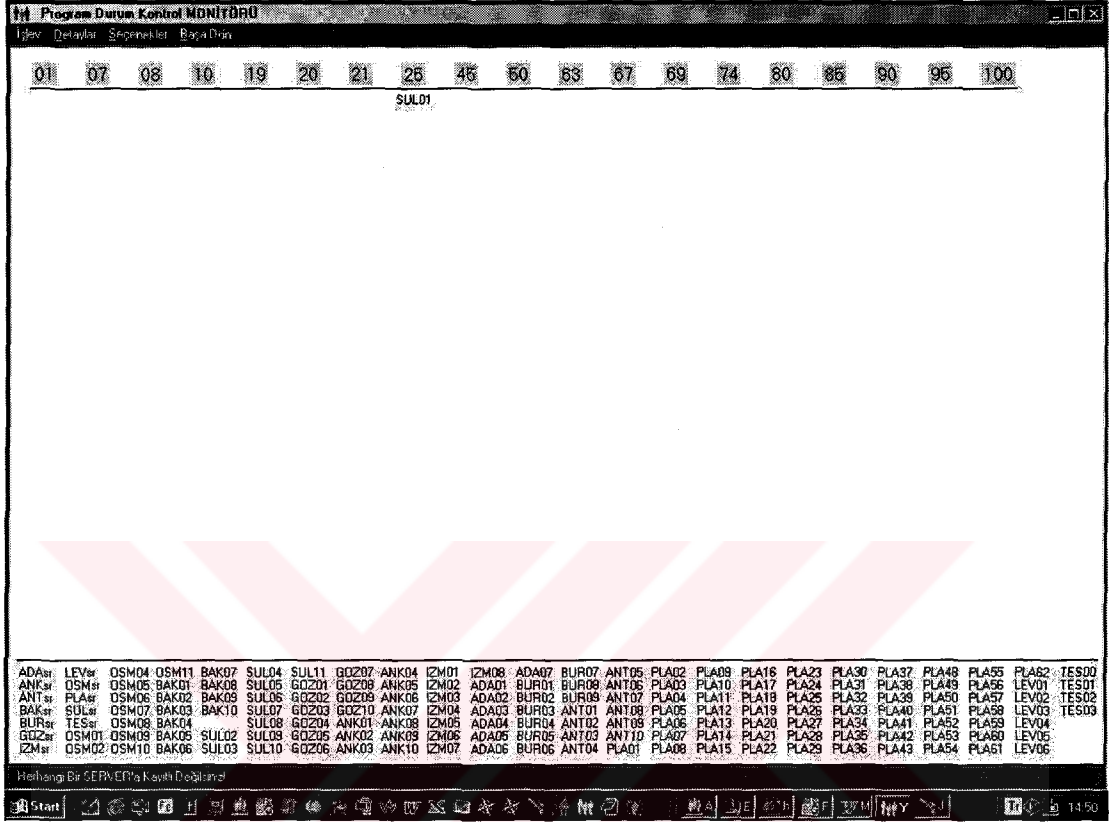
Kayıt olma işleminden sonra İzleyici program artık server'dan gelen mesajları dinlemektedir. Gelen mesaj iki kısımdan oluşur; gönderen bilgisayarın kullanıcı kodu ve takip edilen yazılım dağıtımın hangi aşamada olduğunu anlamamızı sağlayan durum mesajı. Gerek kullanıcı kodu, gerek durum mesajı sistemin sorumlusu tarafından tasarım aşamasında belirlenir. Bunlar sistem devreye girdikten sonra da istendiğinde değiştirilebilir. Önemli olan gönderilecek olan yeni mesajların İzleyici bilgisayar tarafından anlaşılabilmesidir.

İlk önce yapılması gereken gönderici bilgisayarın hangi aşamalarda ve nasıl bir mesaj göndereceğinin tespit edilmesidir. Bundan sonra kullanıcı kodları tespit edilir. Kullanıcı kodlarının bilgisayarın konumunu da belirten bir isim içermesi mesajı gönderen bilgisayarın tanınmasını kolaylaştırır. Mesela Sultanhamam'da bulunan bir bilgisayarın kullanıcı kodu 'SULT_001' olarak verilirse bu onun Sultanhamam'daki 1 nolu bilgisayar olduğunu gösterir. Kullanıcı kodlarının belirli bir standarda uyması işleri kolaylaştırır. Bu programda yukarıda verilen Sultanhamam örneğindeki gibi bulunulan yere göre kullanıcı kodu seçilmiştir. İlk dört karakter yeri gösterir. Arada bir ayıraç (alt tire / underscore karakteri) kullanılır ve sonra üç hanelik sıra numarası belirtilir.

Her gönderici bilgisayarın kendi kullanıcı kodunu bilmesi gerekir. Bunun sağlanması için ya kullanılan yazılım dağıtım ürününün kullanıcı kodu kullanılır ve bu kod yazılan bir programla alınır yada tespit edilen kullanıcı kodu bir dosyaya yazılır veya sistemin kullandığı bir genel değişkene atanır. Mesaj gönderileceği zaman kullanıcı kodu bir program aracılığıyla kaydedildiği yerden alınır ve parametre olarak gönderici programda kullanılır.

Kullanıcı kodu kadar önemli olan ikinci parametre ise gönderici bilgisayardaki yazılım dağıtım durumunu belirten mesajdır. Bu mesaj yazı (text) türünde herhangi bir şey olabilir. Bu programda gönderilen mesajın herkeze kolayca anlaşılabilmesi ve toplam sürecin neresinde olduğunun mesaja bir bakışta anlaşılabilmesi için 1'den 100'e kadar olan tam sayıların bazıları kullanılmıştır. Kullanılan bu sayılar ekranda en üstte yan yana belli aralıklarla yerleştirilir. Ekranda en altta ise sistemde kullanılacak olan bütün kullanıcı kodları kısaltılmış adları kullanılarak yerleştirilir.

Aşağıdaki kullanıcı kodlarından birine ait bir durum mesajı geldiğinde, izleyici program gönderici bilgisayarın kullanıcı kodunu bulur ve onu aşağıdan alıp gönderdiği mesaja göre yukarıdaki ilgili mesajın altına yerleştirir. (Şekil 3.3)

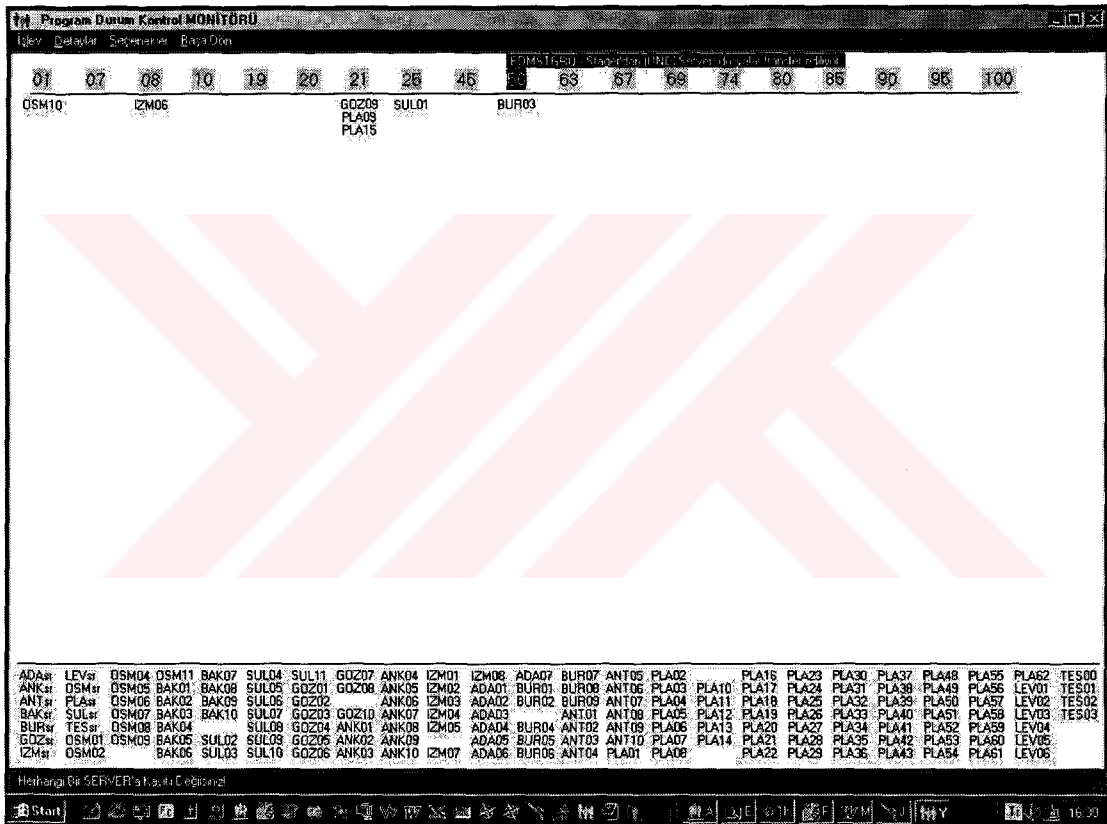


Şekil 3.3 Kullanıcı Kodlarının Mesaja Göre Yerleştirilmesi

Tasarlanan bu arayüzle hangi gönderici bilgisayarın yazılım dağıtımının hangi aşamasında olduğu bir bakışta kolayca görülebilir. Durum mesajı olarak tespit edilen bu sayılar yaklaşık olarak sürecin yüzdesi hakkında bilgi verir. Örneğin “01” mesajı yazılım dağıtımın yeni başladığını, “50” mesajı ortasında bir yerde olduğunu “100” mesajı ise işlemin bittiğini gösterir. En son durum mesajını da (bu programda “100” mesajı) gönderen bilgisayarın kullanıcı kodu değişik bir renkle gösterilmeye başlanır ve bu aşamaya gelen bilgisayar bir süre sonra ekranın aşağısındaki orjinal yerlerine geri dönerler. Yazılım dağıtımını bitirdikleri renklerinden ayırt edilir. Bu renkler “ayarlar.ini” dosyasından ayarlanabilir.

Aynı durum mesajını yaklaşık aynı zamanda bir çok bilgisayar gönderdiğinde bu bilgisayarların kullanıcı kodları aynı durum mesajı altında alt alta listelenir.

