İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ZAMANLANMIŞ RENKLİ PETRİ AĞLARI İLE OTURUM BAŞLATMA PROTOKOLÜ (OBP)'nün MODELLENMESİ

YÜKSEK LİSANS TEZİ

Safiye KIZMAZ

Anabilim Dalı: Elektronik & Haberleşme Mühendisliği

Programı: Elektronik Mühendisliği

HAZİRAN 2010

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

ZAMANLANMIŞ RENKLİ PETRİ AĞLARI İLE OTURUM BAŞLATMA PROTOKOLÜ (OBP)'nün MODELLENMESİ

YÜKSEK LİSANS TEZİ

Safiye KIZMAZ

(504061223)

Tezin Enstitüye Verildiği Tarih : 07 Mayıs 2010
Tezin Savunulduğu Tarih : 10 Haziran 2010
Tez Danışmanı : Doç. Dr. Mürvet KIRCI(İTÜ)
Diğer Jüri Üyeleri : Prof. Dr. Leyla GÖREN(İTÜ)
Doç. Dr. Selçuk PAKER(İTÜ)

HAZİRAN 2010

ÖNSÖZ

Tez çalışmamda rehberliğini ve değerli zamanını esirgemeyen, bilgi ve önerilerini her zaman paylaşan değerli danışman hocam Doç. Dr. Mürvet KIRCI'ya, bugünlere gelmemde emeği geçmiş tüm değerli hocalarıma teşekkürü bir borç bilirim.

Ayrıca yüksek lisans eğitimim süresince manevi destekleri ve sevgileri ile yanımda olan annem, babam ve ablama, her düştüğümde beni elimden tutup kaldıran, sevgisiyle bana güç veren nişanlım Boğaç TURGUT'a ve bu zor yolculuğumda yanımda olan bütün arkadaşlarıma teşekkürlerimi sunarım.

Haziran 2010

Safiye KIZMAZ

Elektronik ve Haberleşme Müh.

İÇİNDEKİLER

Sayfa

ÖNSÖZ ... **iii**
İÇİNDEKİLER ... **v**
KISALTMALAR ... **vii**
ÇİZELGE LİSTESİ ... **ix**
ŞEKİL LİSTESİ ... **xi**
ÖZET ... **xiii**
SUMMARY ... **xv**
1. GİRİŞ ... **1**
1.1 Petri Ağları-PA(Petri Nets-PNs) ... 3
1.1.1 Durum Uzayı ... 4
1.1.2 Petri Ağının Özellikleri ... 4
1.1.2.1 Canlılık ... 4
1.1.2.2 Sınırlılık ... 5
1.1.2.3 Konservatiflik ... 5
1.1.2.4 Uygunluk ... 5
1.1.2.5 Erişebilirlik ... 5
1.2 Oturum Başlatma Protokolü-OBP ... 5
1.2.1 Oturum Başlatma Protokolü Bileşenleri ... 6
1.3 Petri Ağları ile Oturum Başlatma Protokolü Üzerine Yapılmış Çalışmalar ... 7
1.4 Petri Ağı Modellenmesinde ve Analizinde Kullanılan Yazılım Araçlarının Karşılaştırılması ... 8
2. ZAMANLANMIŞ RENKLİ PETRİ AĞLARI (ZRPA) ... **13**
2.1 Zamanlanmış Renkli Petri Ağlarında Kullanılan Tanımlamalar ... 14
3. OTURUM BAŞLATMA PROTOKOLÜ'nün İŞLEMLERİ ... **23**
3.1 Kullanıcı İşlemi ... 24
3.1.1 Davet İçeren Kullanıcı İşlemi ... 25
3.1.2 Davet İçermeyen Kullanıcı İşlemi ... 25
3.2 Sunucu İşlemi ... 26
3.2.1 Davet İçeren Kullanıcı İşlemi ... 26
3.2.2 Davet İçermeyen Kullanıcı İşlemi ... 27
3.3 OBP İşlemlerinde Kullanılan Zamanlayıcıların Birbirleriyle İlişkileri ... 27
4. CPN TOOLS PROGRAMININ ÖZELLİKLERİ ... **29**
4.1 Petri Ağının CPN Tools Programı ile Analizi ... 29
4.1.1 Benzetim ... 29
4.1.2 Durum Uzayı Analizi ... 30
4.1.3 Performans Analizi ... 31
5. OLUŞTURULAN MODELLER ve ANALİZLERİ ... **33**
5.1 OBP'nin DAVET İşleminin Modeli ve Analizi ... 33
5.1.1 DAVET Modelinin Durum Uzayı Analizi ... 36
5.1.2 DAVET Modelinin Performans Analizi ... 39
5.2 OBP'nin DAVETSİZ İşleminin Modeli ve Analizi ... 43
5.2.1 DAVETsiz Modelinin Durum Uzayı Analizi ... 46

5.2.2 DAVETsiz Modelinin Performans Analizi 48
5.3 Klasik Petri Ağı Grafiği Kullanılarak Oluşturulan Model ve Analizi 51
5.3.1 DAVET İşleminin Modeli 52
5.3.2 Yazılım 54
5.3.3 DAVET İşleminin Modelinin Analizi 56
6. SONUÇLAR VE ÖNERİLER 57
KAYNAKLAR 59
EKLER 63
ÖZGEÇMİŞ 113

KISALTMALAR

AODS	**:** Ayrık Olay Dinamik Sistemleri
BZ	**:** Birim Zaman
KT	**:** Kullanıcı Temsilcisi
KTİ	**:** Kullanıcı Temsilcisi İstemcisi
KTS	**:** Kullanıcı Temsilcisi Sunucusu
OBP	**:** Oturum Başlatma Protokolü
PA	**:** Petri Ağı
PSTN	**:** Public Switched Telephone Network(Genel Aktarmalı Telefon Şebekesi)
RFC	**:** Request For Comments (Yorumlar için Talep)
RPA	**:** Renkli Petri Ağları
TU	**:** İşlem kullanıcısı(transaction user)
ZRPA	**:** Zamanlanmış Renkli Petri Ağları

ÇİZELGE LİSTESİ

Sayfa

Çizelge 1.1 : Yerler ve geçişlerin mümküm olabilecek anlamları. 2
Çizelge 3.1 : OBP Cevap Mesajı Kodları 23
Çizelge 5.1 : Modelde Kullanılan Düğümlerin Anlamları 35
Çizelge 5.2 : Davet Modeli Ölü İşaretlemeler 37
Çizelge 5.3 : Performans Analizi Zamanlı Sonuçları 42
Çizelge 5.4 : Performans Analizi Zamansız Sonuçları 42
Çizelge 5.5 : DAVETsiz İşleminin Modelinde Kullanılan Düğümlerin Anlamları...45
Çizelge 5.6 : Davet Modeli Ölü İşaretlemeler 46
Çizelge 5.7 : Performans Analizi Zamanlı Sonuçları 50
Çizelge 5.8 : Performans Analizi Zamansız Sonuçları 51
Çizelge 5.9 : Modelde Kullanılan Düğümlerin ve Renklerin Anlamları.....53
Çizelge A.1 : GönderilecekPaketler Monitörü İçin Veri Toplama Günlük Dosyası .87
Çizelge A.2 : Ortalama Paket Gecikmesi İçin Tutulan Günlük Dosyası 88

ŞEKİL LİSTESİ

Sayfa

Şekil 1.1 : Petri Ağı Bileşenleri....4
Şekil 2.1 : Tanımlamaları Örneklemek için Kullanılan Model....18
Şekil 4.1 : CPN Tools Programında Kullanılan Benzetim Araçları....30
Şekil 5.1 : DAVET Modelinin Durum Uzayı Grafı....37
Şekil 5.2 : Performans Analizi için Modelin En Üst Modeli....39
Şekil 5.3 : Performans Analizi için Modelin Mesajın Ulaşması Modeli....40
Şekil 5.4 : DAVETsiz Modelinin Durum Uzayı Grafı....47
Şekil 5.5 : Performans Analizi için Modelin En Üst Modeli....48
Şekil 5.6 : Performans Analizi için Modelin Mesajın Ulaşması Modeli....49
Şekil 5.7 : DAVET İşleminin Klasik Renkli Petri Ağları Modeli....52
Şekil A.1 : CPN Tools Programının Arayüzü....72
Şekil A.2 : CPN Tools Programının Elemanları Yaratma Aracı....73
Şekil A.3 : CPN Tools Programının Benzetim Aracı....73
Şekil A.4 : CPN Tools Programı ile Oluşturulan Basit Bir Protokol Modeli....74
Şekil A.5 : Protokol Örneğinin Bir Benzetim Adımından Sonraki İşaretlemesi....77
Şekil A.6 : CPN Tools Programı ile Oluşturulan Zamanlanmış Renkli Petri Ağları ile Protokol Modeli....78
Şekil A.7 : CPN Tools Programının Elemanları Durum Uzayı Aracı....81
Şekil A.8 : CPN Tools Programının Elemanları Durum Uzayı Raporu....82
Şekil A.9 : CPN Tools Programının Durum Uzayı Başlangıç Özellikleri....82
Şekil A.10 : CPN Tools Programının Durum Uzayı Canlılık Özellikleri....82
Şekil A.11 : CPN Tools Programının Durum Uzayı Sınırlık Özellikleri....83
Şekil A.12 : CPN Tools Programının Durum Uzayı Jeton Renkleri ile Gösterilen Sınırlık Özellikleri....84
Şekil A.13 : Performans Analizi için Oluşturulan Zamanlı Protokol Modeli....85
Şekil A.14 : Benzetim Performans Raporunun İstatistikleri....88

ZAMANLANMIŞ RENKLİ PETRİ AĞLARI İLE OTURUM BAŞLATMA PROTOKOLÜ (OBP)'nün MODELLENMESİ

ÖZET

Ayrık olay sistemlerinin modellenmesinde kullanımı gittikçe artan petri ağları bu tez çalışmasının temelini oluşturmaktadır. İnternetin hızla yayılmasının bir sonucu olarak doğan ve Internet Protokolü üzerinden çağrı yapılmasına olanak sağlayan Oturum Başlatma Protokolü(OBP) petri ağlarının gelişmiş bir tipi olan Zamanlanmış Renkli Petri Ağları ile modellenmiş ve analizi yapılmıştır. Oluşturulan modellerin erişilebilirlik ağacı ve performans analizleri petri ağlarının modellenmesinde en çok kullanılan yazılım aracı olan CPN Tools programı ile yapılmıştır. Daha sonra model klasik zamanlanmış renkli petri ağı modeliyle yeniden oluşturulmuş ve yazılan bir program ile analizleri gerçekleştirilerek karşılaştırma yapma olanağı sağlanmıştır. Öncelikle temel anlamda petri ağlarının ve OBP'nin yapısı anlatılmıştır.

İkinci bölümde Zamanlanmış Renkli Petri ağları hakkında bilgi verilmiş ve örnek bir model üzerinde CPN Tools programının kullandığı formüller ispatlanmıştır.

Üçüncü bölümde Oturum Başlatma Protokolünün işlemlerinden bahsedilmiştir. İstekleri karşılamak, mesajlara cevap vermek, iletim ortamı güvenilir olmadığında mesajları yeniden iletmek ve süre aşımının üstesinden gelmekle sorumlu olan işlem katmanının işlemlerinin durum makinesi verilerek sağladığı özellikler belirtilmiştir. Oturum Başlatma Protokolünde kullanılan zamanlayıcılar ve birbirleriyle olan etkileşimleri belirtilmiştir.

Dördüncü bölümde modellerin kurulmasında, erişilebilirlik ağacının çıkarılmasında ve analizlerinde kullanılan CPN Tools programının özellikleri, kullanımı anlatılmıştır. Örnekler üzerinden durum uzayı ve performans analizlerinin nasıl yapıldığından bahsedilmiştir.

Beşinci bölümde ise OBP'nin DAVET ve DAVETsiz işlemlerinin modelleri verilmiş durum uzayı ve performans analizleri yapılmıştır. Son olarak CPN klasik petri ağı grafı kullanılarak DAVET modeli yeniden oluşturulmuş ve yazılan bir programla analizleri yapılmıştır. Bu iki benzetim ve analiz arasındaki farklardan bahsedilmiştir.

Son bölüm olan altıncı bölümde yapılan çalışmanın sonuçları ve gelecekte yapılabilecek çalışmalar için önerilerde bulunulmuştur.

MODELLING OF SESSION INITIATION PROTOCOL USING TIMED COLORED PETRI NETS

SUMMARY

Petri Nets which are useful tools to analyze and model of discrete event systems underlie this study. As a result of the rapid spread of the Internet emerged over Internet Protocol, which allows calls only Session Initiation Protocol(SIP) environment analysied and developed with as a type of advanced Petri nets; timed colored Petri nets. Generated model's reachability trees and performance analysis are performed by CPN Tools program which is the most widely used in petri nets modelling. Thereafter INVITE's model is formed again with classical timed coloured petri nets graphs and analyzed with a written program. Provided making comparisons are possible. First of all petri nets and SIP basic structure is explained.

In the second part, Timed Colored Petri Nets are explained and formulas which are used by CPN Tools program are demonstrated.

In the third part, SIP are examined in detail. Transactions features about meeting the requests, replying to messages, retransmitting the message when the environment is unreliable, handling the timeouts are indicated over state machine. Timers which are used in SIP and their interactions with each other are explained.

In the fourth part, CPN Tools program features and usage which are used for forming models and developing reachability tree are explained. How it was made of the state space and performance analysis are discussed via examples.

In the fifth part, SIP INVITE transaction and non-INVITE transaction models are given and their state space and performance analysis are done. Finally INVITE's model is formed with classical timed coloured petri nets graphs and analyzed with a written program. Differences are informed.

At sixth part which is the last part, results of this study and suggessions for the future studies are given.

1. GİRİŞ

Son zamanlarda teknolojinin hızla gelişmesi ile haberleşme ağları-protokolleri, imalat sistemleri, trafik ve mantıksal kontrol sistemleri, askeri kontrol ve emir sistemleri gibi birçok ayrık olaylı dinamik sistemler bulunmuş ve her alanda kullanıma geçmiştir. İşte bu sistemlerin modellenmesi için 1939 yılında Carl Adam Petri tarafından petri ağları ortaya konulmuştur. Petri ağları, ayrık sistemlerin tanımlanması için kullanılan matematiksel modelleme dillerinden biridir. Bu ağlar bilgisayar teknolojisinde eş zamanda çalışan işlerin modellenmesi ve çözülmesinde kullanılan özel grafiklerdir.

Zamanlanmış petri ağları ise ayrık olaylı dinamik sistemler için kullanılan güçlü performans modelleme araçlarından biridir. Sistemlerin zamanı çeşitli ayrık olayların zamanlanmasındaki karmaşık etkileşime dayanır. Bu etkileşim, bir işin başlaması ve bitmesi, bir görevin tamamlanması veya mesajın iletilmesi olaylarından oluşur. Bu gibi dinamik sistemlerin durumu sürekli olarak değil anlık olarak değişir. İşte böyle sistemlere ayrık olaylı dinamik sistemler(AODS) denir. Bu sistemler fiziksel olarak sürekli değerli dinamik sistemlerden farklıdır ve farklı eşitlikler ile tanımlanır.

Petri ağları, olaylar ve şartlar arasındaki ilişkiyi tanımlayabilmek için genel amaçlı bir matematiksel araç olarak ifade edilmiştir. İlerleyen zamanlarda eşleme, asenkron olaylar, sıralı işlemler, eş zamanlı işlemler ve çakışmalar veya kaynak paylaşımı gibi özellikleri modellemek için petri ağlarında önemli araştırmalar yapılmıştır. Bu özellikler AODS'leri karakterize eder. Bu ve bir takım faktörler petri ağlarını AODS'lerin çeşitli tipteki uygulamaları için gelecek vadeden bir araç haline getirmiştir.