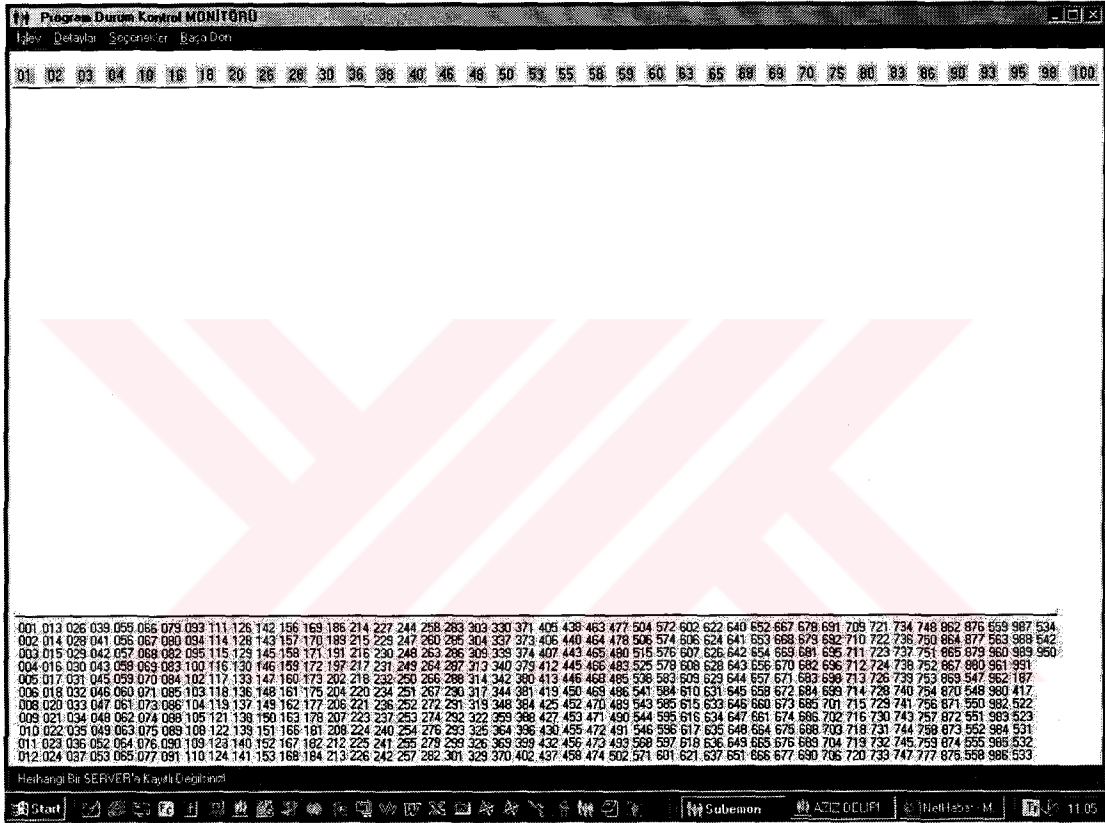
Şekil 3.4'te "21" mesajının altında üç tane farklı kullanıcı kodu görülmektedir. Bu durum mesajlarının açıklaması ise ana programda yapılır. Gönderilen her sayı gönderici bilgisayarda yapılan bir işlemin detayına işaret eder. Bu açıklamalar kullanıcı tarafından istendiği şekilde değiştirilebilir. Program içinden hangi mesajın ne anlama geldiği kolaylıkla öğrenilebilir. Bunun için ekranın yukarısında yan yana sıralanmış olan durum mesajlarından herhangi birinin üzerine fare ile tıklanır ve böylece o mesajın anlamı mesajların hemen üzerinde görüntülenir. Bununla ilgili bir örnek Şekil 3.4'te görülebilir. Burada "50" mesajının üzerine tıklanarak bu mesajın anlamı değişik renklerle yazılan yazıda görüntülenmiştir.



Şekil 3.4 Aynı Konumdaki Bilgisayarların Alt Alta Listelenmesi

Bu açıklamanın ekrandan kaybolması için açıklamanın üzerinde herhangi bir yere sol fare tuşu ile tıklanır. Mesajların üzerine fare ile gelindiğinde "Hint" olarak da açıklamalar geçici olarak görüntülenir ve bir müddet sonra yok olur. Şekil 3.4'te birkaç tane daha bilgisayarın konumları görülmektedir.

Mesajların niteliğinde sayıların seçilmesinin bir başka sebebi de bu tasarıma göre mesajlar yan yana dizildiğinden mümkün olduğu kadar az yer kaplanmasının sağlanmasıdır. Sayılar kullanılarak 35 kadar değişik mesaj yan yana görüntülenebilmektedir. Böylece gönderici bilgisayar 35 tane değişik aşama hakkında bize bilgi verebilir. Şekil 3.5'te 36 tane durum mesajı yan yana görülmektedir. Ayrıca 500 civarında kullanıcı kodu da ekranın alt tarafında listelenmiştir.



Şekil 3.5 Daha Fazla Bilgisayarın Durumunun Gösterilebilmesi

Görüntüde değişiklik yapmadan daha fazla aşamanın takip edilebilmesi de mümkündür. Bu konunun detayı ilgili konu başlığında anlatılmaktadır.

Gönderilecek mesajlar değiştirileceği zaman hem gönderici bilgisayarda değişiklik yapılması gerekmektedir hem de gönderilen mesajları anlayıp yorumlayabilmesi için izleyici programda değişiklik yapılması gerekir. İzleyici bilgisayarda tanımlı olan kullanıcı kodları, kullanılan mesajlar ve bu mesajların

açıklamaları text modunda kaydedilen “ayarlar.ini” dosyasında saklanır. Değişiklikler bu dosya üzerinde kolaylıkla yapılır. Bu konunun detayı ilgili konu başlığında anlatılmaktadır.

Gönderici programın kullandığı bu iki parametrenin (Kullanıcı kodu ve durum mesajı) dışında diğer bir önemli parametre de UDP mesajlaşmada kullanılacak port numarasıdır. Bu port numarası sistem sorumlusu tarafından belirlenir ve Server programın mesajları dinleme portu da ona göre ayarlanır. Server programın mesajları gönderme portu da İzleyici programın mesajları dinleme portu ile aynı numaraya ayarlanır. Tasarlanan bu sistemde gönderici bilgisayar “6666” nolu portu kullanmaktadır. Dolayısıyla Server program bu porttan dinlemektedir. Server program kendisine bu porttan gelen mesajları “6667” nolu porttan İzleyici Bilgisayarlara gönderir. İzleyici bilgisayarlar da sadece bu port numarasından gelen mesajları değerlendirirler.

İzleyici bilgisayarın Server’a kayıt olması, kaydını sildirmesi, yeni bir yazılım dağıtım başlatması, mevcut yazılım dağıtımını bitirmesi ve bir yazılım dağıtımının verilerini Server’dan getirmesi işlemlerinin hepsi “Seçenekler” menüsünün adımlarıdır. Bu işlevlerin çalışma prensibi “3.2 Server Program” kısmında alt alta maddeler şeklinde açıklanmıştır. Aynı kısımda Server programında ilgili kaynak kod vardır. Bu menü adımlarının her birinde Server’a gönderilen mesajlar şöyledir :

İzleyici bilgisayarın Server’a kayıt olması,

`SendUdpMes ('BANA DA YOLLA');`

İzleyici bilgisayarın Server’dan kaydını sildirmesi,

`SendUdpMes ('ARTIK BANA YOLLAMA');`

Yeni bir yazılım dağıtım başlatılması,

`Atxt:='NEW DIST : ' + StartNewDistEdit.Text; // Yazılım dağıtım Adı`

`SendUdpMes(Atxt);`

Mevcut yazılım dağıtımını bitirmesi,

Bu durumda önce Server'dan mevcut yazılım dağıtım ismi istenir. Ve kullanıcıdan bu dağıtımı bitirmek için onay alındıktan sonra

```
SendUdpMes('END DIST : '+copy(txt1,16,length(txt1)-15)); // Mevcut  
yazılım dağıtım adı
```

Bir yazılım dağıtımının verilerini Server'dan getirmesi,

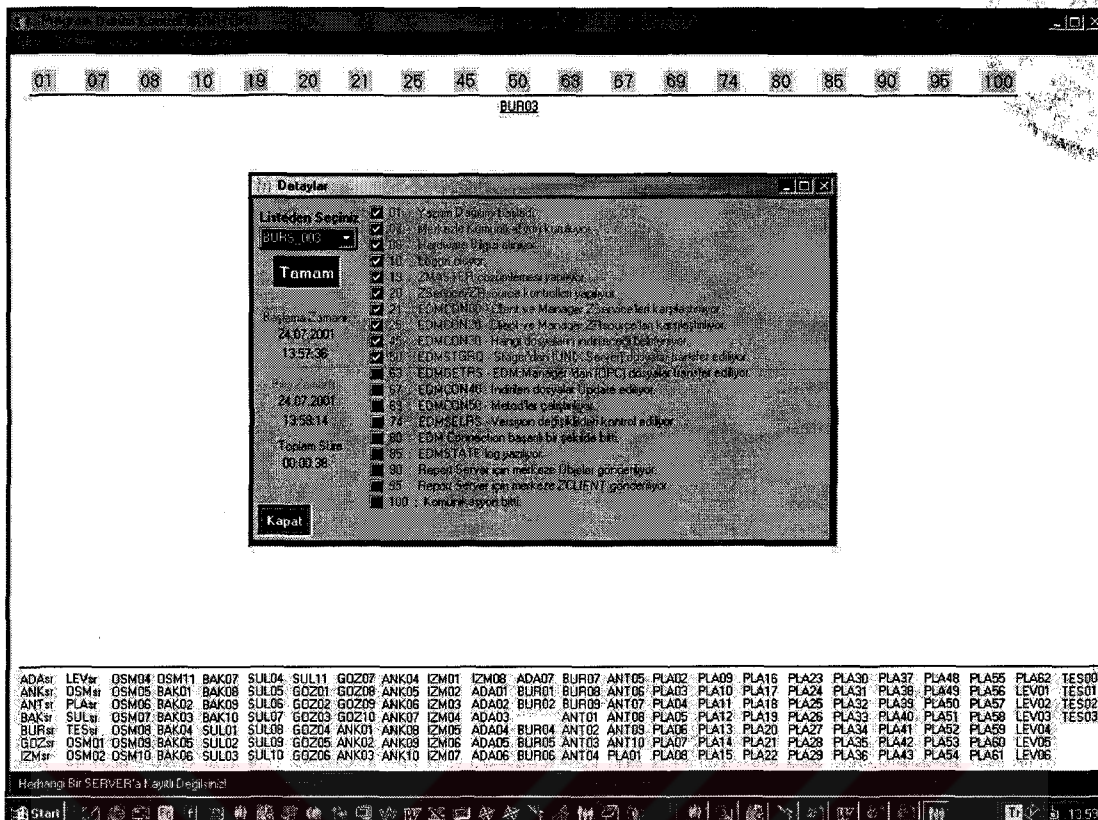
```
SendUdpMes('GET DIST LIST');
```

İzleyici programda Server'dan gelen mesajları değerlendiren kaynak kod kısa açıklamalarla EK-3'tedir.