Oturum Başlatma protokolü (OBP) 1990'lı yılların ortalarında, üniversitelerde gelen büyük ölçekli çoğul ortam konferansı yapabilmeye olanak tanımak için yapılan araştırmalardan doğmuş ve şu an internet üzerinden çoklu ortam hizmetlerinin işletilebilmesi için kullanılan bir protokol haline gelmiştir.

OBP İnternet merkezli çoğul ortam iletişim mimarisinin temelini oluşturmaktadır. OBP, haberleşmek isteyen kişiler için Internet Protokolü(IP) ağları üzerinden oturumlar açar. Bu kapsamda, bir oturum, IP ağı üzerinden iki veya daha fazla kişi arasındaki etkileşimli bir iletişimdir. OBP bir ya da daha fazla katılımcı arasında iletişim oturumlarını açan, değiştiren ve sona erdiren bir kontrol protokolüdür. Bu protokol katılımcıların bir dizi uyumlu medya tipi üzerinde anlaşmasını sağlar ve istekleri her kullanıcının geçerli konumunda karşılayarak ve yönlendirerek kullanıcı gezginliğini destekler.

OBP, görevlerini iki işlem üzerinden gerçekleştirir. Bunlar oturumu kuran DAVET işlemi ve oturumu yöneten ve sonlandıran DAVETsiz işlemidir.

Bu tez çalışmasında, petri modellenmesinde sıklıkla kullanılan CPN Tools programı ile zamanlanmış renkli petri ağları kullanılarak bu işlemlerin modellenmesi ve analizleri yapılmıştır. Daha sonra DAVET işleminin modeli klasik yöntemle tekrar kurulmuş ve Dev-C++ derleyicisi kullanılarak oluşturulan bir yazılımla bu yeni modelin durum uzayı analizleri çıkarılmıştır. OBP zamanlanmış renkli petri ağları ile ilk kez bu tez çalışmasında modellenmiş ve benzetimleri yapılmıştır. Ayrıca yazılan program, öğrenilmesi zaman alan CPN Tools programı sadeleştirilerek sadece amaca yönelik olarak oluşturulmuştur. OBP dışında birçok protokol modellenmesinde de kullanılabilir hale gelmiştir.

Tezin organizasyonuna değinmek gerekirse; öncelikle temel anlamda petri ağlarının ve OBP'nin yapısı anlatılmıştır. İkinci bölümde Zamanlanmış Renkli Petri ağları hakkında bilgi verilmiş ve örnek bir model üzerinde CPN Tools programının kullandığı formüller ispatlanmıştır. Üçüncü bölümde Oturum Başlatma Protokolünün işlemlerinden bahsedilmiştir. Dördüncü bölümde modellerin kurulmasında, erişilebilirlik ağacının çıkarılmasında ve analizlerinde kullanılan CPN Tools programının özellikleri, kullanımı anlatılmıştır. Beşinci bölümde ise OBP'nin DAVET ve DAVETsiz işlemlerinin modelleri verilmiş durum uzayı ve performans analizleri yapılmıştır. Son olarak klasik zamanlanmış renkli petri ağı grafikleriyle modeller oluşturulmuş ve yazılan bir program ile analiz yapılarak aradaki farklar hakkında bilgi verilmiştir. Son bölüm olan altıncı bölümde yapılan çalışmanın sonuçları ve gelecekte yapılabilecek çalışmalar için önerilerde bulunulmuştur.

1.1 Petri Ağları-PA (Petri Nets-PNs)

Petri ağları yerler ve geçişler olmak üzere 2 düğümden oluşmaktadır. Bu yerler ve geçişler birbirlerine oklar ile bağlıdır. Geçiş sayısı belirli ve sıfır olmamalıdır. Oklar yerlerden geçişlere, geçişlerden yerlere yönlendirilmektedir. Oklar bir düğümde başlamalı ve bir düğümde sona ermelidir. Bir düğümden bir diğerine doğru bir ok bulunur. Yerler, durumlara veya koşullara (conditions); geçişler, olaylara karsı düşer. Oklar, genel olarak bir geçişten bir yere doğru yönlendirilmişse geçişe ait çıkış oku(O), bir yerden bir geçişe doğru yönlendirilmiş şekilde ise geçişe ait giriş oku (I) denir. En genel anlamda ise oklar ifade edilirken geçişlere ait giriş ve çıkış okları referans alınarak gösterilir.

Çizelge 1.1: Yerler ve geçişlerin mümkün olabilecek anlamları

Giriş Yeri	**Geçiş**	**Çıkış Yeri**
Ön Koşul	Olay	Art koşul
Giriş Bilgisi	Sıralama	Çıkış bilgisi
Giriş Sinyali	Sinyal İşleyici	Çıkış sinyali
Gerekli Kaynaklar	Görev ya da İş	Kaynakların Boşaltılması
Şartlar	Lojik koşul	Sonuçlar
Yedek bilgiler	İşlemci	Yedek bilgi

Petri ağlarında her yer, sistemin anlık durumunu veren yani durumlara ait bilgi taşıyan sıfır ya da pozitif değerlerde jeton(token) içerir. Bir yerde bulunan jetonlar o yerin ait olduğu koşulun gerçekleştiğini gösterir. Belirli bir zamanda petri ağında yerlerin sahip olduğu jeton sayısına (ağın durumuna) "İşaretleme" denir. Petri ağının ilk andaki (daha hiçbir geçişin ateşlenmediği) işaretlemesine "Başlangıç Durumu İşaretlemesi" denir. Petri ağının yapısı aşağıdaki gibi tanımlanabilir:

$N = (P, T, I, O, M_0)$

$P = \{p_1, p_2, \dots, p_m\}$ sonlu sayıda yerler kümesi

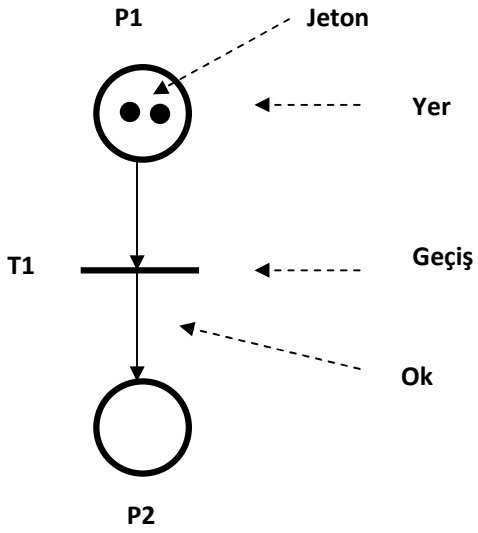
$T = \{t_1, t_2, \dots, t_n\}$ sonlu sayıda geçişler kümesi ($P \cup T \neq \emptyset; P \cap T = \emptyset$).

I : $(P \times T) \rightarrow N$ geçişler için giriş oklarının fonksiyonu

O : $(P \times T) \rightarrow N$ geçişler için çıkış oklarının fonksiyonu

$M_0 = P \rightarrow N$ başlangıç işaretlemesi

Petri ağları grafiksel olarak da gösterilebilir. Yerler daire, geçişler çubuk, jetonlar daireler içinde siyah nokta ile gösterilmektedir.



Şekil 1.1 : Petri ağının bileşenleri

1.1.1 Durum Uzayı

Petri Ağı'nın erişebileceği durumları ve bütün durum değişikliklerini hesaplamaktır.

1.1.2 Petri Ağının Özellikleri

Petri ağları birçok özelliğe sahiptir. Bu özellikler model tasarımda nelerin var olduğunu ya da eksik olduğunu görmemizi sağlar. Bunlar erişilebilirlik(reachability), sınırlılık(boundedness), canlılık(liveness), konservatiflik(conservation), uygunluk (fairness) özellikleridir.

1.1.2.1 Canlılık

Petri ağlarında işaretleme geçişlerin ateşlenmesi ile mümkündür. Ateşlenebilir geçişin bulunup yeni duruma geçebilen petri ağı canlıdır. Ağdaki hiçbir geçişin ateşlenemeyeceği duruma varıldığında "ağ kilitlendi" denir.

1.1.2.2 Sınırlılık

Bir petri ağında, çoğunlukla yerler, telekomünikasyon ve bilgisayar sistemlerinde bilgi, üretim sistemlerinde ürün ve araç depolama alanlarını göstermektedir. Bu alanların kapasitesinin bitmesi ile oluşan taşmadan korunmak için önerilen kontrol taktiklerini belirlemek önemlidir. Modellenmiş sistemde bu taşmaları bulmaya yarayan özellik sınırlılık özelliğidir. Örneğin, ağda bulunan jeton sayısının sonsuza gitmesi bu özellik sayesinde takip edilebilir.

1.1.2.3 Konservatiflik

Petri ağlarında jetonlar kaynakları göstermektedir. Gerçek sistem sayısı belirlendikten sonra bu sistemin petri ağındaki jetonların sayısı işaretlemeye bakmaksızın sabit kalmalıdır. Bu petri ağının konservatif özelliğinin bir sonucudur.

1.1.2.4 Uygunluk

Bu özellik geçişlerin hangi sıklıkta sonsuz olaylar sırasında oluştuğunu gösterir.

1.1.2.5 Erişilebilirlik

Erişebilirlik, bir sistemin dinamik özelliklerini hakkında bilgi edinmek için kullanılan en önemli özelliktir. Ayrık olaylı dinamik sistemlerin tasarımındaki en önemli unsur istenilen duruma erişebilmek ve özel fonksiyonel davranışları göstermesini sağlamaktır.

Bir modelin gerekli fonksiyonel davranışın bir sonucu olarak istenilen duruma nasıl ulaştığını bulmak için, M_o dan M_i'e geçiş arasındaki geçişlerdeki ateşleme sırasını bulmak ve bu sıralamayı göstermek gereklidir. Burada M_i ulaşılmak istenen durumu göstermektedir. M_o 'dan M_i'e dönüşümlerde herhangi bir ateşleme dizisi varsa M_i işaretlemesine M_o işaretlemesinden erişilebilir denir. Bu ateşleme dizisinin grafiksel gösterimine de "Erişilebilirlik Ağacı" adı verilir.

1.2 Oturum Başlatma Protokolü-OBP (Session Initiation Protocol-SIP)

OBP bir internet protokol (IP) ağı üzerinde, iki ya da daha fazla kullanıcı arasında oturumlar kurulmasını, yönetilmesini ve sonlandırılmasını sağlayan protokoldür. RFC 2543 ile tanımlanmış ve RFC 3261 ile geliştirilip olgunluğa kavuşturulmuştur. Ayrıca OBP internet telefonu için kullanılan en yaygın protokoldür.

OBP, uygulama katmanında çalışır ve düz metin bir protokoldür. Oturum açar, oturum parametrelerini değiştirir, oturumu sonlandırır. Oturumlar IP telefon çağrıları, çoklu ortam sunumları veya konferansı şeklinde olabilir. Aynı zamanda mevcut bir oturuma kullanıcı çağırabilir. Mevcut oturuma ortam ekleyebilir, çıkarabilir.

Genel olarak;

- OBP bir işaretleşme protokolüdür.
- Metin temellidir. Bu yüzden hata tespiti kolay, fakat band genişliği fazladır.
- OBP, ağ sunucusu olmadan çağrı kurulabilmesini sağlar.
- OBP, noktadan noktaya çoklu oturum sinyalleşme protokolüdür.
- Uç noktaların (User Agent) yerini ve yeteneklerini tespit eder. Adres çözümleme, adres eşleştirme ve çağrı yönlendirme işlemlerini bu sayede gerçekleştirir.
- İki uç arasında çağrıyı başlatır, çağrıyı yönlendirebilir, beklemeye alabilir, oturum parametrelerini değiştirebilir, çağrıyı sonlandırabilir.
- Kullanıcılar yönetimini gerçekleştirebilir. Çağrı bekletme, telesekreter, aktarma, çağrının niteliğini değiştirme... vb servisleri vardır.

1.2.1 OBP Bileşenleri

OBP, noktadan noktaya bir protokoldür. Her bir uç kullanıcı temsilcisi (KT) (User Agent-UA) olarak isimlendirilir. KT'ler OBP ağında aşağıdaki görevlerden birini yerine getirirler:

· Kullanıcı Temsilci İstemcisi – KTİ (User Agent Client -UAC): OBP isteği gönderen istemci uygulamadır. IP telefonları, yazılımsal OBP telefonları ve PSTN dönüşümünü sağlayan ağ geçitleri KTİ olarak davranırlar.

· Kullanıcı Temsilci Sunucusu – KTS (User Agent Server - UAS): İstemci isteklerini karşılayan ve cevap dönen sunucu uygulama. OBP vekil (proxy) sunucu, OBP yönlendirme (Redirect) sunucusu ve kayıt (registrar) sunucusu KTS olarak davranır.

1.3 Petri Ağları ile Oturum Başlatma Protokolü Üzerine Yapılmış Çalışmalar

Son yıllarda OBP'nin petri ağlarıyla modellenmesi üzerine çok sayıda çalışma yapılmıştır. Öncelikle Ekim 2007'de Yang Peng,Yuan Zhanting, Wang Jizeng "Petri Net Model of Session Initiation Protocol and Its Verification" isimli makaleyi yayınlamışlardır[13]. Burada klasik petri ağlarıyla OBP'nin temel erişebilirlik ağacı çıkarılmıştır. Aynı zamanlarda Huaxu Wan, "SIP for Mobile Networks and Security Model" isimli makaleyi yayınlamıştır[22]. Bu çalışmada OBP'nün kullanıcı işleminin renkli petri ağlarıyla modeli verilmiş ve bu protokolün güvenlik açığıyla ilgili tehditlerden bahsedilmiş ve sadece biri hakkında çözüm sunulmuştur.

Daha sonraki yıllarda Temmuz 2008'de Lay G. Ding ve Lin Liu tarafından "Modelling and Analysis of the INVITE Transaction of the Session Initiation Protocol Using Coloured Petri Nets" makalesi yayınlanmıştır[6]. Bu makale OBP ile oluşturulmuş en genel ilk makaledir. Burada renkli petri ağlarıyla DAVET İşleminin modeli sunulmuş ve modelin durum uzayı çıkarılmıştır. Bu durum uzayı çıkarılırken CPN Tools programı kullanılmıştır. Durum uzayında istenmeyen durumlar görülmüş ve bunların çözümü sunulmuştur. Model ve durum uzayı güvenli ortamda oluşturulmuş OBP'nün güvenlik açıkları göz ardı edilmiştir. Bu makalenin ardından Ocak 2009'da Lin Liu tarafından yukarıdaki makale geliştirilmiş ve "Verification of the SIP İşlem Using Coloured Petri Nets" isimli yeni bir makale çıkarılmıştır[12]. Burada güvensiz ortamda OBP'nün davet işleminin hem kullanıcı hem sunucu tarafının renkli petri ağlarıyla modeli verilmiş ve CPN Tools programı kullanılarak modelinin analizi yapılmıştır.

Bunlar dışında Aralık 2008 de Yanlan Ding, GuiPing Su ve Huaxu Wan tarafından "Handbook :SIP Modeling and Simulation" kitabı çıkarılmıştır. Renkli petri ağlarıyla DAVET işleminin kullanıcı ve sunucu modeli bulunmaktadır[1]. Ayrıca zamanlanmış renkli petri ağlarıyla davet işleminin kullanıcı modeli bulunmaktadır.

Zaman başta OBP olmak üzere birçok haberleşme protokolünde çok önemlidir. Çünkü gerçek zamanda olayın başlaması, devamlılığı ve sona ermesi zaman alır. Zamanlı olmayan ağlarda ise olaylar anlık olup ve biter. OBP'de zamanın bu denli önemli olmasına rağmen zamanlı petri ağları kullanılarak şimdiye kadar OBP'nin modeli çıkarılıp analizi yapılmamıştır.

Ayrıca OBP'nin sadece DAVET işlemi ile ilgili çalışılmış, DAVETsiz işlemi hakkında herhangi bir çalışmaya rastlanılmamıştır. Ayrıca daha önce çıkarılan modellerin performans analizleri yapılmamış ve modellerde ölü işaretlemeler bulunmuştur.

1.4 Petri Ağı Modellenmesinde ve Analizinde Kullanılanın Yazılım Araçlarının Karşılaştırılması

1. CPN Tools : (http://wiki.daimi.au.dk/cpntools/cpntools.wiki)

Desteklediği Petri Ağları : Üst Düzey Petri Ağları, Zamanlanmış Petri Ağları

Bileşenleri: Grafik Editör, Jeton Animasyonu, Hızlı Benzetim, Durum Uzayı, Basit Performans Analizi, Dosya Formatı Değiştirebilme

Ayrıntılar: Zaman birim zaman olarak alınmakta, gerçek zaman kavramları kullanılamamaktadır.

2. Geist3D : (http://www.geist3d.org/)

Desteklediği Petri Ağları : Üst Düzey Petri Ağları, Zamanlanmış Petri Ağları, Yer/Geçiş Ağları

Bileşenleri: **-**

Ayrıntılar: Geist3D 3 boyutlu grafiksel makineleri bütünleşmiş geliştirme çevresiyle beraber modelleme aracıdır. Python(nesne yönelimli, yorumlanabilen, birimsel (modüler) ve etkileşimli bir programlama dili) komut dosyalarını ve petri ağlarının destekler.

3. HPSim: (http://www.winpesim.de/)

Desteklediği Petri Ağları : Zamanlanmış Petri Ağları, Yer/Geçiş Ağları, Stokastik Petri Ağları

Bileşenleri: Grafik Editör, Jeton Animasyonu, Hızlı Benzetim, Basit Performans Analizi

Ayrıntılar: Bu araç petri ağları ile çalışmaya yeni başlayanlar için uygundur. Ancak yer ve geçiş sayısı sınırlıdır. Grafik editörü nesnelerin taşınmasına, istenildiği gibi yerleştirilmesine ve silinmesine izin verir. Henüz gelişmiş analiz yapılamamaktadır.

4. INA :(http://www2.informatik.huberlin.de/~starke/ina.html)

Desteklediği Petri Ağları : Üst Düzey Petri Ağları, Yer/Geçiş Ağları

Bileşenleri : Durum uzayı, Yoğunlaştırılmış Durum Uzayı, Yer sabiti, geçiş sabiti, ağ küçültmek, yapısal analiz, basit performans analizi, gelişmiş performans analizi, Dosya Formatı Değiştirebilme, CTL tabanlı model kontrolü

Ayrıntılar: Grafik özellikleri yok tamamen parametrelere bağlı olarak model tanıtılabiliyor. Ancak zamanlanmış petri ağlarını desteklememektedir.

5. Income Suite : (http://www.synlogic.ch/)

Desteklediği Petri Ağları : Üst Düzey Petri Ağları, Zamanlanmış Petri Ağları,Yer/Geçiş Ağları

Bileşenleri: Grafik Editör, Jeton Animasyonu, Hızlı Benzetim, Basit Performans Analizi, İş Akışı Yönetimi Sistemi

Ayrıntılar: Site ve her türlü kullanım dokümanları Almancadır.

6. ORIS (http://www.stlab.dsi.unifi.it/oris/)

Desteklediği Petri Ağları : Zamanlanmış Petri Ağları

Bileşenleri : Grafik Editör, Jeton Animasyonu, Durum Uzayı, Yoğunlaştırılmış Durum Uzayı, Basit Performans Analizi.

Ayrıntılar: Zengin setleri var ama renkli petri ağlarını desteklememektedir. Ayrıca zaman mantıksal olarak verilmektedir.

7. PetitPetri: (http://scg.unibe.ch/download/petitpetri/)

Desteklediği Petri Ağları : Zamanlanmış Petri Ağları, Yer/Geçiş Ağları

Bileşenleri : Grafik Editör, Hızlı ilk örnek yaratabilme

Ayrıntılar: Renkli Petri Ağlarını desteklememekle birlikte, internet sayfasında kullanımı ile ilgili yeterli bilgi yoktur.

8. PNetLab: (http://www.automatica.unisa.it/)

Desteklediği Petri Ağları : Üst Düzey Petri Ağları, Zamanlanmış Petri Ağları, Yer/Geçiş Ağları

Bileşenleri: Grafik Editör, Durum uzayı, Jeton Animasyonu, Yer sabiti, Geçiş Sabiti, Yapısal Analiz, Basit Performans Analizi, Dosya Formatı Değiştirebilme, Gözcü Kontrol Tabanlı

Ayrıntılar: Renkli Petri Ağları da desteklemektedir. Ancak geçiş ve yer sayıları sınırlıdır.

9. Snoopy:
(www.dssz.informatik.tucottbus.de/index.html?/software/snoopy.html)

Desteklediği Petri Ağları : Zamanlanmış Petri Ağları, Yer/Geçiş Ağları, Stokastik Petri Ağları, Sürekli Petri Ağları

Bileşenleri: Grafik Editör, Jeton Animasyonu, Hızlı Benzetim
Ayrıntılar: Renkli petri ağlarını desteklememektedir, tasarım aşamasındadır.

10. StpnPlay: (http://dce.felk.cvut.cz/capekj/StpnPlay/)

Desteklediği Petri Ağları : Zamanlanmış Petri Ağları, Stokastik Petri Ağları

Bileşenleri: Grafik Editör, Jeton Animasyonu, Hızlı Benzetim, Basit performans analizi, Değiştirilebilir dosya formatı

Ayrıntılar: Renkli petri ağlarını desteklememektedir.

11. TimeNET(http://www.tu-ilmenau.de/fakia/8086.html)

Desteklediği Petri Ağları : Üst Düzey Petri Ağları, Zamanlanmış Petri Ağları, Yer/Geçiş Ağları, Stokastik Petri Ağları

Bileşenleri: Grafik Editör, Jeton Animasyonu, Hızlı Benzetim, Yer sabiti, Geçiş Sabiti, Yapısal Analiz, Basit Performans Analizi, Gelişmiş Performans Analizi

Ayrıntılar: Renkli petri ağlarını desteklememektedir ayrıca model tanımlamaları yapılırken belli fonksiyon kalıpları kullanıldığından, uygulaması ve öğrenilmesi zor bir araçtır.

12. Tina : (http://homepages.laas.fr/bernard/tina/)

Desteklediği Petri Ağları : Zamanlanmış Petri Ağları, Yer/Geçiş Ağları

Bileşenleri : Grafik Editör, Jeton Animasyonu, Durum uzayı, yoğunlaştırılmış durum uzayı, Yer sabiti, geçiş sabiti, durum sınıf uzayı

Ayrıntılar: Renkli petri ağlarını desteklememektedir .

13. TAPAAL : (http://www.tapaal.net/)

Desteklediği Petri Ağları : Zamanlanmış Petri Ağları, Yer/Geçiş Ağları, Hibrit Dinamik Ağları ve Hibrit Nesne Ağları

Bileşenleri: Grafik Editör, Jeton Animasyonu, Gelişmiş performans analizi,değiştirilebilir dosya formatı

Ayrıntılar: Renkli petri ağlarını desteklememektedir, klasik petri ağına zaman kavramı eklenmiştir.

14. Visual Object Net **++:**
(http://www.systemtechnik.tuilmenau.de/~drath/visual_E.htm)

Desteklediği Petri Ağları : Zamanlanmış Petri Ağları, Yer/Geçiş Ağları, Hibrit Dinamik Ağları ve Hibrit Nesne Ağları

Bileşenleri: Grafik Editör, Jeton Animasyonu, Hızlı Benzetim, yapısal analiz, basit performans analizi, nesne hiyerarşisi

Ayrıntılar: Renkli petri ağlarını desteklememektedir.

15. WINSIM : (ftp://cmpe2.emu.edu.tr/SimSystem/)

Desteklediği Petri Ağları : Üst Düzey Petri Ağları, Zamanlanmış Petri Ağları

Bileşenleri: Hızlı Benzetim,basit performans analizi

Ayrıntılar: Renkli petri ağlarını desteklememektedir.

2. ZAMANLANMIŞ RENKLİ PETRİ AĞLARI (ZRPA)

Paralel çalışan sistemlerin geniş kullanım alanlarında zaman önemli bir rol oynamaktadır. Bazı sistemlerin doğru çalışması işlemlerin gerçekleştiği gerçek zamana dayanmaktadır ve farklı tasarımlar sistemin performansına önemli derecede etki etmektedir. Renkli petri ağlarına zamanın eklenmesiyle benzetim tabanlı performans analizi uygulanabilir hale gelmiştir. Performans analizinde ölçümler gecikmeler, iş çıkarma oranı ve görevlerin bekleme süresine göre yapılabilir.

Zamanlanmış renkli petri ağlarında, geçişler gerçek zamanda ateşlenir yani ateşleme zamanı her geçişin olay rengiyle alakalıdır ve ağlarının davranışı durumların ve durum geçişlerinin sırası ile tanımlanır.

Zamanlanmış renkli petri ağlarında değişik renklerin jetonları farklı tipteki kaynakları, yerler şimdiki durumu ve geçişler mevcut hareketleri gösterir. Geçiş oluştuğunda, belli bir gecikme sonrasında jeton harekete geçer. Bu süre geçişlere verilen bir süredir.

ZRPA'da jetonlar zaman bilgisini taşırlar. Bu zaman bilgisi o jetonun kullanıma hazır olduğu zamanı gösterir. Zaman bütün model ya da modelin bir parçası için bütünsel(global) senkronizasyon değeri gibi kullanılır.

Bu zaman bilgisi, geçiş renklerinin etkin olduğu ancak jetonların hazır olmadığı bu nedenle geçişin ateşlemediği durumların olduğunu gösterir. Diğer taraftan belli bir zamanda geçişler hazır ve ateşlenebilirse, zaman bu geçişler ateşlenmeden artmaz. Yani bütün modelin benzetim zamanını artırmak ve dolayısıyla ölü noktaları sırayla uyarmak için belli bir zamana ulaşılmalıdır.

Bir ZRPA modeli sistem zamanının şimdiki değeri olan ve model zamanı adı verilen bir zaman bilgisine sahiptir. Zamanlı geçiş bir gecikme zamanıyla ilişkilidir. Bu gecikme geçiş ateşlenmesindeki jeton üretilmesini düzenler.

Eğer şimdiki zamanda herhangi bir geçiş etkin değil ise sistem zamanı otomatik olarak artar. Bu durumda, minimum etkinleşme zamanı ile bir zorlayıcı eleman seçilir ve sistem zamanı model davranışının kilitlenmemesi için uygun değere çekilir.

Zamanlı geçişlerdeki zaman olasılıklı ya da deterministik olabilir.

RPA'nın birçok uygulaması sistemlerin mantıksal doğruluğunu araştırmak için kullanılır. RPA'ların içine zaman bilgisinin eklenmesi belli bir zaman aralığında sistemin dinamik özelliklerini tanıtmak için olasılık sağlar. RPA'larda kullanılan zaman bütünsel bir saatin tanımlanmasına bağlıdır. Saat değeri model zamanını gösterir ve ayrık olmalıdır. Her jeton bir zaman bilgisi taşır.

ZRPA'lar çoğunlukla zaman içermeyen bir ya da daha fazla renk setlerine sahiptir. Bu renk setlerinden birine sahip jeton her zaman uygundur zamandan bağımsızdır.

Zamanlandırılmış Petri ağı, zamansız(klasik) Petri ağı modeline zaman ifadesinin eklenmesiyle elde edilir. Zamanlandırılmış Petri ağı yaklaşımı ile gerçek zamanlı sistemleri modellemek, analiz etmek, kontrol etmek mümkündür ve uygulama alanında daha gerçek bir sonuç üretirler.

Petri ağları bölümünde yapılan açıklamalarda verildiği gibi, petri ağlarında durumlar yerler ile olaylar geçişler ile ve durumlar ile olaylar arasındaki ilişkiler oklar ile gösterilmektedir. Petri ağı modeli ayrık olaylı sistemi davranışsal olarak incelemeye olanak vermektedir[26]. Yapılacak olan analizde sistemin hangi durumlara varacağı ve yaşanacak olası kilitlenmeler gözlenebilmektedir. Fakat gerçek zamanlı sistemlerde bu inceleme yetersiz kalmaktadır. Günümüzde bir sistem için performans tanımı, sadece görevin tamamlanması olarak değil, en kısa sürede görevin tamamlanması olarak yapılmaktadır. Bu nedenle Petri ağı yapısının içine zaman bilgisi eklenerek "Zamanlanmış Petri Ağı" yapısına geçilmiştir[4]. Zamanlanmış petri ağları, klasik Petri ağı yapısına göre gerçek zamanlı bir sistem tanımlaması ve analizi yapma imkânı vermektedir.

2.1 Zamanlı Renkli Petri Ağlarında Kullanılan Tanımlamalar

Zamanlı renkli petri ağlarında kullanılan tanımlamalar aşağıda verilmektedir. CPN Tools programında da bu tanımlamalar kulanılmaktadır.[13]

Tanım-1) S boş olmayan bir küme, T ise zaman değerleri kümesi olarak tanımlansın. S üzerinde zamanlı kümelerin fonksiyonu ($tm: SxT \rightarrow N$), $tm(s,t) \in N$ olmak üzere $\forall(s,t) \in SxT$ için s elemanının olay sayısı şu toplamla gösterilir:

$$tm(s) = \sum_{t \in T} tm(s,t) \quad \textbf{(2.1)}$$

Bu değer $\forall s \in S$ için anlamlıdır. $tm(s)$ s'in tm'deki görünme sayısıdır. s elemanı için zaman bilgileri bir liste halinde verilirse;

$$tm[s] = [t_1, t_2, \dots, t_{tm(s)}] \quad \textbf{(2.2)}$$

elde edilir. $\forall\ 1 \leq i < tm(s)$ için $t_i \leq t_{i+1}$ şartı sağlanmalıdır. tm zamanlı kümeleri gösteriyor olmasına rağmen büyüklükler şu şekilde tanımlanır:

1. $\forall s \in S: s \in tm \ \Leftrightarrow tm(s) > 0$ **(2.3)**

2. $|tm| = \sum_{s \in S} tm(s)$ **(2.4)**

Eğer $|tm| = \infty$ ise tm'de sonsuza gider. S'teki bütün zamanlı kümeleri S_{TMS} olarak gösterilir. Küme yoksa ϕ_{TMS}ile gösterilir ve şu şekilde tanımlanır.

$\forall(s,t) \in SxT$ için $\phi_{TMS}(s,t) =$ 0'dır. **(2.5)**

Tanım-2) S üzerindeki zamanlı kümeler ve T zaman değeri olmak üzere zaman bilgilerinde toplama, çarpma çıkarma işlemleri şu şekilde yapılır.