3.3.2 Detay Bilgilerin Görüntülenmesi

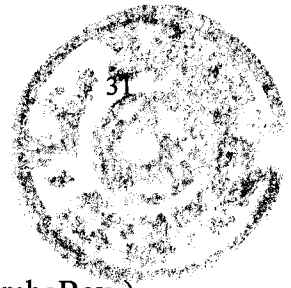
Gönderici bilgisayarlar her mesaj gönderdiğinde İzleyici program gelen mesaja göre gönderenin kullanıcı kodunu yukarıdaki mesajların altına yerleştirir. Böylece bir kullanıcı kodu "01" den başlayarak "100" e kadar olan mesajların altına aşama aşama gider ve böylece yazılım dağıtımının seyri canlı olarak izlenmiş olur. Bir kullanıcı kodu aynı zamanda sadece bir yerde olabileceğinden kullanıcı kodunun daha önce hangi aşamalardan geçtiği ekranda görülmez. Sadece son durumu ekranda görülebilir. Geçmiş aşamaları da görebilmek için kullanıcı kodunun üzerine farenin sol tuşu ile tıklanır. Karşımıza Şekil 3.6'daki gibi bir durum çıkar.

Şekil 3.6'da görüldüğü gibi "BUR03" kullanıcı kodlu bilgisayar "50" isimli aşamaya kadar gelmiş ve detaylar penceresinde o ana kadar geçmiş olduğu bütün aşamalar kutucuklara işaret konarak ve açıklamaları değişik renkle yazılarak gösterilmiştir. Herhangi bir zamanda da ekranın yukarısına çıkmış olsun veya olmasın bir kullanıcı kodunun üzerine farenin sol tuşuyla tıklandığında detaylar penceresinden program için hazırlanarak seçilmiş bütün aşamalar mesaj ismi ve açıklamalarıyla liste olarak görülebilir.



Şekil 3.6 Detayların Görüntülenmesi

Detaylar penceresinde görülen ve izleyici açısından çok önemli olan bir bilgi daha vardır : tarih ve saat bilgisi. Bu ekranda, seçilmiş olan kullanıcı kodunun ilk gönderdiği mesaj tarih ve saati ile son gönderdiği mesaj tarih ve saati; ayrıca bu ikisinin arasında geçen zaman görüntülenir. Böylece yazılım dağıtımının ne zaman başladığı, ne zaman bittiği ve ne kadar sürdüğü takip edilmiş olur. Yazılım dağıtım esnasında herhangi bir kullanıcı kodu uzun süre ekranda hareket etmediği zaman bunun belli bir aşamada takılı kaldığına kanaat getirilebilir. Sistem sorumlusu gönderici bilgisayarın belli bir işi yapmadan önce mi yoksa yaptıktan sonra mı mesaj göndereceğine tasarım aşamasında kendisi karar verir. Buna göre gönderilen mesaj ya o mesajın işaret ettiği işin başladığına yada bittiğine dair bilgi vermiş olur. Buradan yola çıkarak belli bir aşamada takılı kalan kullanıcı kodu eğer mesajlar işe başlarken gönderiliyorsa o işte takılı kalmıştır; eğer mesajlar iş bittikten sonra gönderiliyorsa o işi yapmış ve bir sonraki işi yaparken takılı kalmıştır. Bu kullanıcı kodunun üzerine farenin sol tuşuna tıklanarak detay penceresinden bitiş zamanına bakıldığında ne kadar süredir o aşamada olduğu anlaşılır. Çünkü bitiş zamanı en son



gönderilen mesajın zamanıdır.

Detaylar penceresinin sol üst köşesinde görünen seçim kutusu (ComboBox) içinde o ana kadar durum mesajı göndermiş olan bütün bilgisayarların kullanıcı kodu listesi görünür. Buradan istenilen kullanıcı kodu seçilerek onun detaylarını görmek mümkündür.

3.3.3 Kullanıcı Durumlarının Dosyaya Kaydedilmesi

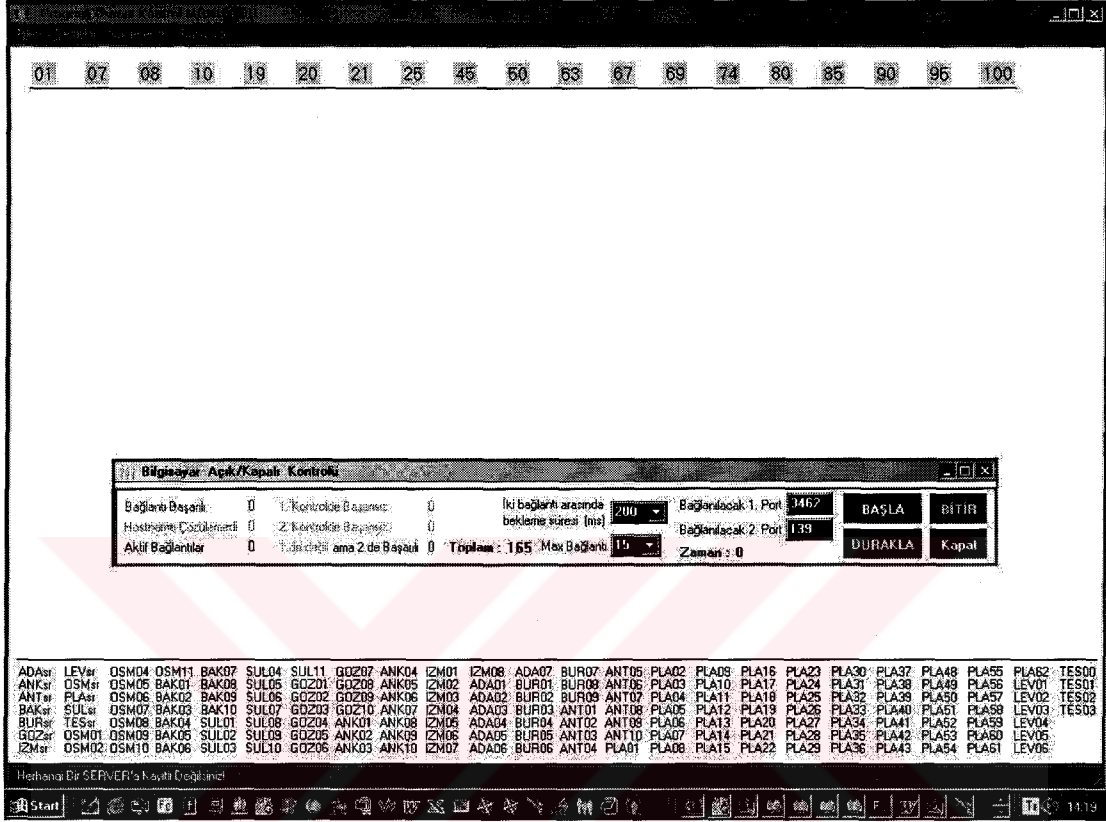
İzleyici bilgisayara gelen bütün durum mesajları günün tarihini “yıl-ay-gün” olarak içeren bir dosyaya kayıt edilir. Mesela 15.5.2001 tarihinde izleyici bilgisayara gelen durum mesajları “20010515.log“ isimli bir dosyaya yazı(text) modunda alt alta kaydedilir. Bu dosya sonradan incelemeye yönelik olarak tutulmaktadır. Ayrıca bilgisayara gelen durum mesajlarından sonra kullanıcı kodlarının ekranın yukarısında yerleştirildiği pozisyonlar, başlama ve bitiş zamanları gibi bilgiler istendiğinde “İşlev” menüsünden “Bilgisayarların Son Durumunu Kaydet” menü adımı seçilerek istenen bir isimle harddisk’e kaydedilebilir. Burada dikkat edilmesi gereken gelen mesajların değil sadece son durumların kaydedildiğidir.

Yazılım dağıtımının herhangi bir anında bilgisayarların durumu ekranın o anki durumunu fotoğraf çeker gibi saklamak için bu işlev yardımıyla kaydedilebilir ve istenildiği zaman yine “İşlev” menüsünden “Bilgisayarların Son Durumunu Yükle” menü adımı seçilerek istenen dosya diskten seçilir ve son durum karşımıza hızlı bir şekilde gelir. Bu işlem Server’den Yazılım dağıtım bilgilerini getirmekten çok daha hızlıdır çünkü kullanıcı kodu başına bir işlem yapılır. Diğerinde ise bütün mesajlar teker teker değerlendirilerek gerçek yazılım dağıtım aşamalarının hızlı bir tekrarı gerçekleştirilir.

3.3.4 Bilgisayarların Açık / Kapalı Kontrolü

İzleyici programın ikinci bir işlevi de yazılım dağıtımından önce yada dağıtım esnasında bir bilgisayardan mesaj gönderimi kesildiğinde kullanılabilecek olan bilgisayarların Açık / Kapalı (Bilgisayar ağında ulaşılabilir durumda yada değil)

kontrolünü yapan kısımdır. Bu işlevi aktif hale getirmek için “İşlev” menüsünden “Bilgisayarların Açık / Kapalı Kontrolü” menü adımı seçilir. Karşımıza Şekil 3.7’deki gibi bir ekran gelir.



Şekil 3.7 Bilgisayarların Açık / Kapalı Kontrolünün Yapılması

Burada program kendisine tanımlanmış olan kullanıcı kodlarına ait olan IP adresi yada Hostname'lerine, "ayarlar.ini" dosyasında belirtilmiş olan yada dinamik olarak programdan değiştirilebilen TCP port numarasından TCP olarak bağlantı kurmaya çalışır. Bağlantının başarılı olması bilgisayarın açık ve bilgisayar ağından ulaşılabilir olduğunu garanti eder. Bu durumda o kullanıcı kodu ayarlanan renge boyanır.(standart olarak açık yeşil renk) Eğer bağlantı başarısız ise bu o port numarasından o bilgisayara ulaşamadığı anlamına gelir. Bu kullanıcı kodları da kırmızı renge boyanır. Kontrol sonunda başarısız olanlara tanımlanmış olan başka bir port numarasından ikinci defa bağlantı denenir. Birincide başarısız olup ikincide başarılı olanlar sarı renge boyanır. İkincide de başarısız olanlar kırmızı olarak kalır. İki kullanıcı koduna bağlantı kurulma aşamasında belli bir süre beklenir. Bu süre Şeki 3.7'de görülen "iki bağlantı arası bekleme süresi" dir ve ayarlanabilir. Aynı

anda aynı port numarası kullanılarak birçok bilgisayara bağlantı kurulduğundan maksimum bağlantı sayısı sınırlandırılmıştır. Belli bir sayıdan sonra işletim sistemi hata alabilmektedir. Bu sayı işletim sistemine ve sistemdeki yüke göre değişebilir.