$s \in S$ için zaman bilgisi listesi şu şekilde olsun:

$$tm[s] = [t_1, t_2, \dots, t_{tm(s)}], \quad \textbf{(2.6)}$$

$$tm_1[s] = [t_1^1, t_2^1, \dots, t_{tm_1(s)}^1], \quad \textbf{(2.7)}$$

$$tm_2[s] = [t_1^2, t_2^2, \dots, t_{tm_2(s)}^2] \quad \textbf{(2.8)}$$

Buna göre aşağıdaki eşitlikler geçerlidir:

1. $\forall 1 \leq i \leq tm_1(s)$ için $tm_1[s] \leq_{[T]} tm_2[s] \Leftrightarrow tm_1(s) \leq tm_2(s)$ ve $t_i^1 \geq t_i^2$'dir. **(2.9)**

2. $t \in T$ için $t \geq t_1$, $tm[s]-_T t$ tarih bilgisi listesidir.

$t_i \leq t$ de i nin en büyük değeri için,

$$tm[s]-_T t = \left[t_1, t_2, t_3, \dots, t_{i-1}, t_{i+1}, \dots, t_{tm(s)}\right] \quad \textbf{(2.10)}$$

3. $tm_1[s] \leq_{[T]} tm_2[s]$ iken zaman bilgisi listesi şu şekildedir:

$$tm_2[s] -_{[T]} tm_1[s] = (\left(\left([t_1^2, t_2^2, \dots, t_{tm_2(s)}^2] -_T t_1^1\right) -_T t_2^1\right) \dots -_T t_{tm_1(s)}^1) \quad \textbf{(2.11)}$$

4. $t \in T$ için zaman bilgisi listesi şu şekildedir.

$$tm[s]_{+t} = [t_1 + t, t_2 + t, \dots, t_{tm(s)} + t] \quad \textbf{(2.12)}$$

Tanım-3) S üzerinden zamanlı kümeler için ve T zaman değerleri için toplama, çıkarma, çarpma $tm, tm_1 \; ve \; tm_2$ zamanlı kümeler için şu şekildedir.

1. $\forall (s,t) \in SxT: (tm_1 + + + tm_2)(s,t) = tm_1(s,t) + tm_2(s,t)$ **(2.13)**

2. $tm_1 \text{«} = tm_2 \Leftrightarrow \forall s \in S: tm_1[s] \leq_{[T]} tm_2(s)$ **(2.14)**

3.$tm_1 \text{«} = tm_2$ olduğu zaman zamanlanmış kümeler şu şekilde tanımlanabilir.

$$\forall s \in S : (tm_2 - - - tm_1)(s) = \; tm_2(s) - tm_1(s) \quad \textbf{(2.15)}$$

$$\forall s \in S : (tm_2 - - - tm_1)[s] = \; tm_2[s] -_{[T]} tm_1[s] \quad \textbf{(2.16)}$$

4. $t \in T$ için zamanlanmış multisetler şu şekilde tanımlanabilir.

$$\forall s \in S : tm_{+t}(s) = tm(s) \; ve \; tm_{+t2}[s] = tm[s]_{+t} \quad \textbf{(2.17)}$$

Tanım-4) Zamanlanmış renkli petri ağlar 9'lu ifade ile tanımlanabilir.

$RPA_T = (P, T, A, V, C, G, E, I, \sum)$

1. P sonlu sayıda yerler kümesi

2. T sonlu sayıda geçişler kümesi ($P \cap T = \emptyset$)

3. Yönlendirilmiş okların kümesi $A \subseteq P \; xT \; \cup TxP$

4. $\sum$ boş olmayan renk setleri kümesi. Her renk seti zamanlı ya da zamansız olabilir.

5. V, $\forall v \in V \; için \; Type[v] \in \sum$ olmak üzere türlernmiş değerler kümesidir.

6. $C: P \rightarrow \sum$, renk kümelerinin fonksiyonudur ve her yerdeki renk setini gösterir. C(p) zamanlı ise p yeri de zamanlıdır.

7. $G: T \rightarrow EXPR_v$ güvenlik fonksiyonudur. $Type[G(t)] = Bool$ olmak üzere her t geçişi için güvenliği gösterir.

8. $E: A \rightarrow EXPR_v$ okların ifadesini gösteren fonksiyondur. Her ok için gösterilir.

p yeri zamansız ise $Type[E(a)] = C(p)_{MS}$

p yeri zamanlı ise $Type[E(a)] = C(p)_{TMS}$

9. $I: P \rightarrow EXPR_{\emptyset}$ her p yeri için başlangıç fonksiyonunu göstermektedir.

p yeri zamansız ise $Type[I(p)] = C(p)_{MS}$

p yeri zamanlı ise $Type[I(p)] = C(p)_{TMS}$

Tanım-5) Zamanlanmış renkli petri ağlarında kullanılan bazı kavramlar şu şekildedir.

1. $\forall p \in P$ yeri için jetonların işaretleme fonksiyonu şu şekildedir:

- p zamansız ise $M(p) \in C(p)_{MS}$
- p zamanlı ise $M(p) \in C(p)_{TMS}$

2. Zamanlı işaretleme çifti, M işaretlemeyi, $t^* \in T$ bütünsel saati göstermek üzere (M, t^*)'dir.

3. $\forall p \in P$ ve $M_0(p) = I(p)$ olmaz üzere ilk işaretleme çifti $(M_0, 0)$ şeklinde gösterilir.

Tanım-6) $Y \in BE_{MS}$ adımı (M, t^*) işaretlemesinde t' zamanında ancak ve ancak aşağıdaki şartları sağlarsa etkindir.

1. $\forall (t, b) \in Y: G(t)\langle b \rangle$ **(2.18)**

2.Zamansız $p \in P$ yerleri için $\overset{++}{_{ÇK}}\sum_{(t,b)\in Y} E(p,t)\langle b \rangle \ll= M(p)$ **(2.19)**

3.Zamanlı $p \in P$ yerleri $\overset{++}{_{ÇK}}\sum_{(t,b)\in Y} (E(p,t)\langle b \rangle)_{+t'} \ll= M(p)$ **(2.20)**

4. $t^* \leq t'$ **(2.21)**

5. Yukarıdaki şartların sağlanması durumunda t'en küçük değerindedir.

Y adımı (M, t^*)'de t' zamanında etkinken, yine bu zamanda oluşur ve (M', t') işaretlemesine ulaşılır.

6. Zamansız $p \in P$ yerleri için

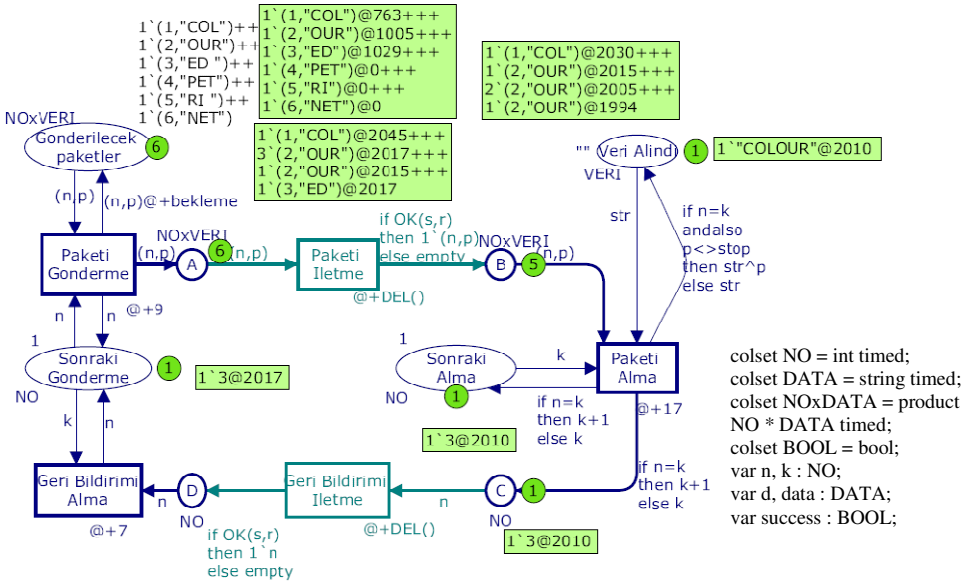
$M'(p) = (M(p) - - \overset{++}{_{ÇK}}\sum_{(t,b)\in Y} E(p,t)\langle b \rangle) + + \overset{++}{_{MS}}\sum_{(t,b)\in Y} E(p,t)\langle b \rangle$ **(2.22)**

7. Zamanlı $p \in P$ yerleri için

$$M'(p) = (M(p) -- {}^{++}_{ÇK}\sum_{(t,b)\in Y} E(p,t)\langle b\rangle_{+t'}) ++ {}^{++}_{MS}\sum_{(t,b)\in Y} E(p,t)\langle b\rangle_{+t'} \quad \textbf{(2.23)}$$

Y adımındaki (t,b) zorlayıcı elemandır. ÇK çoklu kümeleri göstermektedir.

İşaretlemeler, etkin ve oluşan adımları gösteren zamanlı kümeleri tanıtmak için Şekil 2.1'deki modeli kullanacağız.



Şekil 2.1 : Tanımlamaları Örneklemek için Kullanılan Model (Bütünsel saat = 1993BZ)

Aşağıdaki tanımlamaları yapabilmek için NOxDATA renk kümesinde tanımlı zamanlı kümeleri (tm_A, tm_B, tm_{PA}) kullanacağız.

Tanım 1'e göre zamanlı kümeler şu şekildedir:

$tm_A =$ 1‘(1,"COL")@2045 +++ 3‘(2,"OUR")@2017 +++
1‘(2,"OUR")@2015 +++ 1‘(3,"ED ")@2017 +++

$tm_B =$ 1‘(1,"COL")@2030 +++ 1‘(2,"OUR")@2015 +++
2‘(2,"OUR")@2005 +++ 1‘(2,"OUR")@1994

$tm_{PA} =$ 1‘(1,"COL")@2010 **(2.24)**

t zaman bilgisi, s jeton rengi olmak üzere zamanlı küme tm_B şu şekilde gösterilir :

$$tm_B(s,t) = \begin{cases} 1 \text{ eğer } (s,t) = ((1,\text{"COL"}),2030) \\ 1 \text{ eğer } (s,t) = ((2,\text{"OUR"}),2015) \\ 2 \text{ eğer } (s,t) = \big((2,\text{"OUR"}),2005\big) \\ 1 \text{ eğer } (s,t) = \big((2,\text{"OUR"}),1994\big) \\ 0 \qquad\qquad\qquad \text{aksi takdirde} \end{cases}$$

(2.25)

Bu görünüm +++ operatörünün yaptığı işle aynıdır.

$tm_B(s,t)$ tanımından anlaşılacağı gibi $(2,\text{"OUR"})$ katsayısı bir kez 2015, bir kez 1994 ve iki kez 2005 zamanlarında toplam 4 kez görülmüştür. Bu katsayısının zaman bilgisi listesi şu şekilde gösterilir:

$$tm_B[(2,\text{"OUR"})] = [1994,2005,2005,2015] \quad \textbf{(2.26)}$$

Bu listeden de anlaşılacağı gibi tm zamanlı kümesindeki s elemanının zaman bilgisi listesinin uzunluğu ile tm'deki s olaylarının sayısı aynıdır.

$|tm|$ tm zamanlı kümesinin boyutunu gösterir. tm_A için$|tm| = 6$, tm_B için $|tm| =$ 5'dir.

Örnek için kullanılan tm_A ve tm_B'nin toplamları şu şekilde gösterilir:

$$(tm_A +++ tm_B)(s,t) = \begin{cases} 1 \text{ eğer}(s,t) = \big((1,\text{COL}),2045\big) \\ 1 \text{ eğer}(s,t) = \big((1,\text{COL}),2030\big) \\ 3 \text{ eğer}(s,t) = \big((2,\text{OUR}),2017\big) \\ 2 \text{ eğer}(s,t) = \big((2,\text{OUR}),2015\big) \\ 2 \text{ eğer}(s,t) = \big((2,\text{OUR}),2005\big) \\ 1 \text{ eğer}(s,t) = \big((2,\text{OUR}),1994\big) \\ 1 \text{ eğer}(s,t) = \big((3,\text{ED }),2017\big) \\ 0 \qquad\qquad\qquad \text{aksi takdirde} \end{cases}$$

(2.27)

1. eşitliğe göre, 2 tane zamanlı küme düşünelim;

$tm_{PA} = 1‘(1,"\text{COL}")@2010$

$tm_B = 1‘(1,"\text{COL}")@2030 +++ 1‘(2,"\text{OUR}")@2015 +++$
$2‘(2,"\text{OUR}")@2005 +++ 1‘(2,"\text{OUR}")@1994$ **(2.28)**

Bu iki küme için ve (2,"OUR") katsayısı için,

$tm_{RP}((2,"\text{OUR}")) = 1$

$tm_{RP}\big((2,\text{OUR})\big) = [2010]$

$tm_B\big((2,\text{OUR})\big) = 4$

$tm_B\big((2,\text{OUR})\big) = [1994, 2005, 2005, 2015]$ **(2.29)**

$s \in S \ için\ tm_1(s) \leq tm_2(s)$ şartına göre $tm_{PA}\big((2,"\text{OUR}")\big) \leq tm_B\big((2,"\text{OUR}")\big)$,

$tm_1[s] \leq_{[T]} tm_2[s]$ şartı için $tm_{PA}[(2,"\text{OUR}")] \leq_{[T]} tm_B[(2,"\text{OUR}")]$ sağlanmaktadır, çünkü $tm_{PA}[s]$'daki 2010 zaman bilgisi, $tm_{PBA}[s]$'deki 1994 zaman bilgisine karşılık düşer ve bu değerden daha büyüktür.

Küçük zamanlı kümelerin daha büyük zaman bilgisi olması geçişlerin etkin durumda olduğunu gösterir.

Örnekte kullanılan tm_B ve tm_{PA} ve çıkarma işlemi şu şekilde gösterilir:

$$\text{tm}_\text{B}(\text{s}) = \begin{cases} 1 \text{ eğer s} = (1,"\text{COL}") \\ 4 \text{ eğer s} = (2,"\text{OUR}") \\ 0 \text{ aksi takdirde} \end{cases}$$

$$\text{tm}_\text{PA}(\text{s}) = \begin{cases} 1 \text{ eğer s} = (2,"\text{COL}") \\ 0 \text{ aksi takdirde} \end{cases}$$

$$(\text{tm}_\text{B} --- \text{tm}_\text{PA})(\text{s}) = \begin{cases} 1 \text{ eğer s} = (1,"\text{COL}") \\ 3 \text{ eğer s} = (2,"\text{OUR}") \\ 0 \text{ aksi takdirde} \end{cases}$$

(2.30)

Eğer zaman bilgileri listesinde çıkarma yapmak için tm_2'deki her zaman bilgisinden tm_1'deki bu zaman bilgisine en yakın olan değer çıkartılır. Örneğin;

$$\mathrm{tm_B}[(2,\text{"OUR"})] = [1994, 2005, 2005, 2015]$$

$$\mathrm{tm_{PA}}[(2,\text{"OUR"})] = [2010]$$

$$(\mathrm{tm_B} --- \mathrm{tm_{PA}})[(2,\text{"OUR"})] = [1994,2005,2015] \quad \textbf{(2.31)}$$

olur.

2010 zaman bilgisine en yakın zaman bilgisi 2005 olduğu için listeden çıkarılmıştır.