İşlemi başlatmak için ilk önce eğer değiştirilecekse port numaraları değiştirilir. Bu değişiklikler kalıcı olarsa “ayarlar.ini” dosyasına yazılmalıdır. “Başla” butonuna tıklandığında program iki bağlantı arasında belirtilen süre kadar beklemek koşuluyla ard arda kullanıcı kodlarının IP adreslerine bağlantı kurmaya başlar. Bağlantının durumuna göre kullanıcı kodlarının renkleri değişir. Ekranda başarılı bağlantılar, başarısız bağlantılar, aktif bağlantılar(cevap beklenenler) ve Hostname’i çözilemeyenlerin sayısı dinamik olarak gösterilir. İşlem bitince bir mesajla ekrana iletilir.

Bu kontrol bize yazılım dağıtımı öncesinde kendisi kapalı olan yada yazılım dağıtımının kullandığı TCP portu kapalı olan bilgisayarları tespit etmemize ve gerekiyorsa bunlara müdahale etmemize olanak sağlar. Mesela bu programda kullanılan 3462 numaralı port EDM yazılım dağıtımı ürünü tarafından yazılım dağıtımını başlatma mesajı göndermek için kullanılır. Bu porttan iletişim sağlanamayan bilgisayarlar bu mesajı alamayacak, dolayısıyla yazılım dağıtımı başlamayacak demektir. Bu bilgisayarlara uzaktan erişim ile bağlanılıp gerekli müdahale yapılabilir. Kendisi kapalı olan bilgisayarların ağ kartları “Wake on LAN” (ağ iletişimiyle uyanma) özelliğini destekliyorsa bu yöntemle uzaktan açılabilir ve yazılım dağıtımını alması sağlanabilir.

3.3.5 “ayarlar.ini” Dosyasının Yapısı ve Değişiklikler

İzleyici programda başta kullanıcı kodu ve durum mesajı tanımları olmak üzere bir çok şey ayarlanabilir. Bu ayarlamalar programın kendisiyle aynı dizinde bulunan “ayarlar.ini” dosyasında yapılır. Örnek bir ayarlar.ini dosyası EK-4 te bulunabilir.

Örnek dosyada görülebileceği gibi [NETWORK] başlığı altında Server’la iletişim için kullanılacak port numaraları; Bilgisayarların Açık/Kapalı kontrolü sırasındaki maksimum bağlantı sayısı, iki bağlantı arasında beklenecek zamanı ve hangi port numaralarından TCP bağlantı deneneceği yazılır.

[REGISTER] başlığı altında kayıt olunabilecek server'ların ismi ve "..." işaretinden sonra IP adresi yazılır. Birden fazla server varsa herbiri alt alta yazılır.

[STATUSCOORD] başlığı altında ekranın yukarısında çıkan durum mesajlarının yazı tipi büyüklüğü, ilk durum mesajının ekranın sol tarafından ve üstünden piksel olarak uzaklıkları, bir sonraki durum mesajının kaç piksel sağa yerleştirileceği, alt alta gelen kullanıcı kodları listelenirken satır yüksekliği ve programa gelecek durum mesajının maksimum uzunluğu yazılır.

[USERIDCOORD] başlığı altında, kullanıcının kendisinin yazılım dağıtım başlattığının takip edilip edilmeyeceği "self_connected_user=" satırı ile belirtilir. Şöyle ki bu satırın sonunda "TRUE" olursa baştan ikinci mesajı gönderen kullanıcıların yazılım dağıtımını kendilerinin başlattıkları anlaşılır. Bu durumda ikinci mesajı gönderen kullanıcı kodlarının altı çizili hale gelir. Bu özelliklerin sağlanması için sistem sorumlusu, yazılım dağıtım merkezden otomatik başlatıldığında birinci mesajı göndermeli, kullanıcı kendisi başlattığında ikinci mesajı göndermelidir. Bunu ayırt etmek için biraz kod yazmak gerekebilir. Bu özellik opsiyoneldir. "self_connected_user=FALSE" yazılırsa bu özellik devre dışı kalır ve ikinci mesajı gönderenler herhangi bir şekilde ayırt edilmezler. Bundan sonraki satırda kullanıcı kodunun ve IP adresinin karakter olarak uzunluğu, programda kullanıcı kodunun hangi karakterlerinin kullanılacağı (programda gerçek kullanıcı kodunun kısaltmasını kullanmak yer avantajı sağlıyor. Mesela BAKR_009 yerine BAK09 kullanılabilir. Bu da ilk üç karakter ile son iki karakterin kullanılması demektir.) Bundan sonraki beş satırda ekranın alt tarafında liste halinde yerleştirilen kullanıcı kodlarının ekranın sol tarafından ve üstünden piksel olarak uzaklıkları, kullanıcı kodlarının toplam kaç satırda listeleneceği, bir sonraki kullanıcı kodunun kaç piksel sağa yerleştirileceği, alt alta gelen kullanıcı kodları listelenirken satır yüksekliği yazılır.

[DETAILSCOORD] başlığı altında ise detayları gösteren pencerenin ekranın sol tarafından ve üstünden piksel olarak uzaklıkları, pencerenin genişliği ve yüksekliği, pencere içinde listelenen durum mesajlarının ilkinin pencerenin sol tarafından ve üstünden piksel olarak uzaklıkları ve alt alta listelenen durum mesaj açıklamalarının satır yüksekliği yazılır.

[COLORS] başlığı altında ekranda kullanılan sırasıyla şu renklerin değerleri verilebilir :

Detay penceresindeki işaretlenmemiş olan durum mesajlarının rengi

Detay penceresindeki işaretlenmiş olan durum mesajlarının rengi

Ekranın üst tarafındaki durum mesajlarının rengi

Ekranın üst tarafındaki durum mesajlarının arka plan rengi

Ekranın alt tarafındaki kullanıcı kodlarının ilk rengi

Yazılım dağıtımını bitiren kullanıcı kodlarının rengi

Bilgisayar Açık/Kapalı kontrolü başlarken kullanıcı kodlarının rengi

Hostname'i çözilemeyen kullanıcı kodlarının rengi

Bağlantısı aktif olan fakat henüz sonuç alınamayan kullanıcı kodlarının rengi

Bağlantısı başarılı olan kullanıcı kodlarının rengi

Bağlantısı başarısız olan kullanıcı kodlarının rengi

Bağlantısı 1. Kontrolde başarısız fakat 2. de başarılı olan kullanıcı kodlarının rengi

Delphi'de tanımlı olan standart renkler ise şunlardır :

ClWhite : Beyaz

ClGray : Koyu gri

ClSilver : Gümüş gri

ClYellow : Sarı

ClRed : Kırmızı

ClMaroon : Koyu kırmızı

ClBlue : Mavi

ClAqua : Açık mavi

ClNavy : Koyu mavi

ClLime : Açık yeşil

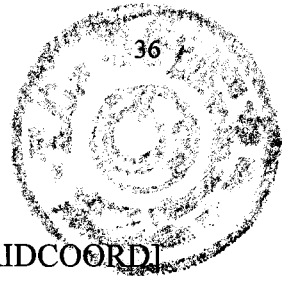
ClGreen : Koyu yeşil

ClOlive : Zeytin yeşili

ClTeal : Petrol yeşili

ClFuchsia : Pembe

ClPurple : Mor



CIBlack : Siyah

[USERID] başlığı altında ise önce maksimum uzunlukları [USERIDCOORD] kısmında belirtilen kullanıcı kodu ve kullanıcı IP adresleri, arada bir boşluk karakteri bırakılarak yan yana yazılır. Sonra istenirse kullanıcı hakkında bir açıklama yine bir boşluk bırakıldıktan sonra yazılır. Bu şekilde bütün kullanıcılar alt alta listelenir.

[IND] başlığı altında maksimum uzunluğu [STATUSCOORD] kısmında belirtilen durum mesajı ve “ : “ yazısı araya konarak durum mesajı ile ilgili açıklama yazılır. Bu açıklama programda detaylar penceresinde görülen açıklamalardır. Bu şekilde bütün durum mesajları alt alta listelenir.

3.3.6 Daha Fazla Aşamanın Takip Edilebilmesi

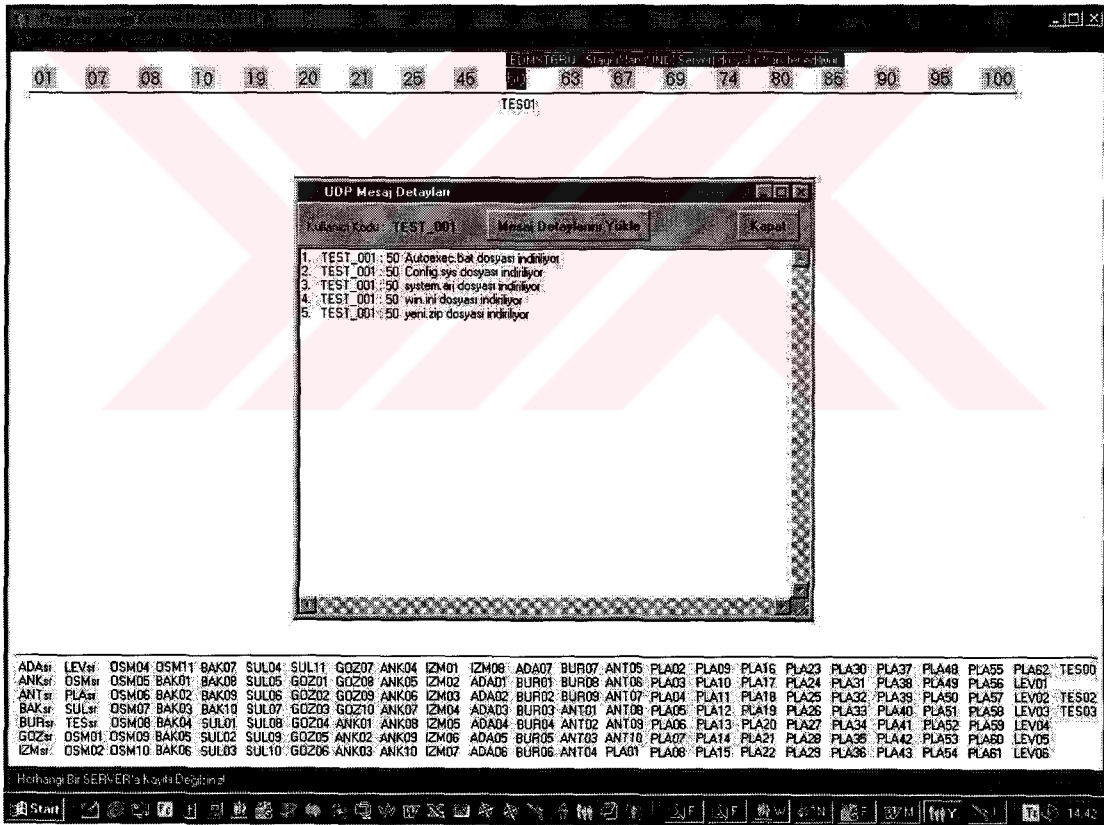
Daha önce “3.3.1 İzleyici Programın Temel Çalışma Prensipleri” kısmında İzleyici programda durum mesajları ekranın üst kısmında yan yana dizildiği için 35 civarında bir mesajdan fazlasının ekrana sığmasının mümkün olmadığından bahsedilmişti. Ekranda durum olarak daha fazla mesaj gösterilemese de ekranda listelenmiş olan durum mesajları “ana durum mesajı “ olarak kabul edilip bunlarla alakalı olarak alt durum mesajları gönderilebilir.