Zaman kümesindeki çıkarma da şu şekilde gösterilebilir:

$$\mathrm{tm_B} --- \mathrm{tm_{PA}} = 1\text{‘}(1,\text{"COL"})@2030 +++ 1\text{‘}(2,\text{"OUR"})@2015 +++ 1\text{‘}(2,\text{"OUR"})@2005 +++ 1\text{‘}(2,\text{"OUR"})@1994 \quad \textbf{(2.32)}$$

Tanım 6'ya göre PaketiAlma yerinin (M,1993) zamanlı işaretlemesinde zorlayıcı elemanı:

$$b_{PA} = \langle n = 2, d = \mathrm{OUR}, k = 3, data = "COLOUR"\rangle \quad \textbf{(2.33)}$$

olsun. Bundan sonraki adım şu şekilde gösterilir.

$$PA = 1'(PaketiAlma, b_{PA}) \quad \textbf{(2.34)}$$

2010 zamanında bu geçişin etkin olduğunu düşünelim. B giriş yeri için,

$$\overset{++}{\underset{ÇK^{(t,b)\in Y}}{\sum}} (E(p,t)\langle b\rangle)_{+2010} = (E(B, PaketiAlma)\langle b_{PA}\rangle)_{+2010}$$

$$= (1'(2,"OUR")@0)_{+2010}$$

$$= 1'(2,\mathrm{OUR})@2010 \quad \textbf{(2.35)}$$

gösterilir.

2010 zamanında $(M, 1993)$ işaretlemesinde PA geçişi oluştuğunda B ve C yerlerinin işaretlemeleri şu şekildedir;

$$
\begin{aligned}
M'(B) &= ((1'(1,\text{"COL"})@2030 +++ 1'(2,\text{"OUR"})@2015 +++ \\
&\quad 2'(2,\text{"OUR"})@2005 +++ 1'(2,\text{"OUR"})@1994 --- \\
&\quad 1'(2,\text{"OUR"})@0_{+2010} +++ (\emptyset_{TMS})_{+2010} \\
&= ((1'(1,\text{"COL"})@2030 +++ 1'(2,\text{"OUR"})@2015 +++ \\
&\quad 2'(2,\text{"OUR"})@2005 +++ 1'(2,\text{"OUR"})@1994 --- \\
&\quad 1'(2,\text{"OUR"})@2010 +++ \emptyset_{TMS} \\
&= ((1'(1,\text{"COL"})@2030 +++ 1'(2,\text{"OUR"})@2015 +++ \\
&\quad 2'(2,\text{"OUR"})@2005 +++ 1'(2,\text{"OUR"})@1994 --- \\
M'(C) &= ((1'3@2010 --- (\emptyset_{TMS})_{+2010}) +++ (1'3@17)_{+2010} \\
&= (1'3@2010 --- \emptyset_{TMS} +++ 1'3@2027 \\
&= (1'3@2010 +++ 1'3@2027
\end{aligned}
\tag{2.36}
$$

3. OTURUM BAŞLATMA PROTOKOLÜ'NÜN İŞLEMLERİ

OBP, dört katmandan oluşmuştur, bunların her biri birçok fonksiyonu gerçekleştirir. Bunlar aşağıdan yukarı, söz dizimi ve şifreleme(syntax and encoding), iletim(transport), işlem(transaction) ve işlem kullanıcısı(transaction user-TU) katmanlarıdır. Söz dizimi ve şifreleme katmanı formatı OBP mesajının yapısını belirtir. İletim katmanı, OBP mesajlarını alt katmandan alır ve geri gönderir. OBP iletim katmanın en üstündeki katman işlem katmanıdır. Her işlem istekleri gönderen bir kullanıcı işlemi ve cevap veren sunucu işleminden oluşur. En üst katman İşlem Kullanıcısı OBP işlemlerini yaratır ve yok eder ve işlem katmanı tarafından desteklenen servisleri kullanır.

En önemli katman işlem katmanıdır. İstekleri karşılamak, mesajlara cevap vermek, iletim ortamı güvenilir olmadığında mesajları yeniden iletmek ve süre aşımının üstesinden gelmekle sorumludur. Bu işlem bir kullanıcı birde sunucudan oluşmaktadır. Kullanıcı tarafı kullanıcı işlemi, sunucu tarafı sunucu işlemi olarak adlandırılmaktadır. Kullanıcı tarafı istek, sunucu tarafı cevap gönderir. Kullanıcıdan sunucuya bir istek veya sunucudan kullanıcıya cevap OBP mesajları ile olur. Her istek mesajı sunucuda özel bir işleme başvurmak için bir metot taşır (DAVET veya ALINDI). Her cevap kabulü, reddi veya OBP isteğinin tekrarlı olması için durum kodu gösterir.(Çizelge 3.1)

Çizelge 3.1: OBP Cevap Mesajı Kodları

Cevap	Görevi
1xx	İstek mesajlarının alındığını ve işleme konulduğunu bildirir.
2xx	Arama işleminin başarılı bir şekilde alındığını, anlaşıldığını ve kabul edildiğini belirtir.
3xx	İsteğin yerine getirilebilmesi için yeni faaliyetlerin yapılması gerektiğini bildirir.
4xx	İstekteki yanlış söz dizimini belirtir.
5xx	Sunucu isteği cevaplayamadığını belirtir.
6xx	İsteği cevaplayacak sunucu yok.
Not: xx 00 ile 90 arasındaki sayılar anlamına gelir	

OBP işlemi boyunca gönderilen ilk mesaj her zaman istek mesajıdır ve bir istek tipine dayanır, bu işlem DAVET işlemi(eğer DAVET isteği ise) veya DAVETsiz işlemi(DAVET ve ALINDI dışındaki istek) olarak bilinir. DAVETsiz işlemi oturumu yönetir ve sonlandırır, DAVET işlemi oturum kurar.

OBP karşılıklı yanıt gerektiren bir protokoldür. Elemanlar arasındaki iletişim bağımsız mesajlar serisi ile gerçekleşir. Özellikle, bir OBP işlemi tek bir istek ve bu isteğe karşı sıfır ya da daha fazla koşullu cevap ve bir ya da daha fazla final cevabı içeren cevaptan oluşur. DAVET işlemi, sadece 2xx cevabının final cevap olmadığı ALINDI mesajını içerir. Eğer cevap 2xx ise ALINDI bu işlemin bir parçası değildir. Bu ayrımın sebebi KTİ'ye 200(OK) cevaplarının iletilmesindeki önemidir.

Kullanıcı işleminin amacı, kullanıcının içinde bulunduğu elemandan (TU–transaction user, UA veya durumlu vekil) istek almak ve sunucu tarafına bu isteği güvenli bir şekilde iletmektir. Buna ek olarak DAVET isteği durumunda, kullanıcı işlemi 2xx cevabını kabul eden herhangi bir final cevabı için ALINDI isteği oluşturur.

Özet olarak, sunucu işleminin amacı iletim katmanının istekleri almak ve bunları TU'ya iletmektir. Sunucu işlemi ağdan gelen yeniden gönderme isteklerini filtreler. Sunucu işlemi, TU'dan cevapları kabul eder ve ağdan iletim katmanına iletir. DAVET işlemi durumunda, ALINDI isteklerini cevapları kabul eden final cevabı için absorbe eder.

3.1 Kullanıcı İşlemi

Kullanıcı işlemi durum makinesinin yönetim fonksiyonlarını sağlar[20]. TU kullanıcı işlemi ile basit bir ara yüz ile iletişim kurar. TU yeni bir oturum yaratmak istediği zaman, kullanıcı işlemi yaratır ve bir IP adresi, port ve isteğin gönderildiği OBP isteğini geçirir. Kullanıcı işlemi kendi durum makinesinin başlatır. TU'dan kullanıcı işleminin geçerli cevapları geçer.

Kullanıcı işleminin TU tarafından iletilen istek metoduna dayanan 2 tür durum makinesi vardır. Biri DAVET isteklerini işletir, bu tip makine DAVET kullanıcı işlemi olarak adlandırılır. Diğer tip işlem ise DAVET ve ALINDI dışındaki istekleri işler ve bu da DAVETsiz işlem olarak adlandırılır.

ALINDI için kullanıcı işlemi bulunmamaktadır. TU, ALINDI göndermek isterse, iletim için onu direk iletim katmanına gönderir.

DAVET işlemi diğer metotlardan farklılık gösterir. Normal olarak, DAVET isteğine cevap vermek için bir insan girişi gerekmektedir. Cevap göndermek için olan uzun gecikmeler 3-yollu anlaşma ile karşı gelmektedir. Başka bir deyişle, diğer metotların istekleri hızlıca tamamlaması beklenir. DAVETsiz işlem anlaşmaya dayanmadığı için TU DAVETsiz isteklerine hemen cevap vermelidir.

3.1.1 Davet İçeren Kullanıcı İşlemi

DAVET işlemi 3 yollu anlaşma içerir. Kullanıcı işlemi DAVET gönderir, sunucu işlemi cevap gönderir ve kullanıcı işlemi ALINDI gönderir. Güvensiz iletimler için kullanıcı işlemi T1 zamanında başlayan ve her yeniden iletimden sonra 2 katına çıkan bir zaman aralığında istekleri yeniden gönderir. T1 gidiş dönüş zamanı olarak değerlendirilir(RTT-round trip time) ve ilk değeri 500 ms'dir. Bütün işlem zamanlayıcıları T1 zamanı ile tanımlanmıştır. T1'i değiştirmek bu değerleri etkiler. Güvenilir ortamlarda istek yeniden gönderilmez. 1xx cevabını aldıktan sonra herhangi bir yeniden iletim tamamen durur ve kullanıcı başka cevapları bekler. Sunucu işlemi güvenilir olarak iletmediği 1xx cevabı gönderebilir. Güvensiz sistemler için cevap periyodik olarak yeniden iletilir, güvenilir sistemler için bu bir kez gönderilir. Kullanıcı işleminde alınan her final cevabı için işlem ALINDI gönderir bunun amacı cevaplar için yeniden iletimleri sona erdirmektir.

Bir DAVET işlemi 4 duruma sahiptir: *Çağırma*(Calling), *İlerleme*(Proceeding), *Tamamlandı*(Completed) ve *Sonlandı*(Terminated). TU bir DAVET isteği ile yeni bir kullanıcı işlemi yarattığında önce ilk olan *Çağırma* durumuna girmelidir. Kullanıcı işlemi işlem isteğini iletir.

Davet içeren kullanıcı işleminin sağlaması gereken durumlar Ek A.1'de anlatılmıştır.

3.1.2 Davet İçermeyen Kullanıcı İşlemi

DAVETsiz işlemi ALINDI kullanımı yapmaz. Güvensiz ortamlar için istekler T1 zamanında başlayan ve T2 zamanına ulaşana kadar 2 katına çıkarılarak belirlenen aralıklarla yeniden iletilir. Eğer geçici bir cevap alınırsa, T2 zaman aralığında güvensiz ortamlar için yeniden iletim devam eder.

Sunucu işlemi yalnız yeniden iletim isteği aldığında geçici ya da son cevabı yeniden iletir. Bu istek yeniden iletimlerin geçici cevaptan sonra devam etmesi gerekliliğini getirir.

DAVET işleminden farkı, DAVETsiz işlemi 2xx cevabı için özel bir yöntem içermez. Bunun sonucu sadece DAVETsiz işlemine giden tek 2xx cevabının KTİ'ye gönderilmesidir.

Bir DAVETsiz işlemi 4 duruma sahiptir: *Deneniyor*(Trying), *İlerleme*(Proceeding), *Tamamlandı*(Completed) ve *Sonlandı*(Terminated). Bir istekle yaratılan kullanıcı işlemi TU başladığında *Deneniyor*(Trying) durumuna gelir.

Davet içermeyen kullanıcı işleminin sağlaması gereken durumlar Ek A.2'de anlatılmıştır.

3.2 Sunucu İşlemi

Sunucu işlemi isteklerin TU'ya ulaştırılması ve cevapların güvenli iletimi ile sorumludur. Bu şekilde bir durum makinesi vardır. Bu işlem çekirdek tarafından bir istek alındığında yaratılır ve işlem bu cevabı yönetmekle sorumludur.

Kullanıcı işlemi gibi durum makinesi alınan isteğin DAVET isteği olmasına dayanır.

3.2.1 Davet İçeren Sunucu İşlemi

Sunucu işlemine bir cevap geldiğinde *İlerleme* durumuna geçer. TU son cevabı ürettiğini bilmedikçe geçici cevabını üretmek zorundadır. Bu geçici cevap ağ trafiğinden kaçmak için istek yeniden iletimlerini hızla sıraya sokmak için gereklidir.

Cevap TU'ya iletilmek zorundadır. TU geçici cevapların her birini sunucu işlemine gönderir. Sunucu işlemi *İlerleme* durumunda iken iletim için bunların her biri iletim katmanına gönderilir. İletim katmanı tarafından güvenilir olarak gönderilmezler yani yeniden iletilmezler ve sunucu işleminin durum değiştirmesine sebep olmazlar. *İlerleme* durumunda iken TU'dan alınan son geçici cevap yeniden iletim için iletim katmanına gönderilmek zorundadır.

Davet içeren sunucu işleminin sağlaması gereken durumlar Ek A.3'te anlatılmıştır.

3.2.2 Davet İçermeyen Sunucu İşlemi

Deneniyor durumu başlangıç durumudur ve DAVET veya ALINDI'dan başka bir istek iletir. İstek TU'ya iletilir. Bu durumda bazı isteklerin yeniden iletimi göz önüne alınmaz.

Davet içermeyen sunucu işleminin sağlaması gereken durumlar Ek A.4'te anlatılmıştır.

3.3 OBP İşlemlerinde Kullanılan Zamanlayıcıların Birbirleriyle İlişkileri

Kullanıcı işlemi 3 tane zamanlayıcı kullanır. *Zamanlayıcı* B en büyük değere ayarlanır ve kullanıcı işlemi sunucu tarafından geçici *Çağırma* durumunda bekler. Bu zamanlayıcı iletim ortamında işlem aktifken kullanılır.

Zamanlayıcı A sadece ortam güvensiz olduğunda, DAVET isteklerinin yeniden iletimi için kullanılır. *Zamanlayıcı* D ortam güvenilir olmadığında kullanılır, güvenli ortamda 0 s, güvensiz ortamda 32 s'dir.

Bu üç zamanlayıcı arasında A ve B ilişkilidir. RFC'de bu ilişki açıkça belirtilmemiştir. Zamanlanmış renkli petri ağları kullanarak mümkün olayları bulabilmek için önce bu ilişkiye açıklık getirmemiz gerekir. RFC'ye göre, zamanlayıcı A'nın ilk değeri T1 500 ms'dir ve kullanıcı ve sunucu arasındaki tahmin edilen bir gidiş-dönüş hareketi kadardır. Her zaman *Zamanlayıcı* A ateşlenir, bir önceki değeri kendi değerine eşit olduğunda yeniden başlatılır. *Zamanlayıcı* B 64*T1 kadardır. Bu nedenle *Zamanlayıcı* B ateşlendiğinde zamanlayıcı A'nın T1, 2*T1, 4*T1,8*T1, 16*T1, 32*T1 zamanlarında 6 kez oluştuğunu görürüz. B ateşlenmeden cevabı almada ya da göndermede hata oluşursa 6'dan daha az olur. *Zamanlayıcı* A ateşlenirse DAVET isteği yeniden iletilir. Çünkü kullanıcı işlemi yaratıldığında, orijinal DAVET istediği iletilir ve *Zamanlayıcı* B ateşlenmeden en fazla 7 tane DAVET isteği gönderilir.

Sunucu işlemi de 3 tane zamanlayıcıya sahiptir. G,H,I zamanlayıcıları. *Zamanlayıcı* H hem güvenilir hem güvensiz ortamlar için kullanılır. ALINDI alınmadan önce sunucu işleminin *Tamamlandı* durumunda beklerken en yüksek değerine ayarlanır. G zamanlayıcısı sadece güvensiz ortamlarda kullanılır ve 300-699 cevaplarının yeniden iletilmesini kontrol eder.

Tamamlandı durumuna geçildiğinde, H zamanlayıcısı 64*T1 değerine, G zamanlayıcısı T1 değerine ayarlanır. G zamanlayıcısı 2'den daha fazla ateşlendiğinde $2^{i-2} * T1$ ve T2'nin en küçük değerleri ile ayarlanır. Bundan dolayı, H zamanlayıcısının süresi dolmadan önce, G zamanlayıcısı 2*T1, 4*T1, 8*T1, 8*T1, 8*T1, 8*T1, 8*T1, 8*T1,8*T1, 8*T1 aralıklarında 10 kez oluşur. H zamanlayıcısından önce eğer bir ALINDI alınırsa veya iletim hatası oluşursa, G zamanlayıcısı 10'dan daha az kere oluşmuş olur. I zamanlayıcısı güvenilir ortamlar için 0, güvensiz ortamlar için 5 s'ye ayarlanır.

DAVETsiz kullanıcı işleminde 3 tane zamanlayıcı kullanılmaktadır. E,F,K zamanlayıcıları. *İlerleme* duruma girdiğinde kullanıcı işlemi 64*T1 zamanı ile ateşlenen F zamanlayıcını başlatır. İstek iletim için iletim katmanına gönderilir.

Güvensiz ortamlar kullanılırsa kullanıcı işlemi T1 zamanında ateşlenen E zamanlayıcısını başlatır. Aynı durumda iken E zamanlayıcısı ateşlenirse zamanlayıcı başlangıç durumuna döner ama bu zamanlayıcının değeri MIN(2*T1, T2) olur. Zamanlayıcı tekrar ateşlenirse MIN(4*T1, T2) değerine döner. Bu yöntem yeniden iletimlerin T2 zaman aralığında üstel artmasıyla devam eder. T2'nin ilk değeri 4s'dir. DAVETsiz sunucu işleminin eğer hemen cevap dönmezse isteğe ne kadar sürede cevap verdiğini gösterir. T1 ve T2'nin varsayılan değerleri 500 ms, 1 s, 2 s, 4 s, 4 s, 4 s, etc. dir. Kullanıcı işlemi *Deneniyor* durumunda olduğunda F zamanlayıcısı ateşlenirse bu zaman aşımını TU'ya haber verir ve *Sonlandı* durumuna girer. Kullanıcı işlemi *Tamamlandı* durumuna bir kerelik girdiğinde güvensiz ortamlar için K zamanlayıcısı T4 zamanında ateşlenebilecek şekilde ayarlanır. Bu güvenli sistemler için 0 s'dir. *Tamamlandı* durumunda ek cevapların yeniden iletimi saklanır. Bu durumda K zamanlayıcı ateşlenirse kullanıcı işlemi *Sonlandı* durumuna geçer.

Davet içermeyen sunucu işleminde sadece J zamanlayıcısı kullanılır. Sunucu işlemi *Tamamlandı* durumunda iken güvensiz ortamlar için 64*T1 değerinde J zamanlayıcısı ayarlanır bu değer güvenilir ortamlar için 0 s'dir.

4. CPN TOOLS PROGRAMININ ÖZELLİKLERİ

Modelleme ve doğrulamanın pratik uygulaması, büyük modellerin yapımı ve yönetimini destekleyen bilgisayar yazılım araçlarına dayanmaktadır.

CPN Tools programı renkli petri ağlarının oluşturulması, benzetimi, durum uzayı analizi ve performans analizi için kullanılmaktadır. Bu program zamanlı ve zamansız hiyerarşik petri ağlarını desteklemektedir.

Windows XP, Vista ve Linux işletim sistemleri ile uyumludur. Dünya genelinde 140 farklı ülkeden 8000 kullanıcıya sahiptir ve web sayfasından indirmek ücretsizdir.

Renkli petri ağlarının grafiksel gösterimi ile işlem yapılır.

CPN Tools programı söz dizimini kontrol eder ve hata mesajlarını kullanıcıya gösterir. Bu kontrol, model geliştirmesi boyunca yapılır, yani model parçalı olarak da çalıştırılabilir. Asıl çıktı benzetim adımında oluşturulur. Bu adımda modelde verilen her türlü uygulanabilir olaylar görülebilir.

CPN Tools programında fonksiyonel programlama dili olan Standart ML (Meta-Language) kullanılmaktadır. Bu programlama dili, veri tiplerinin kolayca tanımlanması, verinin idare edilmesinin belirtilmesini ve parametreştirilebilir model oluşturulmasını sağlar.

CPN Tools programının arayüzü ve modelleme dili Ek A.5'de ayrıntılı olarak verilmiştir.

4.1 Petri Ağının CPN Tools Programı ile Analizi

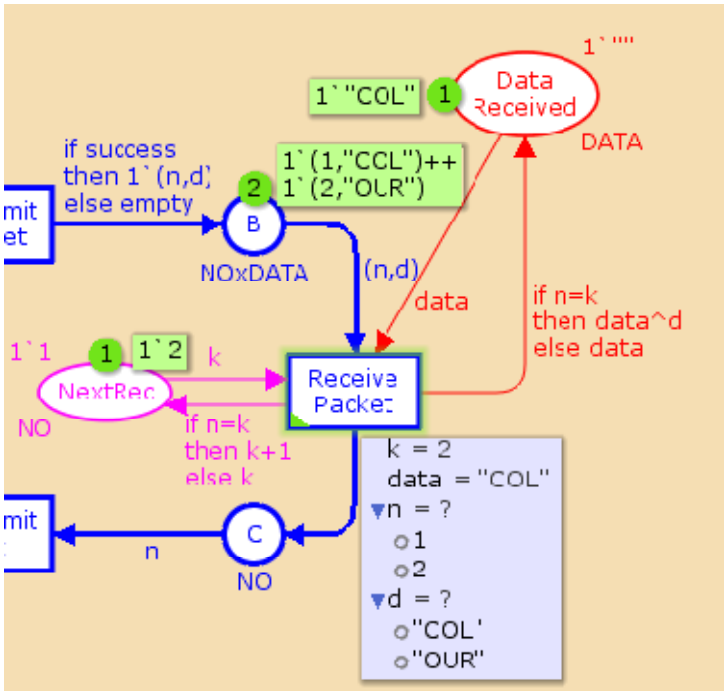
4.1.1 Benzetim

CPN programı 2 tip benzetimi destekler, interaktif ve otomatik. Interaktif benzetimde, kontrol kullanıcıdadır ve benzetim adımı uygulanabilir olaylarla tek tek oluşturulur. Bu adımlar grafik ortamda da izlenebilmektedir.

Otomatik benzetimde ise, kullanıcı kaç adım benzetim yapılacağını, durması gereken yerleri belirtir ve benzetim kullanıcı etkisi olmadan çalıştırılır. Ekranda sadece son sonuç gösterilir. Ancak benzetim sonuçları kaydedilir ve bu sonuçlar otomatik benzetimdaki her adımı içerir.

Simülatör büyük sistemler için birçok veri kullanmaktadır. Simülatör modelin boyutundan bağımsız olarak her saniyede koşturduğu adımları sağlamak için bölge özelliklerini kullanır. Böylece büyük sistemleri de simüle edebilir.

CPN Tools programı benzetim boyunca ulaşılan işaretlemeler, etkin geçişler ile ilgili bilgi almak amacıyla grafiksel benzetim geri beslemesi kullanır. Yerlerdeki küçük daire içindeki sayılar jeton sayısını, kutu ise jetonların rengini göstermektedir. Geçişlerin etrafındaki yeşil çerçeve ise o geçişin etkin olduğunu gösterir.



Şekil 4.1 : CPN Tools Programında Kullanılan Benzetim Araçları

4.1.2 Durum Uzayı Analizi

Durum uzayı Petri Ağı'nın erişebileceği durumları ve bütün durum değişikliklerini hesaplamaktır. Durum uzayı erişilebilirlik ağacı, erişilebilirlik grafı ya da ortaya çıkma grafı olarak da literatürde geçmektedir.

CPN Tools programı gelişmiş durum uzayını destekler. Durum analizinde, ani artış probleminin etkisi gelişmiş modeller ile azaltılmaya çalışılmıştır. CPN Tools programı sistemin özelliklerini durum uzayı kullanarak analiz etmek için birçok özelliği destekler.

Sistemin davranışını takip etmek için ilk adım standart davranış özelliklerinin cevabını içeren durum uzayı raporunu yaratmaktır. Bu özellikler ölü noktaların bulunma durumu, yerlerdeki en büyük ve en küçük jeton sayısıdır.

Sistem yaratılmasının ilk bölümlerinde, durum uzayı raporunda tasarım hatalarına oldukça rastlamaktadır. Çünkü bu rapor otomatik olarak yaratılmaktadır.

CPN Tools programı ile bütün ya da parçalı olarak durum uzayı analizi yapılabilir. Durum uzayı özelliklerini kolayca anlamak için benzetim için önemli olmayan ama durum uzayı için önemli olan birçok söz dizimi kısıtlaması yapılmalıdır. Örneğin sayfadaki yerler ve geçişler bireysel isimlere sahip olmadıkça ve okların tanımlamaları yapılmadıkça durum uzayı analizi çalıştırılamaz. Bunun için CPN Tools programında söz dizimi devamlı kontrol edilir.

4.1.3 Performans Analizi

Veri toplama ile birleşik otomatik benzetim yoluyla benzetim tabanlı performans analizi desteklenmektedir.

Bu analizin temel fikri, sistemin performansı ile ilgili bilginin toplanması süresinde modelin aşırı uzun benzetimlerinin sayısını yönetmektir. Veri kuyruğun boyutunu, paketlerin gecikmesini ve çeşitli elemanların yükünü içerir. Verinin toplanması veri toplama monitörüne(data collector monitors) dayanır. Bu monitörler kullanıcı tarafından özelleştirilebilir ve otomatik benzetimin bireysel adımları süresinde toplanır. Veri günlük(log) dosyalarına(örneğin elektronik çizelge) işlem sonrası için yazılmaktadır. Performans raporu ortalama, standart sapma ve güven aralığı gibi verileri toplamak için özet bilgiler içermektedir. Benzetim tabanlı performans analizi toplu benzetimi kullanır çünkü modelin parametre uzayını keşfetmek mümkündür ve güvenilir sonuçlar almak için birçok benzetimi yönetebilmektedir.

Performans analizi hesaplanırken, iki ayrı olay arasındaki geçişlerde ne kadar zaman harcandığı ile ilgilenilir.

Performans ölçümleri benzetim boyunda hesaplanan ve toplanan nümerik verilere dayanır. CPN Tools programınızda veri toplama monitörleri bu amaç için kullanılmaktadır. Farklı amaçları olan birçok farklı veri toplama monitörleri bulunmaktadır. CPN Tools programında standart olan ve daha önceden tanımlanmış veri toplama monitörleri bulunmaktadır.

5. OLUŞTURULAN MODELLER ve ANALİZLERİ

5.1 OBP'nin DAVET İşleminin Modeli ve Analizi

Önerilen model EK C.1'de verilmiştir.

5.1 OBP'nin DAVET İşleminin Modeli ve Analizi

Önerilen model kullanıcı ve sunucu işlemlerinin durum makineleri göz önüne alınarak yapılmıştır. Tanımlamalar aşağıda belirtildiği gibidir:

```
colset  CEVAP      =  with r100 | r101 | r2xx | r3xx;
colset  ISTEK      =  with DAVET | ALINDI| ZamanAsimi;
colset  DURUMS  =  with bos | ilerlemeT | ilerlemeS | onaylandiS | tamamlandiS |
                      sonlandiS;
colset  zaman_cons = int timed;
colset  DURUMC  = with cagirma | ilerleme| tamamlandi| sonlandi;
colset  Cevap         = subset CEVAP with [r101, r2xx, r3xx];
colset  CEVAPQ    = list CEVAP;
colset  INT           = int with 0..1;
var     cevapQ :CEVAPQ;
var     zaman_passS, zaman_calS: zaman_cons;
val     n = 3;
var     a,f: INT;
var     zaman_pass, zaman_cal: zaman_cons;
val     T1=500;
var     re : Cevap;
var     b: INT;
var     cev : CEVAP;
var     sc : DURUMC;
var     ss : DURUMS;
var     istek: ISTEK;
fun    OT(t:zaman_cons):bool=if t>64*T1 then true else false
```

Durumun sonuna S konulması onun sunucu için olduğu anlamına gelmektedir. Çizelge 5.1'de ise yerler ve geçişlere yüklenen anlamlar belirtilmiştir.

Zamanlayıcıların nasıl çalıştığını göstermek amacıyla model güvensiz ortamlar için oluşturulmuştur.

Modelin sol tarafı kullanıcı işlemidir. DURUMC renk kümesi ile tanımlanan kullanıcı yeri bu işlemin durumunu gösterir ve ilk değeri *Çağırma* durumudur. Kullanıcı yerine 5 tane geçiş bağlıdır. "İstek Gönderildi" geçişi iletim için DAVET isteklerinin OBP iletim katmanına nasıl gönderildiğini modeller.

Kullanıcı yeri *Çağırma* durumunda iken ve henüz iletim katmanında DAVET isteği yok iken etkindir. "Cevap Alındı" geçişi kullanıcı işleminin cevaplarını nasıl aldığını ve ALINDI'ları nasıl gönderdiğini modeller, kullanıcı işlemi sonlanmadığında ve bir cevap geldiğinde etkindir. Kullanıcı işlemi r3xx mesajını aldığında, ALINDI mesajını OBP iletim katmanına gönderir ve *Tamamlandı* durumuna geçer. r100, r101 veya r2xx mesajları alındığı zaman ALINDI gönderilmez, kullanıcı r100 veya r101 aldığında *İlerleme* durumuna, r2xx aldığında *Sonlandı* durumuna geçer.

A ve B zamanlayıcıları DAVET isteği gönderilmeden önce başlamaz. Kullanıcı *Çağırma* durumundayken ve DAVET isteği gönderildiği zaman etkin olur.

D zamanlayıcısı kullanıcı yerinin ne kadar süre *Tamamlandı* durumunda kaldığını kontrol eder. Bu geçişin ateşlenmesiyle kullanıcı yeri *Sonlandı* durumuna geçer.

"Kullanıcı İletim Hatası" geçişi ise iletim hatalarını kontrol eder. Eğer kullanıcı işlemi *Tamamlandı* ya da *Çağırma* durumundayken etkin olabilir. Bu geçiş ateşlenirse kullanıcı işlemi *Sonlandı* durumuna geçer. Kullanıcı işlemi *Tamamlandı* durumundayken, SIP iletim katmanında hata meydana gelirse, ALINDI sunucu tarafına ulaşamaz ve bu mesaj İstekler yerinin kuyruğuna konur.