Mesela “50 “ mesajı “merkezden dosyalar indiriliyor” şeklinde açıklaması olan bir mesaj olsun ve merkezden 10 tane dosya indirilecek olsun. “TEST_001” kullanıcı kodlu bilgisayar dosyalar indirilmeye başlanırken “TEST_001 : 50” şeklinde bir mesaj gönderir. Durum mesajının maksimum uzunluğu üç karakter ise burada izleyici program anlamlı olan “50“ mesajını alır ve bu mesaja göre kullanıcı kodunu ekranın üst tarafında ilgili durum mesajının altına yerleştirir. Eğer kullanıcı birinci dosya inerken “TEST_001 : 50 Autoexec.bat dosyası indiriliyor” ve ikinci dosya inerken “TEST_001 : 50 Config.sys dosyası indiriliyor” şeklinde mesajlar yollarsa anlamlı durum mesajı yine “50” dir (çünkü maksimum durum mesaj uzunluğu 3’tür) ve kullanıcı kodunun ekrandaki konumu değişmez, herhangi bir hareket olmaz. Fakat izleyici program bu gelen mesajları her bir kullanıcı kodu için ayrı ayrı hafızada tutarsa ve istendiğinde bu mesajları alt alta gösterirse kullanıcı

37

kodu hakkında daha detaylı bir bilgiye sahip olmuş oluruz. Kullanıcı kodu “50” durum mesajının altında belki uzun süre duracaktır ve bu mesajları göndermezsek uzun süre bu kullanıcı ile ilgili bilgi alamayacaktık. Kullanıcı 10 tane dosya indiriyor ve merkezden o anda hangi dosyayı indirmekte olduğu görülemiyor olacaktı. Sadece dosya indirme aşamasında olduğu bilinecekti.

Bunun gibi bütün durum mesajlarının temsil ettiği aşamalar alt aşamalara bölünerek daha detaylı bilgi ve açıklama içeren mesajlar gönderilebilir. Gönderilen bu mesajlar her kullanıcı için ayrı ayrı dizilerde toplanır ve İşlev menüsünden “Bilgisayarların son durumunu kaydet” menü adımı seçildiğinde, diske bu bilgiler de kaydedilir. Bu mesajları görüntülemek için kullanıcı kodunun üzerine farenin sağ tuşu ile tıklanır. Karşımıza Şekil 3.8’deki gibi bir görüntü çıkar.



Şekil 3.8 Belli Bir Bilgisayarın Mesajlarının Görüntülenmesi

Şekil 3.8’de görüldüğü gibi “TEST_001” kodlu kullanıcı, açıklaması kısaca “dosyalar transfer ediliyor” olan “50” durum mesajına değişik açıklamalar ekleyerek

her seferinde göndermektedir. Bu açıklamaların her biri bize bilgisayarın durumu hakkında yeni bilgiler vermektedir. Bu özellik kullanılarak ekranın yukarıdaki durum mesajları ve bunların altında gelecek detay mesajlar daha iyi belirlenebilir.

Bu mesajlar üzerinde tasarım aşamasında iyi çalışılıp güzel bir sistem kurulduğunda ekrana sığdırabilen 35 tane ana durum mesajının her biriyle alakalı ortalama 10 tane de alt detay mesajı tespit edilirse, bu durumda toplam 350 tane ayrıntılı aşama hakkında merkeze bilgi yollanabilir. Toplam 350 adet aşama ile bir yazılım dağıtım yada benzer bir sistem hakkında oldukça detaylı bilgi sahibi olunabilir.





4. ÜRETİM ORTAMINDAKİ SONUÇLAR

Tasarlanan bu sistem üretim ortamında Yapı Kredi Bankası'nın Türkiye'nin İstanbul, Ankara, İzmir, Adana, Bursa, Antalya gibi büyük illerindeki şubelerinde konumlanmış olan 165 adet gönderici bilgisayarına başarıyla adapte edilmiştir. Yurt çapına dağılmış toplam 12 şube, her şubede 1 tane HP 9000 serisi Unix sistemi ve buna bağlı olarak çalışan 7 ila 50 adet arasında Microsoft Windows 98 işletim sistemi yüklü iş istasyonu vardır. Bu iş istasyonlarında ve Unix Server'larda Novadigm firmasının EDM yazılım dağıtım ürünü kurulu bulunmaktadır. Yazılım dağıtım kapsamında bütün iş istasyonlarına gönderilecek dosyalar aynıdır. Uzaktaki bu bölgelerle olan iletişimin hızı sınırlı olduğundan iş istasyonlarına gönderilecek dosyalar ilk önce bir defaya mahsus olmak üzere Unix server'lara gönderilir. İş istasyonları ile Server'lar aynı yerel alan ağında (LAN) olduğu için Server ile bağlantıları oldukça hızlıdır. Server'lara giden bu dosyaları iş istasyonları LAN'dan oldukça hızlı bir şekilde kopyalayabilirler.

Tasarımın yapısı gereği Server'ların bu dosyaları merkezden transfer etmeleri oldukça elzemdir. Çünkü server dosyaları transfer edemezse ya da arada takılıp kalırsa o server'a bağlı olan iş istasyonları dosyaları alamamış olur. Dolayısıyla iş istasyonlarının yanında server'ların da durumunu takip etmemiz gereklidir.

İş istasyonlarında bu dökümanda açıklanan Windows 95/98 tabanlı gönderici program kullanılmış, Unix Server'larda ise "perl" diliyle yazılmış gönderici program kullanılmıştır. Bunun için Unix Server'lara "perl" kurulmuştur. EDM yazılımı bir çok ".exe" dosyasından oluşur. Bu ".exe" programları, "Rex" rutinleri ile düzenlenip çağrıldığından ve bu rex dosyaları kodları açıkta olup yorumlayıcı (interpreter) tarafından çalıştırıldığından bu "rex" dosyalarının içinde kolaylıkla değişiklik yapıp gönderici program gerekli parametrelerle çağrılabilir. Üretim ortamında sistem çalışırken alınmış bir görüntü Şekil 4.1'de gösterilmiştir.

Program Durum Kontrol MONİTORU																		
İzley Detaylar Seçenekler Başla Dön																		
01	07	08	10	19	20	21	25	45	50	63	67	69	74	80	85	90	95	100
PLA12	PLA49	ANT07	I2M02	ANT06	ANT01	ANK04	ADA02	ANK07	SUL04	GOZ01	BAK10	OSM05	OSM09	BAK05	BAK08	BAK07	SUL01	
PLA33		PLA15	BUR01	ANT02	ANT04	GOZ05	GOZ05	SUL09	OSM11	BAK04	BAK03	SUL10	OSM02	TES01	BAK02	BAK02	OSM06	
PLA25		PLA07	BUR06	PLA01	ANK08	BUR04	I2M05	I2M06	SUL08	BAK09	SUL05	SUL02	SUL03				BUR01	
PLA28		PLA30	ADA03	BUR05	GOZ03	ADA07	ANK09	ANK05	SUL11	SUL06	BAK06	BAK06	BAK06					
PLA41		PLA06	PLA21	ANT03	ANK10	BUR03	I2M01	GOZ06	GOZ10	ANK03	SUL07							
PLA55		BUR06	ADA01	PLA03	ANK10	BUR03	I2M01	GOZ06	GOZ10	ANK03	SUL07							
PLA37		PLA40	PLA17	ADA05	PLA09	BUR02	ANK02	GOZ07	GOZ07									
PLA80		PLA43	PLA27	PLA26	ANT03	GOZ02	ANK06	PLA13	I2M04									
PLA56		PLA11	PLA08	ANK01	I2M08	SUL01			GOZ08									
PLA89		PLA19	PLA36	PLA05	ANT10	PLA02												
PLA36		PLA57	PLA22	I2M07														
LEV04		PLA10	ADA05	PLA16														
PLA51		PLA39	PLA31															
PLA54		PLA24	PLA04															
PLA61		ADA04	PLA34															
LEV06		PLA35	PLA18															
PLA32		PLA42	PLA29															
		PLA14																
		LEV03																
		PLA52																
ADA01	LEV01												PLA23			PLA48	PLA62	TES00
ANK01	OSM01															PLA50	LEV01	TES01
ANT01	PLA01																LEV02	TES02
BAK01																		TES03
	OSM02												PLA20				PLA59	
GOZ01																PLA53		
I2M01																	LEV05	
Herhangi Bir SERVER'a Layış Değilmi																		
Start																		

Şekil 4.1 Üretim Ortamından Bir Görüntü

Bu şekilde ekranın yukarısında görülen kullanıcı kodları yazılım dağıtımının belli aşamalarındadır. Aşağıdakilerden sol taraftakiler (orjinal renkli resimde yeşil renkte görülenler) yazılım dağıtım sürecini bitirmiş olanlardır. Aşağıdakilerden sağdakiler ise (orjinal renkli resimde sarı renkte görülenler) henüz yazılım dağıtım sürecine başlamamış olanlar yani hiç durum mesajı göndermemiş olanlardır.