Modelin sağ tarafı sunucu işlemini gösterir. Sunucu yerine 5 geçiş bağlıdır. "Sunucu" geçişi işlemin durumunu gösterir ve DURUMS renk kümesi ile tanımlanmıştır. Sunucu işleminin ilk değeri boştur bu da bize sunucu işleminin DAVET isteği geldiği zaman hemen yaratılabileceğini gösterir. "İstek Alındı" geçişi DAVET veya ALINDI isteğini karşılar. Sunucu boş durumundayken DAVET isteği gelirse *İlerlemeT* durumuna geçer yani sunucu işlemi yaratılır. Böylece Kullanıcı işleminin cevapları nasıl gönderdiğini modelleyen "Cevap Gönderildi" geçişi etkin olur ve Cevaplar yerine r100 mesajını koyar ve sunucu işlemi *İlerlemeS* durumuna geçer. Cevap Gönderildi yeri tekrar etkin olursa r101 geçici cevabı ya da r2xx veya r3xx son cevabı Cevaplar yerine gönderilir. r2xx gönderilirse *SonlandıS* durumuna, r3xx gönderilirse *TamamlandıS* durumuna geçer ve r101 gönderilirse kendi durumunda kalır.

Çizelge 5.1: Modelde Kullanılan Düğümlerin Anlamları

Kullanıcı İşlemi	
Yerler	**Anlamları**
Kullanıcı	DAVET Kullanıcı İşleminin durumlarını modeller.
DAVET Gönderildi	İletilen ya da yeniden iletilen DAVET isteklerini sayar.
A	A zamanlayıcısını modeller.
B	B zamanlayıcısını modeller.
Geçişler	**Anlamları**
İstek Gönderildi	İletim için DAVET isteklerinin OBP iletim katmanına nasıl gönderildiğini modeller.
Cevap Alındı	Kullanıcı işleminin cevaplarını nasıl aldığını ve ALINDI'ları nasıl gönderdiğini modeller.
Zamanlayıcı D	D zamanlayıcısını modeller.
Kullanıcı İletim Hatası	Kullanıcı iletim hatasını kontrol eder.
A ve B Zamanlayıcıları	A ve B zamanlayıcılarının kullanımını modeller.
Uygulama Katmanı	
Yerler	**Anlamları**
İstekler	Kullanıcı tarafından sunucu tarafına iletilen istekleri modeller.
Cevaplar	Sunucu tarafından müşteri tarafına iletilen cevapları modeller.
Geçişler	**Anlamları**
İstek	İstekler yerinden gelen hataları modeller.
CevapQ	Cevaplar yerinden gelen hataları modeller.
Sunucu İşlemi	
Yerler	**Anlamları**
Sunucu	Kullanıcı işleminin durumlarını modeller.
R3xx	G zamanlayıcısı ateşlendiği zaman yeniden iletilen r3xx mesajının sayısını kaydeder.
G	G zamanlayıcısının kullanımını modeller.
H	H zamanlayıcısının kullanımını modeller.
Geçişler	**Anlamları**
İstek Alındı	DAVET veya ALINDI isteğini karşılar.
Cevap Gönderildi	Kullanıcı işleminin cevapları nasıl gönderdiğini modeller.
Sunucu İletim Hatası	Kullanıcı iletim hatalarını kontrol eder.
Zamanlayıcı I	I zamanlayıcısının kullanımını modeller.
G ve H Zamanlayıcıları	H ve G zamanlayıcılarının kullanımını modeller.

Sunucu *TamamlandıS* durumundayken, ALINDI mesajı alırsa sunucu "Cevap Alındı" geçişi sayesinde *SonlandıS* durumuna geçer. "Cevap Alındı" geçişi sunucu *TamamlandıS* durumuna geçtikten sonra mesajın gelmemesini sağlar.

H zamanlayıcısı sunucu *Tamamlandı* durumuna geçmesi ile etkin olur. Bu geçiş ateşlenirse sunucu işlemi *SonlandıS* durumuna geçer. Sunucu İletim Hatası kullanıcı iletim hatalarını kontrol eder. Sunucu *İlerlemeS* ya da *TamamlandıS* durumundayken iletim katmanına cevap gönderildiği zaman etkin olur. Ateşlenirse cevap Cevaplar kuyruğuna konar.

Modelin orta kısmı ise iletim ortamıdır. Kullanıcı tarafından sunucu tarafına iletilen istekleri modelleyen İstekler yeri, Sunucu tarafından müşteri tarafına iletilen cevapları modelleyen Cevaplar yeri, İstekler yerinden gelen hataları modeller. "İstek" geçişi, Cevaplar yerinden gelen hataları modelleyen "CevapQ" geçişinden oluşmaktadır.

5.1.1 DAVET Modelinin Durum Uzayı Analizi

Analizde önem taşıyan yerler EK B.1'de verilen durum uzayı raporunda görülen ölü noktalar(deadlocks) ve canlı noktaların(livelocks) varlığıdır. Modelde istenmeyen davranış belirtilen ancak gerçekleşmeyen olayların var olmasıdır. Başka bir deyişle, etkin olmayan hiçbir geçişin olmamasıdır. Davet modelinde sadece iki olayın ölü noktalarının bulunması beklenir. Biri DAVET işleminin sona erdiği yani ya kullanıcının ya da sunucunun *Sonlandı* durumuna geçtiği zamandır. Diğer olay ise "CevapQ" ve "İstek" geçişlerinin etkin olduğu yani mesajların kaybolduğu işaretlemelerdir. Diğer ölü işaretlemeler ölü nokta olarak belirtilir. Canlı noktalar ise modelin gerçekleştirdiği durumları gösterir.

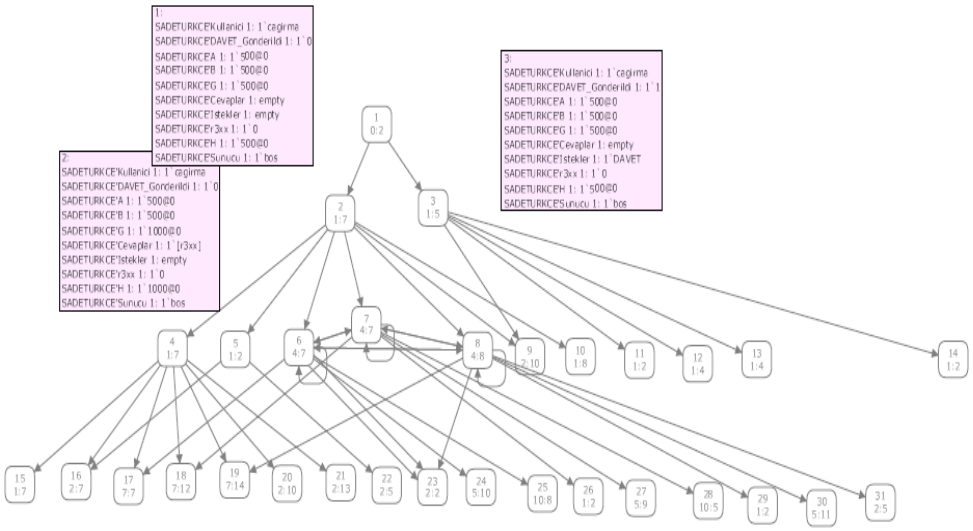
DAVET modelimizin istenen özellikleri sağlayıp sağlamadığını durum uzayı raporunda yer almaktadır. Bu rapor, modelin her türlü durumunu analiz ettiği için, raporun oluşturulması bazen saatler sürmektedir. 12 saat sonunda oluşan durum analizinde 93429 düğüm, 355903 ok hesaplanmıştır. Bu nedenle ilk 5 dk'da oluşan durumlar göz önüne alınmıştır. Buna parçalı durum analizi denmektedir. Bu 5 dk değeri deneyerek elde edilmiştir. Sistem belli bir süre sonunda kararlı bir hale gelmektedir. Birçok denemenin sonunda en fazla 5 dk'da modelin kararlı hale geldiği görülmüştür. Parçalı durum uzayı raporunda 5 dk'da 12687 düğüm, 67828 ok hesaplanmıştır. Bunlardan 9232 düğüm, 43403 ok canlıdır yani durum uzayında bir döngü içerirler. Bu grafiğe Sıkıca Bağlı Elementler(Strongly Connected Components-SCC) adı verilmektedir. Ölü işaretlemeler 6933 tanedir ve bunlar 9999,9998,9997,9996,9995… işaretlemeleridir.

Çizelge 5.2'de görüldüğü gibi ölü işaretlemeler iki işlemden birinin sonra erdiği durumlardır. Bunlarda beklenen durumlardır.

Çizelge 5.2 : Davet Modeli Ölü İşaretlemeler

Düğüm	**9999**	**9998**	**9997**	**9996**
Kullanıcı	tamamlandı	tamamlandı	tamamlandı	sonlandı
İstekler	[ALINDI, ALINDI]	[DAVET, DAVET,ALINDI]	[DAVET, ALINDI,ALINDI]	[DAVET, ALINDI,ALINDI]
Sunucu	ilerlemeS	ilerlemeS	sonlandıS	1`sonlandıS
Cevaplar	[r101, r101, r3xx]	[r3xx]	[r3xx]	[r3xx]
Davet Gönderildi	1`1	1`1	1`1	1`1
A	1`1000@500	1`1000@500	1`1000@500	1`1000@500
B	1`1000@500	1`1000@500	1`1000@500	1`1000@500
G	1`1000@0	1`1000@0	1`1000@0	1`1000@0
H	1`1000@0	1`1000@0	1`1000@0	1`1000@0
R3xx	1`0	1`0	1`0	1`0

Canlı noktaları görebilmek için durum uzayını çizdirebiliriz.



Şekil 5.1 : DAVET Modelinin Durum Uzayı Grafı

Durum uzayı ve SCC grafı arasındaki fark düğümler arasında olan döngülerden kaynaklanmaktadır. Bu döngüler canlı nokta olarak değerlendirilmez. Örneğin 7 ve 8 numaralı düğümler kendi içinde döngü oluşturmuşlardır.

CPN Tools programı sayesinde benzetimlikte(simülatör) gördüğümüz işaretlemelerin durum uzayında hangi düğüme karşılık geldiğini, durum uzayında gördüğümüz düğümün simülatördeki işaretlemelerini görebiliriz. Bunun için "State Space" aracındaki "SimToStateSpace" ve "StateSpaceToSim" seçeneklerini kullanabiliriz. Kullanıcının DAVET isteğini gönderdiği İstekler yerinin DAVET i aldığı ve "İstek Alındı" geçişinin etkin olduğu işaretleme 11700 numaralı düğüme karşı düşmektedir. Bu da bize her düğümün işaretlemesini gösterir ve modelin anlaşılmasında kolaylık sağlar.

Model oluşturulurken sınırsız durum uzayı probleminden kaçınılmak için bir takım sınırlamalar yapılmıştır. Sınırsız durum uzayı problemi modelin durum uzayı hesaplanmak istediğinde, sonlanmayan bir döngüye girerek sonucun çıkarılamamasıdır. Model ilk oluşturulduğunda bu problemle devamlı karşılaşılmıştır. Sonunda modelden tek tek düğümler çıkarılarak problemin kullanılan sayı değerlerinin yüksek olmasından kaynaklandığı anlaşılmıştır. INT renk kümesi tanımlanması sadece 0 ve 1 sayıları ile, "A ve B Zamanlayıcıları" geçişi [a>=1 andalso a<2] ile "G ve H Zamanlayıcıları" geçişi b<2 ile ve "Sunucu" geçişinin *İlerlemeT* veya *İlerlemeS* ve n değerine ulaşıncaya kadar ile sınırlandırılmıştır. Böylece sınırsız durum uzayı problemi çözülmüştür.

Model oluşturulduktan sonra "G ve H Zamanlayıcıları" geçişinin hiç etkin olmadığı görülmüştür. r3xx yerine başlangıç değeri verilmediği için olduğu anlaşılmış ve problem giderilmiştir.

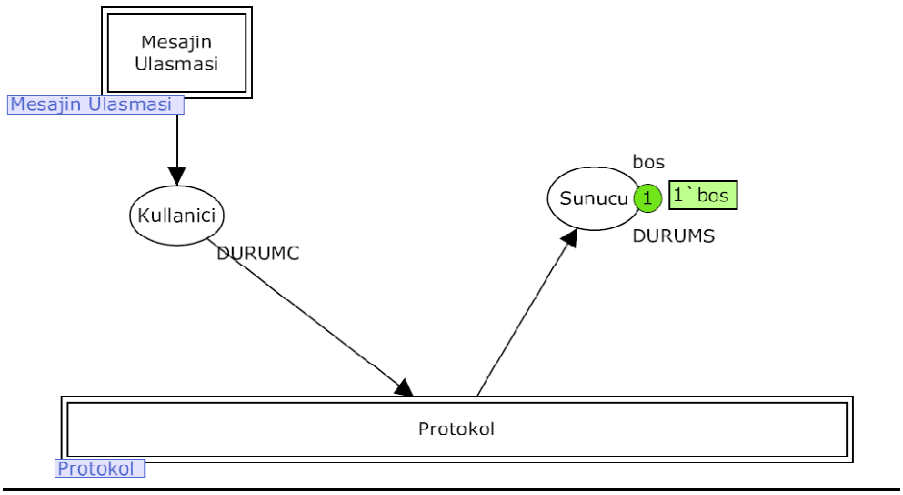
Modelin uygunluk özelliği incelenirse, "A ve B zamanlayıcıları", "Cevap Alındı", "G ve H zamanlayıcıları", "İstek", "İstek Alındı", "İstek Gönderildi", "Kullanıcı İletim Hatası", "Sunucu İletim Hatası", "Zamanlayıcı D", "CevapQ" geçişlerinin "No Fairness" olduğunu görürüz. Bunun anlamı bu geçişlerin sınırsız kez etkin olabileceğini göstermektedir. Ancak biz bu geçişlere sınırlamalar koyarak bunun önüne geçebildik.

Modelin sınırlılık özellikleri incelenirse, EK B.1'deki rapora göre, "A", "B", "G ve H", "Kullanıcı", "Sunucu", "r3xx" yerlerinde en fazla 1 en az 1 kez jeton olmalıdır. "İstekler" yerinde en fazla 5 en az 0, "Cevaplar" yerinde ise en fazla 8 en az 0 jeton olmalıdır. Örneğin "Cevaplar" yerinde 5 tane jeton şu şekilde bulunur: 5`[]++, 1`[r100]++, 4`[r101]++, 4`[r2xx]++, 8`[r3xx]. Buradan da anlaşılacağı gibi "Cevaplar" yerinde aynı anda 8 tane jeton bulunabilmektedir.

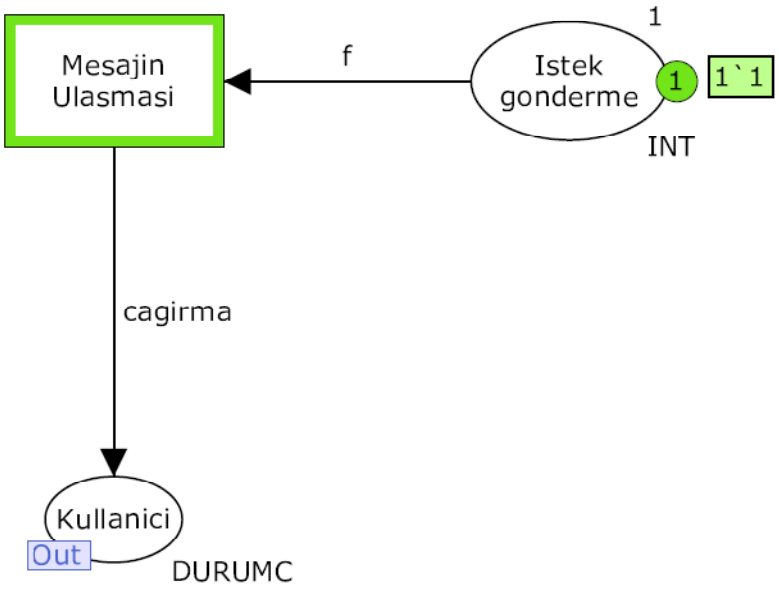
5.1.2 DAVET Modelinin Performans Analizi

Performans analizinin yapılabilmesi için sistemin genel bir modüle ihtiyaç vardır. CPN Tools programı bu modül üzerinden performans analizi yapabilmektedir.