Yine benzer yapı yurtiçinde hemen her yerde bulunan Yapı Kredi Bankası'nın yaklaşık 500 adet şubesindeki IBM OS/2 işletim sistemi yüklü server'lara ve bu server'lara bağlı toplam 8000 adet iş istasyonuna da adapte edilmiş ve sorunsuz şekilde çalışmaktadır. Bu sistemde hem server'da hem de iş istasyonlarında gönderici program "rex" diliyle yazılmıştır.

Üretim ortamında bu sistemin içerisinde yer alan üç programda da belirgin herhangi bir sorun yaşanmamıştır. İzleyici programın çalıştırılacağı bilgisayarın işlemci olarak oldukça güçlü olması özellikle çok gönderici bilgisayarın tanımlı

olduğu ortamlarda aranması gereken bir özelliktir. Çünkü her bir mesaj geldiğinde mesajın kimden geldiğini anlamak için sistemdeki bütün kullanıcı kodları teker teker mesajı gönderenle karşılaştırılmaktadır. Ayrıca gelen durum mesajının hangisi olduğunun anlaşılabilmesi için, yine sistemdeki bütün durum mesajları ile teker teker karşılaştırılmaktadır. Saniyede onlarca mesaj gelebileceğini hesaplarsak bu işlemin her bir mesaj için gerçekleştirileceği düşünüldüğünde işlemci hızının önemi daha iyi anlaşılır.

Burada bahsedilenlerin dışında sistemde herhangi bir sorunla karşılaşılmamıştır.



5. SONUÇ

Tasarlanan bu izleme sistemiyle kendine bağılı şubeleri bulunan kuruluşların aynı bilgisayar ağında bulunan bilgisayarlarında çalışan yazılım dağıtım süreci canlı olarak izlenebilmektedir. Yazılım dağıtımının canlı olarak izlenmesi ve oluşan problemlerin, takılmaların yada bazı aşamaların atlanmasının anında görülebilmesi ve gerekli tedbirlerin alınması bilgisayar sistemlerinin düzgün çalışması için hayati önem taşır. Bu işlemler yüzlerce bilgisayarı etkilediğinden, ekranda kolay ve kullanışlı bir kullanıcı arayüz tasarımının yapılması tersliklerin hızlı bir şekilde görsel olarak tespit edilmesini sağlar.

Problem teşhisi dışında yazılım dağıtım sürecinin bitiş zamanı tahmin edilebilir. Ayrıca hangi bilgisayarın ne kadar zamanda bu süreci tamamladığını yada bir adımda ne kadar süre takılı kaldığını tespit etmek mümkün olmaktadır.

Zamanımız verimlilik zamanıdır. Böyle bir zamanda büyük bir kuruluşun verimliliğinin ve sistem kaynaklarının kullanılabilirlik yüzdesinin bir kaç puan artması o kuruluşa çok büyük avantaj sağlar .

Ayrıca sistemde kullanılan mesajların yapısı ve içeriği herhangi bir sınırlamaya tabi değildir. Bu sistemde gerekli konfigürasyon yapıldıktan sonra belli aşamaları olan her türlü program, işlem, yedekleme sistemleri, veri tabanları, ürün kuruluşları, veri transferleri vs. izlenebilir. Bunun için bu tür sistemlerde kritik aşamalar tespit edilir ve bu aşamalarda mesajların gönderilmesi sağlanır.

Günümüzde veritabanı bütün sektörlerde yaygın olarak kullanılır. Bazı kuruluşların veri tabanındaki tablolar o kadar büyüktür ki bunları yedeklemek saatler sürer. Onlarca büyük tablonun her biri teker teker teyp kartuşlara aktarılır. Mesela böyle bir sisteme tasarlanan bu mesajlaşma sistemi entegre edilebilir. Yedeklemeyi yapan ve kontrol eden prosedürün içinde önceden tespit edilen aşamalarda merkeze uygun mesajlar gönderilir. Merkezdeki kullanıcı böylece yedekleme işlemini canlı olarak izleyebilir ve yedekleme işleminin aşamalarının ne kadar sürdüğünü tespit ve kayıt edebilir.

6. YAPILABİLECEK EK ÇALIŞMALAR

İzleyici program kendisinde tanımlı olan kullanıcı kodlarının gönderdiği mesajları alıp değerlendirebilir. Bunun için programın kullanacağı her bağımsız ortam için programın “ayarlar.ini” dosyası değiştirilmeli, kullanıcı kodunun uzunluğuna, formatına ve toplam kullanıcı kod sayısına göre ekran koordinatları ve bütün ayarlar yapılmalıdır. Kendisine tanımlanmamış bir kullanıcı kodundan mesaj geldiğinde program o mesajı kaybeder. Çünkü mesajlar ancak kullanıcı kodlarına göre hazırlanmış olan dizilerde tutulur ve diske kaydedilir. Ekranda bu tip mesajları görüntüleyecek bir ortam da tasarlanmamıştır. Gerçekte ortamda var olan fakat sisteme tanımlanması unutulmuş kullanıcı kodları olduğunda da durum aynıdır.

Mesajlar kaybedilmiş olur.

Böyle bilinmeyen kullanıcı kodlarından gelen mesajlar için eğer gelen durum mesajı tanımlı olanlardan birine uyuyorsa programda dinamik olarak objeler oluşturulup diğerlerinden farklı renklerde olmak üzere bu objeler ilgili durum mesajlarının altına yerleştirilebilir. Ya da farklı bir pencerede bu tip mesajlar istendiğinde gösterilebilir.

Eklenebilecek başka bir özellik de yazılım dağıtımı esnasında özellikle belli bir aşamada takılı kalan bilgisayarlara erişim ve müdahale kolaylığı sağlanmasıdır. Kullanıcı kodlarının temsil ettiği bilgisayarların IP adresleri bilindiğinden bunlara “Telnet” ve/veya “ftp” seanslarının otomatik olarak açılması sağlanabilir. Bu bilgisayarlardan merkezde incelenmek üzere bazı dosyaların otomatik olarak ftp kullanılarak merkeze transfer edilmesi sağlanılabilir. Tabi bunun için bu bilgisayarlarda telnet ve ftp için gerekli konfigürasyon yapılmalıdır.

Uzaktaki bu bilgisayarlarda sadece mesaj göndermeye yarayan gönderici program geliştirilerek merkeze o bilgisayarla ilgili bir takım teknik bilgileri de gönderecek hale getirilebilir. Mesela disk büyüklüğü, hafıza miktarı, mikro-işlemci tipi ve hızı, boş disk alanı, ekran kartı, varsa bilgisayarın ve monitörün marka ve modeli, sisteme bağlı bulunan yazıcı vs. gibi bilgiler ilk durum mesajından önce her

yazılım dağıtımında bir kereye mahsus olmak üzere özel tipte mesajlar şeklinde merkeze gönderilebilir. Merkezdeki izleyici programda da bu bilgileri değerlendirecek ve iyi bir arayüzle kullanıcıya sunup raporlayacak bir takım geliştirmeler yapılabilir. Bu işlev bir nevi envanter takibi olacaktır.

Gönderici program daha da geliştirilerek merkezden gönderilecek bir takım işletim sistemi komutlarını o bilgisayarda çalıştırabilir. Mesela yazılım dağıtım takılı kaldığında yazılım dağıtım programının ürettiği “log” dosyalarını vs. silip yazılım dağıtım tekrar başlatılabilir yada gerektiğinde işletim sistemi yeniden yüklettirilebilir.



KAYNAKLAR



- 1- ACAR, E., EDM Design Objectives, Ümit Bilgisayar, İstanbul, 1998
- 2- BARGHOUTI, N.S. and KRISCHNAMURTHY, B., Using Event Contexts and Matching Constraints to Monitor Software Processes, Software Engineering Research Department AT&T Bell Laboratories Murray Hill, NJ 07974 USA, 1995
- 3- DELİPINAR, A.B., Üretim Ortamında Yaşanan Yazılım Dağıtım Problemleri, Yapı Kredi, Gebze, 2000.
- 4- GRANT, A.R. and HIGGINS, C.A., Computerized Performance Monitors as Multidimensional Systems: Derivation and Application, University of Victoria and University of Western Ontario, 1996
- 5- JOYCE, J. et al, Monitoring Distributed Systems, ACM Trans. on Computer Systems; 5 (2); pp. 121- 150; May 1987. Dieter Wybranietz, Monitoring and Performance Measuring Distributed Systems During operation, 1988
- 6- KAMINKA, G.A. et al, Monitoring Deployed Agent Teams, Computer Science Dept. Carnegie Mellon University Pittsburgh, 2001
- 7- SAS, R., EDM Administrator's Guide, Novadigm, 1997
- 8- TOKUDA, H. et al , A Real-Time Monitor for a Distributed Real-Time Operating System, Computer Science Department Carnegie Mellon University Pittsburgh, Pennsylvania, 1988
- 9- WYBRANIETZ, D. and HABAN, D., Monitoring and Performance Measuring Distributed Systems During operation, University of Kaiserslautern, 1988
- 10- ZAMPROGNO, M., EDM Branch Design, Yapı Kredi, Gebze, 1998

ÖZGEÇMİŞ

Kişisel Bilgiler	• Adı Soyadı : Aziz Barış DELİPİNAR • Doğum Tarihi : 16.05.1972 • Adres : Yapı Kredi Bankacılık Üssü, Inventüs 2 Gebze / KOCAELİ • Telefon : 0 262 647 31 09
Eğitim	• 1983 - 1990 Samsun Anadolu Lisesi / SAMSUN • 1990 - 1995 Ortadoğu Teknik Üniversitesi, Elektrik-Elektronik Müh. Bölümü ANKARA • 1997 - Gebze Yüksek Teknoloji Enstitüsü, Bilgisayar Müh. Bölümü Gebze / KOCAELİ
İş Tecrübesi	• Bilgi İşlem Sorumlusu Ağustos 1995 – Nisan 1996 Alemdar Holding, Kaynarca / İSTANBUL • Analist Programcı Nisan 1996 – Ekim 1996 ITD A.Ş., Levent / İSTANBUL Telefon bankacılığı ve sesli yanıt sistemleri programcısı ve sistem sorumlusu • Proje Sorumlusu Ağustos 1997 – Yapı Kredi Teknoloji Hizmetleri A.Ş. Gebze / KOCAELİ Bütün YKB bilgisayar ortamları yazılım dağıtım birimi sorumlusu, tasarım uzmanı
Yabancı Dil	• İngilizce (İleri seviyede) • Almanca ve Fransızca (başlangıç seviyesinde)
Bilgisayar	• Microsoft Windows işletim sistemleri, Unix, OS / 2, AS/400 • Microsoft SQL Server, Ofis Uygulamaları C, C++, Java, Pascal, Delphi, Rexx programlama dilleri

EK-1 GÖNDERİCİ BİLGİSAYARDAKİ KODLAR

```
if (ParamCount = 4) then
```

```
begin
```

```
    Server:=ParamStr(1);
```

```
    Rport:=strtoint(ParamStr(2));
```

```
    ZUserid:=ParamStr(3);
```

```
    mesaj:=ParamStr(4);
```

```
    if Length(Server) > 0 then
```

```
        begin
```

```
            C:=ZUserid + ' : ' + mesaj;
```

```
            NMUDP1.RemoteHost := Server;
```

```
            NMUDP1.RemotePort := Rport;
```

```
            NMUDP1.ReportLevel := Status_Basic;
```

```
            MyStream := TMemoryStream.Create;
```

```
            try
```

```
                MyStream.Write(C[1], Length(C));
```

```
                NMUDP1.SendStream(MyStream);
```

```
            finally
```

```
                MyStream.Free;
```

```
            end;
```

```
        end;    // if Length(Server) > 0 then
```

```
    end // if (ParamCount = 4) then
```

EK-2 SERVER BİLGİSAYARDAKİ KODLAR

```
procedure TRec_Send_Form.NMUDP_RECDataReceived(Sender:
```

```
TComponent;
```

```
NumberBytes: Integer; FromIP: String; Port: Integer);
```

```
var
```

```
C: String;
```

```
MyStream: TMemoryStream;
```

```
found: boolean;
```

```
i,choice :integer;
```

```
begin
```

```
/* Aşağıdaki kısımda gelen mesaj sistem değişkenlerine alınır. */
```

```
MyStream := TMemoryStream.Create;
```

```
try
```

```
NMUDP_REC.ReadStream(MyStream);
```

```
SetLength(C, NumberBytes);
```

```
MyStream.Read(C[1], NumberBytes);
```

```
Memo1.Lines.Add(C+' - received from '+FromIP);
```

```
finally
```

```
MyStream.Free;
```

```
end;
```

```
choice:=0;
```

```
/* Aşağıdaki kısımda bir izleyici bilgisayardan gelen kayıt olma isteği değerlendirilir. */
```

```
if (C='BANA DA YOLLA') then
```

```
begin
```

```
found:=false;
```

```
for i:=1 to maxtarget do
```

```
begin
```

```
if (SENDIP[i]=FromIP) then
```

```
begin
```

```
found:=true;
```

```
choice:=i;
```

```
end;
```

```

end;

if (not found) and (sendcount<maxtarget) then
begin
  for i:=1 to maxtarget do
  begin
    if (NMUDPSEND[i].Tag=0) then
    begin
      choice:=i;
      inc(sendcount);
      SENDIP[choice]:=FromIP;
      NMUDPSEND[choice].RemoteHost:=SENDIP[choice];
      NMUDPSEND[choice].Tag:=1;
      StatusBar1.SimpleText:=StatusBar1.SimpleText + ' - ' + SENDIP[choice];
      break;
    end; // if (NMUDPSEND[i].Tag=0) then
  end; // for i:=1 to maxtarget do

end; // if (not found) and (sendcount<maxtarget) then

SendMessage(C,choice);

end // if (C='BANA DA YOLLA') then

```

/* Kayıt silme isteği değerlendirilir. */

```

else if (C='ARTIK BANA YOLLAMA') then
begin
  found:=false;
  for i:=1 to maxtarget do
  begin
    if (SENDIP[i]=FromIP) then
    begin
      found:=true;
      choice:=i;
      break;
    end;
  end;

```

end;

if (found) then

begin

dec(sendcount);

SendMessage(C,choice);

SENDIP[choice]:= "";

NMUDPSEND[choice].Tag:=0;

NMUDPSEND[choice].RemoteHost:= "";

StatusBar1.SimpleText:='Gönderilecek Host :';

for i:=1 to maxtarget do

begin

if (NMUDPSEND[i].Tag <> 0) then

StatusBar1.SimpleText:=StatusBar1.SimpleText + ' - ' +

NMUDPSEND[i].RemoteHost;

end;

end; // if (found) then

end // else if (C='BANA YOLLAMA') then

/* Yeni bir yazılım dağıtım başlatma isteği değerlendirilir. */

else if (Copy(C,1,11)='NEW DIST : ') then

begin

found:=false;

for i:=1 to maxtarget do

begin

if (SENDIP[i]=FromIP) then

begin

found:=true;

newdistchoice:=i;

WillBeLogged:=True;

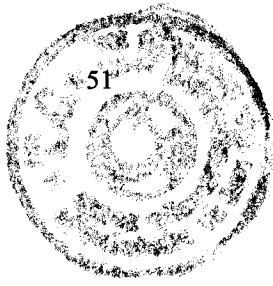
WriteLogFile:=Copy(C,12,length(C)-11);

strncpy(writefile,datapath + '\ ' + Writelogfile+ '.log');

break;

end;

end;



```
if (found) then
begin
    SendMessage(WriteLogFile, newdistchoice);
end; // if (found) then
end // else if (Copy(C,1,11)='NEW DIST : ') then
```

**/* Mevcut yazılım dağıtım isimlerinin listesini alma isteği değerlendirilir.
*/**

```
else if (C='GET DIST LIST') then
begin
    found:=false;
    for i:=1 to maxtarget do
    begin
        if (SENDIP[i]=FromIP) then
        begin
            found:=true;
            senddistchoice:=i;
            break;
        end;
    end;
end;
```

```
if (found) then
begin
    FileListBox1.Directory:=datapath;
    FileListBox1.Update;
    Application.ProcessMessages;

    for i:=0 to FileListBox1.Items.Count-1 do
    begin
        SendMessage('DIST LIST : ' + FileListBox1.Items.Strings[i], senddistchoice);
        Sleep(50);
    end;
    SendMessage('DIST LIST ENDED', senddistchoice);
end; // if (found) then
```

end // else if (C='GET DIST LIST') then

/* Seçilen yazılım dağıtım ismine ait kayıt edilmiş mesajlar izleyici bilgisayara gönderilir. */

else if (Copy(C,1,12)='SEND DIST : ') then

begin

found:=false;

for i:=1 to maxtarget do

begin

if (SENDIP[i]=FromIP) then

begin

found:=true;

senddistchoice:=i;

ReadLogFile:=Copy(C,13,length(C)-12);

break;

end;

end;

if (found) then

begin

SendMessage(ReadLogFile, senddistchoice);

LoadLog;

end; // if (found) then

end // else if (Copy(C,1,12)='SEND DIST : ') then

/* Şu anda aktif olan yazılım dağıtım ismi izleyici bilgisayara gönderilir. */

else if (C='GET CURRENT DIST') then

begin

found:=false;

for i:=1 to maxtarget do

begin

if (SENDIP[i]=FromIP) then

begin

found:=true;



```
    senddistchoice:=i;  
    break;  
end;  
end;
```

```
if (found) then  
begin  
    if WillBeLogged then SendMessage('CURRENT DIST : '+ WriteLogFile,  
        senddistchoice);  
end; // if (found) then  
  
end // else if (C='GET CURRENT DIST') then
```

/* Şu anda aktif olan yazılım dağıtımı sona erdirilir. */

```
else if (Copy(C,1,11)='END DIST : ') then  
begin  
    found:=false;  
    for i:=1 to maxtarget do  
        begin  
            if (SENDIP[i]=FromIP) then  
                begin  
                    found:=true;  
                    senddistchoice:=i;  
                    break;  
                end;  
        end;  
    end;  
  
    if (found) then  
        begin  
            if (Copy(C,12,length(C)-11)=WriteLogFile) then  
                begin  
                    WillBeLogged:=false;  
                    SendMessage('DIST ENDED : '+ WriteLogFile, senddistchoice);  
                end;  
        end; // if (found) then
```

end // else if (C='GET CURRENT DIST') then

**/* özel mesaj gelmedi ; gelen mesaj aynen İzleyici Bilgisayarlara
gönderilir */**

else

begin

Sendtoclients(C);

if WillBeLogged then SaveLog(C);

end;

end;



EK-3 İZLEME BİLGİSAYARINDAKİ KODLAR

```
procedure TMainForm.NMUDP1DataReceived(Sender: TComponent;  
NumberBytes: Integer; FromIP: String; Port: Integer);
```

```
var MyStream: TMemoryStream;
```

```
intext,inuserid,inind :shortstring;
```

```
kk,i,indi,isimi:integer;
```

```
begin
```

```
MyStream := TMemoryStream.Create;
```

```
try
```

```
    NMUDP1.ReadStream(MyStream);
```

```
    SetLength(intext, NumberBytes);
```

```
    MyStream.Read(intext[1], NumberBytes);
```

```
Finally
```

```
    MyStream.Free;
```

```
end;
```

```
/* Server'la iletişim aşamasında Server'dan mesaj bekleniyorsa */
```

```
if (WaitAck) then
```

```
begin
```

```
    if (FromIP=NMUDP_REG.RemoteHost) then
```

```
        begin
```

```
/* Server'a kayıt olma işlemi Server tarafından onaylandı. */
```

```
    if (intext='BANA DA YOLLA') then
```

```
        begin
```

```
            TimerAck.enabled:=False;
```

```
            WaitAck:=false;
```

```
            registered:=true;
```

```
            StatusBar1.SimpleText:=' ' + RegisterForm.RegisterToServerComboBox.text
```

```
+ ' adlı SERVER"a kayıtlısınız.';
```

```
            Screen.Cursor:=crDefault;
```

```
            RegisterForm.visible:=False;
```

```
            showMessage(FromIP + ' adresli SERVER"a başarılı bir şekilde kayıt oldunuz!
```

```
);
```

end

/* Server'dan kayıt silme işlemi Server tarafından onaylandı. */

else if (intext='ARTIK BANA YOLLAMA') then

begin

TimerAck.enabled:=False;

WaitAck:=false;

registered:=false;

StatusBar1.SimpleText:=' Herhangi Bir SERVER"a Kayıtlı Değilsiniz!';

Screen.Cursor:=crDefault;

UnRegisterForm.visible:=False;

showmessage(FromIP + ' adresli SERVER"dan başarılı bir şekilde kayıt
sildirdiniz! ');

end

/* Server'dan kaydedilmiş yazılım dağıtımlarının listesi geliyor. */

else if (copy(intext,1,12)='DIST LIST : ') then

begin

GetDistForm.GetDistComboBox.Items.Add(copy(intext,13,length(intext)-12));

End

/* Server'dan gelen yazılım dağıtım listesi bitti. */

else if (intext='DIST LIST ENDED') then

begin

TimerAck.enabled:=False;

WaitAck:=false;

Screen.Cursor:=crDefault;

GetDistForm.Show;

End

/* Server'dan mevcut yazılım dağıtım ismi geldi. */

else if (copy(intext,1,15)='CURRENT DIST : ') then

begin

txt1:=intext;

TimerAck.enabled:=False;

WaitAck:=false;

Screen.Cursor:=crDefault;

if MessageDlg(copy(txt1,16,length(txt1)-15)+ ' : adlı Yazılım Dağıtımını
bitirmek mi istiyorsunuz? ', mtConfirmation, [mbYes, mbNo], 0) = mrYes then

begin

Screen.Cursor:=crHourGlass;

WaitAck:=True;

TimerAck.enabled:=True;

SendUdpMes('END DIST : '+copy(txt1,16,length(txt1)-15));

end;

end

/* Server, Yazılım Dağıtımını bitirme işlemini onayladı. */

else if (copy(intext,1,13)='DIST ENDED : ') then

begin

TimerAck.enabled:=False;

WaitAck:=false;

Screen.Cursor:=crDefault;

MessageDlg(copy(intext,14,length(intext)-13)+' adlı Yazılım Dağıtımı bitti.',
mtInformation,
[mbOk], 0);

end

/* Server , Yeni Yazılım Dağıtımı başlatma işlemini onayladı. */

else if (intext=StartNewDistForm.StartNewDistEdit.Text) then

begin

TimerAck.enabled:=False;

WaitAck:=false;

Screen.Cursor:=crDefault;

StartNewDistForm.visible:=False;

showmessage(intext + ' isimli yeni bir Yazılım Dağıtım başlatıldı.');

end;

end; // if (FromIP=NMUDP_REG.RemoteHost) then

```
end // if (WaitAck) then
```

```
/* Server 'dan özel mesaj gelmedi, Durum mesajları geliyor. Gelen  
mesajlara göre kullanıcı kodları gerekli durum mesajlarının altına  
yerleştiriliyor */
```

```
else
```

```
begin
```

```
if (intext='SEND DIST ENDED') then
```

```
begin
```

```
Screen.Cursor:=crDefault;
```

```
exit;
```

```
end;
```

```
/* Mesajı gönderen kullanıcı kodu ve gönderilen durum mesajı tespit  
ediliyor.*/
```

```
kk:=Pos(':',intext);
```

```
inuserid:=copy(intext,1,kk-2);
```

```
inind:=Trim(copy(intext,kk+2,3));
```

```
indi:=0;
```

```
isimi:=0;
```

```
/* Mesajı gönderen kullanıcı kodunun endeksi belirleniyor */
```

```
for i:=1 to usercount do
```

```
begin
```

```
if (inuserid=useridl[i]) then
```

```
begin
```

```
isimi:=i;
```

```
break;
```

```
end;
```

```
end;
```

```
if (isimi=0) then
```

```
begin
```

```
exit;
```

```
end;
```

/* Gönderilen mesajlar her bir kullanıcı kodu için dizilerde tutuluyor */

```
if (MemoStrNo[isimi] < MAXMESSAGECOUNT) then
begin
    inc(MemoStrNo[isimi]);
    MemoStr[isimi][MemoStrNo[isimi]]:=intext;
end;
```

/* Gönderilen durum mesajının endeksi belirleniyor */

/* Gönderenin mevcut durumu kaydediliyor , bitiş zamanı belirleniyor */

```
for i:=1 to indcount do
begin
    if (inind=indstr[i]) then
    begin
        indi:=i;
        Userstatus[isimi]:=indi;
        UserPath[isimi][indi]:=indstr[i];
        UserEndTime[isimi]:=Now;
        if (self_connected_user) then
        begin
            if (indi=2) then
            begin
                UserAtt[isimi]:=true;
                UserStartTime[isimi]:=Now;
            end
            else if (indi=1) then
            begin
                UserAtt[isimi]:=false;
            end;
        end; // if (self_connected_user) then
        break;
    end;
end;

if (indi=0) then exit;
```

/* Gönderenin ilk mesajı ise başlama zamanı belirleniyor */

if (UserStartTime[isimi]=0) then UserStartTime[isimi]:=Now;

/* Gönderen Kullanıcı Kodu gerekli yere yerleştirilmek üzere yerleştirme prosedürü çağrılıyor */

PlaceUser(isimi,indi,true);
end; //else
end;

/* Yerleştirme prosedürü */

procedure
TMainForm.PlaceUser(isimi:integer;indi:integer;will_be_logged:boolean
);

var

i: integer;
attended,finished :boolean;

begin

if (UserAtt[isimi]) then isim[isimi].font.style := [fsUnderline]
else if not (isim[isimi].font.style=[fsUnderline]) then isim[isimi].font.style:=[]
else if ((isim[isimi].font.style=[fsUnderline])and (indi=1)) then
isim[isimi].font.style:=[];

/* Kullanıcı kodu eski yerinden siliniyor */

if (isimcoord[isimi][0]<>0) then // if filled in somewhere before
begin
fixcolumnn(isimcoord[isimi][0],isimcoord[isimi][1]);
dec(indentrycount[isimcoord[isimi][0]]);
end;

/* Kullanıcı kodu yeni yerinde boş olan ilk satıra yerleştiriliyor */

isimcoord[isimi][0]:=indi; // set new x coord

```

    isimcoord[isimi][1]:=giveycoord(indi); // set new y coord
    indyfil[isimcoord[isimi][0]][isimcoord[isimi][1]]:=isimi; // set new y coord to 1
    inc(indentrycount[indi]);
    isim[isimi].left:=indlab[indi].left - ((isim[isimi].width - indlab[indi].width)div 2);
    isim[isimi].top:=status_top_margin + 3 + ((isimcoord[isimi][1])+2) *
status_row_height;

```

```

if (indi=indcount) then
begin
    isim[isimi].font.color:=completedcolor;
    inc(indlastcount);
    finished:=true;
    for i:=1 to indcount-1 do
        if (indentrycount[i]>0) then finished:=false;

```

/* Gelen bütün mesajlar dosyaya yazılıyor */

```

    if (will_be_logged) then LogToFile(isimi);

```

/* Son mesajı gönderen kullanıcı kodu için Timer çalıştırılıyor. */

```

if (not(Timer5.enabled)) then Timer5.enabled:=true;

```

```

end; // if (indi=indcount) then
end;

```

/* 5 saniye dolunca Time kodu çalışıyor ve kullanıcı kodunu ekranın aşağısındaki yere atılıyor. */

```

procedure TMainForm.Timer5Timer(Sender: TObject);
begin
    Timer5.enabled:=false;
    // decindlast;
    if (indlastcount > 0) then decindlast;
end;

```

EK-4 ÖRNEK BİR "AYARLAR.INI" DOSYASI

[NETWORK]

Listen UDP On Port=6667

Send Acknowledge On Port=6666

Max Connections=15

Period=200

1st Port to Connect=3462

2nd Port to Connect=139

[REGISTER]

YKY : 10.8.8.150

[STATUSCOORD]

Status_Font_Size=12

left_margin=20

top_margin=17

column_width=49

row_height=12

status_message_length=3

[USERIDCOORD]

self_connected_user=TRUE

Userid_Length_in_Chars=8

Userip_Length_in_Chars=75

Use Chars from 1 to 3 and from 7 to 8 and from 1 to 0

left_margin=10

top_margin=580

row_count=7

column_width=42

row_height=12

[DETAILSCOORD]

Details_Form_left=150

Details_Form_top=100

Details_Form_width=550

Details_Form_height=350

left_margin=110

top_margin=9

row_height=15

[COLORS]

Details CheckBox Not Selected Color=CISilver



Details CheckBox Selected Color=CILime

Status Label Color=CIWhite

Status Background Color=CIBlue

Client Initial color=CIYellow

Client Completedcolor=CILime

Start Check Color=CIGray

Hostname Not Resolved Color=CIFuchsia

Active Connnectios Color=CIWhite

Connection OK Color=CILime

Failed On Connection Color=CIRed

Failed on 1 but OK on 2 Color=CIYellow

[USERID]

ADAN__sr 10.162.7.10

ANKA__sr 10.162.1.10

ANTL__sr 10.162.17.10

BAKR__sr 10.162.16.10

BURS__sr 10.162.5.10

GOZT__sr 10.162.11.10

IZMR__sr 10.162.3.10

LEVN__sr 192.168.83.31

OSMN__sr 10.162.13.10

PLAZ__sr 192.168.83.22

SULT__sr 10.162.9.10

TEST__sr 192.168.83.27

OSMN_001 10.162.13.187

OSMN_002 10.162.13.120

OSMN_004 10.162.13.15

OSMN_005 10.162.13.14

OSMN_006 10.162.13.21

OSMN_007 10.162.13.116

OSMN_008 10.162.13.18

OSMN_009 10.162.13.19

OSMN_010 10.162.13.20

OSMN_011 10.162.13.17

BAKR_001 10.162.16.12

BAKR_002 10.162.16.14

BAKR_003 10.162.16.16

[IND]

- 031 : Notify ile Başladı.
- 032 : Yazılım Dağıtımı kendisi başlattı.
- 07 : Merkezle Komünikasyon kuruluyor.
- 08 : Hardware Bilgisi alınıyor.
- 10 : Logon oluyor.
- 19 : ZMASTER çözümlemesi yapılıyor.
- 20 : ZService/ZRsource kontrolleri yapılıyor.
- 21 : EDMCON00 - Client ve Manager ZService'leri karşılaştırılıyor.
- 25 : EDMCON20 - Client ve Manager ZRsource'ları karşılaştırılıyor.
- 45 : EDMCON30 - Hangi dosyaların indirileceği belirleniyor.
- 50 : EDMSTGRQ - Stager'dan (UNIX Server) dosyalar transfer ediliyor.
- 63 : EDMGETRS - EDM:Manager 'dan (OPC) dosyalar transfer ediliyor.
- 67 : EDMCON40 - İndirilen dosyalar Update ediliyor.
- 69 : EDMCON50 - Metod'lar çalıştırılıyor.
- 74 : EDMSELRS - Versiyon değişiklikleri kontrol ediliyor.
- 80 : EDM Connection başarılı bir şekilde bitti.
- 85 : EDMSTATE.log yazılıyor.
- 90 : Report Server için merkeze Objeler gönderiliyor.
- 95 : Report Server için merkeze ZCLIENT gönderiliyor.
- 100 : Komünikasyon bitti.