Bizim modelimizin modülü şekil 5.2'te verildiği gibidir.



Şekil 5.2 : Performans Analizi için Modelin En Üst Modeli



Şekil 5.3 : Performans Analizi için Modelin Mesajın Ulaşması Modeli

Mesajın Ulaşması modülünün amacı protokol modelinin başlatılmasını sağlamaktır. Kullanıcı işleminin TU'su gibi çalışmaktadır. Kullanıcı yerini *Çağırma* durumuna geçirir. Performans analizi için bir takım monitörler tanımlanmalıdır.

Bizim modelimiz için kullandığımız monitörler şu şekildedir:

- Marking_size_Protokol'Kullanıcı_1 : Kullanıcı yerindeki jetonları sayar.
- Marking_size_Protokol'Sunucu_1: Sunucu yerindeki jetonları sayar.
- Marking_size_Protokol'Cevaplar_1: Cevaplar yerindeki jetonları sayar.
- Marking_size_Protokol'Istekler_1: İstekler yerindeki jetonları sayar.
- Marking_size_Protokol'DAVET_Gönderildi_1: DAVET Gönderildi yerindeki jetonları sayar.
- Marking_size_Protokol'A_1: A yerindeki jetonları sayar.
- Marking_size_Protokol'B_1 : B yerindeki jetonları sayar.
- Marking_size_Protokol'r3xx_1 : r3xx yerindeki jetonları sayar.
- Marking_size_Protokol'G_1: G yerindeki jetonları sayar.
- Marking_size_Protokol'H_1: H yerindeki jetonları sayar.

- Count_trans_occur_Protokol'Istek_Kayıp_1: “İstek Kayıp” geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'Cevap _Kayıp_1: “Cevap Kayıp” geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'G ve H_zamanlayıcıları_1: “G ve H Zamanlayıcıları” geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'Cevap_Gönderildi_1: “Cevap Gönderildi” geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'Istek_Alindi_1: “İstek Alındı” geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'Zamanlayıcı_I_1: “Zamanlayıcı I” geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'Sunucu _İletim_Hatası_1: “Sunucu İletim Hatası” geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'Istek_Gönderildi_1: “İstek Gönderildi” geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'Kullanıcı_İletim_Hatası_1: “Kullanıcı İletim Hatası” geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'Cevap_Alındı_1 : “Cevap Alındı” geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'Zamanlayıcı_D_1: “Zamanlayıcı D” geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'A ve B_zamanlayıcıları_1: “A ve B Zamanlayıcıları” geçişinin oluşma sayısını sayar.
- Group 1: İsteklere karşı verilen cevap oranıdır. Arada mesaj kaybı olup olmadığına bakar.

50 adımlık benzetim sonuçlarının zamanlı performans analizleri şu şekildedir.

Çizelge 5.3 : Zamanlı Performans Analizi Sonuçları

Timed statistics								
Name	**Count**	**Sum**	**Avrg**	**Min**	**Max**	**First Time**	**Last Time**	**Time Interval**
Marking_size_Protokol'A_1	2	22500	1.000000	1	1	0	22500	22500
Marking_size_Protokol'B_1	2	22500	1.000000	1	1	0	22500	22500
Marking_size_Protokol'Cevaplar_1	49	0	0.000000	0	1	0	22500	22500
Marking_size_Protokol'DAVET__Gönderildi_1	3	22500	1.000000	1	1	0	22500	22500
Marking_size_Protokol'G_1	26	22500	1.000000	1	1	0	22500	22500
Marking_size_Protokol'H_1	26	22500	1.000000	1	1	0	22500	22500
Marking_size_Protokol'İstekler_1	4	0	0.000000	0	1	0	22500	22500
Marking_size_Protokol'Kullanıcı_1	5	22500	1.000000	0	1	0	22500	22500
Marking_size_Protokol'Sunucu_1	26	22500	1.000000	1	1	0	22500	22500
Marking_size_Protokol'r3xx_1	26	22500	1.000000	1	1	0	22500	22500

50 adımlık benzetim sonuçlarının zamansız performans analizleri şu şekildedir.

Çizelge 5.4: Zamansız Performans Analizi Sonuçları

Untimed statistics					
Name	**Count**	**Sum**	**Avrg**	**Min**	**Max**
Count_trans_occur_Protokol'A_ve_B_zamanlayıcıları_1	0	0	0.000000	0	0
Count_trans_occur_Protokol'Cevap_Alındı_1	0	0	0.000000	0	0
Count_trans_occur_Protokol'Cevap_Gönderildi_1	0	0	0.000000	0	0
Count_trans_occur_Protokol'Cevap__Kayıp_1	1	1	1.000000	1	1
Count_trans_occur_Protokol'G_ve_H__zamanlayıcıları_1	24	24	1.000000	1	1

Çizelge 5.4: Zamansız Performans Analizi Sonuçları(Devam)

Count_trans_occur_Protokol'İstek_Alındı_1	0	0	0.000000	0	0
Count_trans_occur_Protokol'İstek_Gönderildi_1	1	1	1.000000	1	1
Count_trans_occur_Protokol'İstek_Kayıp_1	0	0	0.000000	0	0
Count_trans_occur_Protokol'Kullanıcı_İletim_Hatası_1	1	1	1.000000	1	1
Count_trans_occur_Protokol'Sunucu__İletim_Hatası_1	0	0	0.000000	0	0
Count_trans_occur_Protokol'Zamanlayıcı_D_1	0	0	0.000000	0	0
Count_trans_occur_Protokol'Zamanlayıcı_I_1	0	0	0.000000	0	0
Group_1	50	25	0.500000	0	1

Performans analizlerinden de anlaşılacağı gibi "Cevap Kayıp" geçişi ve "Kullanıcı İletim Hatası" geçişi birer kez ateşlenmiştir. "Kullanıcı" yerinin ateşlenmesi 5 kez gerçekleşmiştir. Bu da bize 5 kere de 1 kez hata oluşabileceği göstermektedir. Ek B.3'te verilen analiz sonuçlarında Group monitorun her adımda 1 ya da 0 değerini gösterdiğini yani kaybolan mesajın sadece bir kez oluştuğunu anlayabiliriz. Kullanıcının gönderdiği 5 tane istekten biri iletim hatasına uğramış diğer 4'ü "İstekler"yerine ulaşabilmiştir.

5.2 OBP'nin DAVETSİZ İşleminin Modeli ve Analizi

Önerilen model EK C.2'deki gibidir.

Model kullanıcı ve sunucu işlemlerinin durum makineleri göz önüne alınarak yapılmıştır. Tanımlamalar aşağıda verildiği gibidir. Durumun sonuna S konulması onun sunucu için olduğu anlamına gelmektedir.

Çizelge 5.5'de ise yerler ve geçişlere yüklenen anlamlar belirtilmiştir.

Zamanlayıcıların nasıl çalıştığını göstermek amacıyla model güvensiz ortamlar için oluşturulmuştur.

Tanımlamalar :

```
colset   CEVAP = with msjyok| r100 | r101 | r2xx | r3xx;
colset   ISTEK   = with istekM;
colset  DURUMS  =  with  deneniyorS|  ilerlemeT  |  ilerlemeS  |  onaylandiS  |
tamamlandiS | sonlandiS;
colset   zaman_cons=int timed;
colset   DURUMK = with deneniyor| ilerleme| tamamlandi | sonlandi;
colset   Cevap = subset CEVAP with [r101, r2xx, r3xx];
colset   ISTEKQ= list ISTEK;
colset   INT = int with 0..1;
var      istekQ:ISTEKQ;
var      zaman_passS, zaman_calS: zaman_cons;
var      a: INT;
var      zaman_pass, zaman_cal: zaman_cons;
val      T1=500;
var      re : Cevap;
var      b: INT;
var      cev : CEVAP;
var      cevap : CEVAP;
var      sc : DURUMC;
var      ss : DURUMS;
var      istek: ISTEK;
fun     OT(t:zaman_cons):bool=if t>64*T1 then true else false;
```

Modelin sol tarafı kullanıcı işlemidir. DURUMK renk kümesi ile tanımlanan kullanıcı yeri bu işlemin durumunu gösterir ve ilk değeri *Deneniyor* durumudur. Kullanıcı yerine 5 tane geçiş bağlıdır. "İstek Gönderildi" geçişi İletim için isteklerinin OBP iletim katmanına nasıl gönderildiğini modeller. "Cevap Alındı" geçişi Kullanıcı işleminin cevaplarını nasıl aldığını modeller. Kullanıcı işlemi sonlanmadığında ve bir cevap geldiğinde etkindir.

"E ve F Zamanlayıcıları" geçişi *Deneniyor* durumundayken aktif durumdadır.

"K Zamanlayıcısı" kullanıcı yerinin ne kadar süre *Tamamlandı* durumunda kaldığını kontrol eder. Bu geçişin ateşlenmesiyle kullanıcı yeri *Sonlandı* durumuna geçer.

Kullanıcı İletim Hatası geçişi ise iletim hatalarını kontrol eder. Eğer kullanıcı işlemi *Tamamlandı* ya da *Deneniyor* durumundayken etkin olabilir. Bu geçiş ateşlenirse kullanıcı işlemi *Sonlandı* durumuna geçer. Kullanıcı işlemi *Tamamlandı* durumundayken, OBP iletim katmanında hata meydana gelirse, cevap sunucu tarafına ulaşamaz ve bu mesaj İstekler yerinin kuyruğuna konur.

Çizelge 5.5: DAVETsiz İşleminin Modelinde Kullanılan Düğümlerin Anlamları

Kullanıcı İşlemi	
Yerler	**Anlamları**
Kullanıcı	DAVET Kullanıcı İşleminin durumlarını modeller.
İstek Gönderildi	İletilen ya da yeniden iletilen DAVET isteklerini sayar.
y	İstek Gönderildi geçişinin aktive olmasını sağlar
E	E zamanlayıcısını modeller.
F	F zamanlayıcısını modeller.
Geçişler	**Anlamları**
İstek Gönderildi	İletim için DAVET isteklerinin OBP iletim katmanına nasıl gönderildiğini modeller.
Cevap Alındı	Kullanıcı işleminin cevaplarını nasıl aldığını modeller.
Kullanıcı İletim Hatası	Kullanıcı iletim hatasını kontrol eder.
K Zamanlayıcısı	A ve B zamanlayıcılarının kullanımını modeller.
E ve F Zamanlayıcısı	F zamanlayıcısını kullanımını modeller.
Uygulama Katmanı	
Yerler	**Anlamları**
İstekler	Kullanıcı tarafından sunucu tarafına iletilen istekleri modeller.
Cevaplar	Sunucu tarafından müşteri tarafına iletilen cevapları modeller.
Geçişler	**Anlamları**
İstek Kayıp	İstekler yerinden gelen hataları modeller.
Cevap Kayıp	Cevaplar yerinden gelen hataları modeller.
Sunucu İşlemi	
Yerler	**Anlamları**
Sunucu	Kullanıcı işleminin durumlarını modeller.
Geçişler	**Anlamları**
İstek Alındı	DAVET veya ALINDI isteğini karşılar.
Cevap Gönderildi	Kullanıcı işleminin cevapları nasıl gönderdiğini modeller.
Sunucu İletim Hatası	Kullanıcı iletim hatalarını kontrol eder.
Zamanlayıcı J	J zamanlayıcısının kullanımını modeller.

Modelin sağ tarafı sunucu işlemini gösterir. Sunucu yerine 4 geçiş bağlıdır. “Sunucu“ geçişi işlemin durumunu gösterir ve DURUMS renk kümesi ile tanımlanmıştır. Sunucu işleminin ilk değeri *Deneniyor* durumudur. “İstek Alındı” geçişi istekleri karşılar. *Deneniyor* durumunda iken TU sunucu işlemine geçici bir cevap iletirse sunucu işlemi *İlerleme* durumuna geçer. Böylece Kullanıcı işleminin cevapları nasıl gönderdiğini modelleyen “Cevap Gönderildi” geçişi etkin olur.

"J Zamanlayıcısı" geçişi, sunucu *Tamamlandı* durumuna geçmesi ile etkin olur. Bu geçiş ateşlenirse sunucu işlemi *SonlandıS* durumuna geçer.

"Sunucu İletim Hatası" geçişi, kullanıcı iletim hatalarını kontrol eder. Sunucu *İlerlemeS* ya da *TamamlandıS* durumundayken iletim katmanına cevap gönderildiği zaman etkin olur. Ateşlenirse cevap Cevaplar kuyruğuna konar.

Modelin orta kısmı ise iletim ortamıdır. Kullanıcı tarafından sunucu tarafına iletilen istekleri modelleyen "İstekler" yeri, Sunucu tarafından müşteri tarafına iletilen cevapları modelleyen "Cevaplar" yeri, İstekler yerinden gelen hataları modeller. "İstek Kayıp" geçişi, "Cevaplar" yerinden gelen hataları modelleyen "Cevap Kayıp" geçişinden oluşmaktadır.

5.2.1 DAVETsiz Modelinin Durum Uzayı Analizi

DAVETsiz modelimizin istenen özellikleri sağlayıp sağlamadığını CPN Tools programının çıktı olarak verdiği, EK B.4'de gösterilen, durum uzayı raporundan anlayabiliriz.

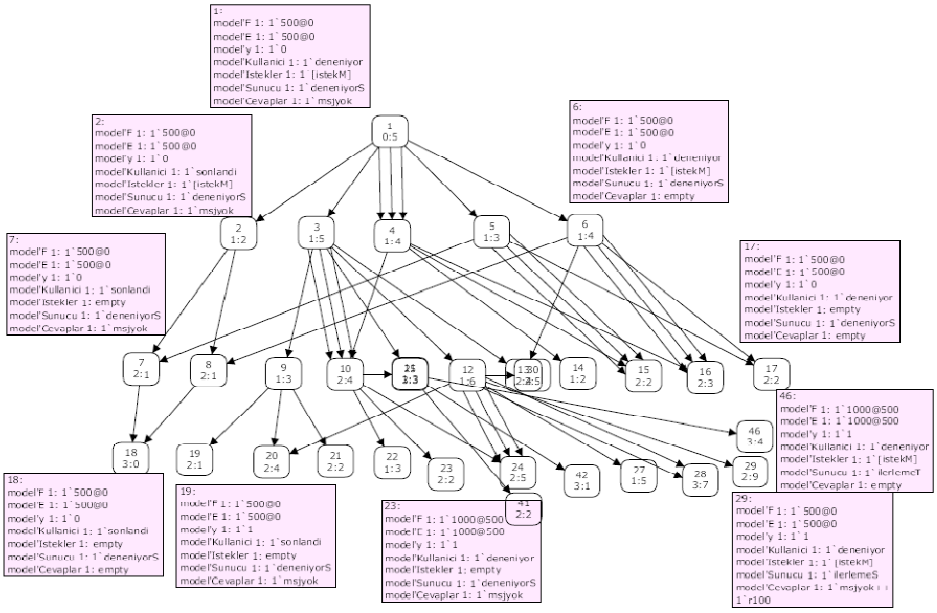
Modelin fazla durumu olduğundan parçalı durum uzayı raporu alınmıştır. 5 dk'da 43038 düğüm, 279558ok hesaplanmıştır. Bunlardan 36618 düğüm, 264586 ok canlıdır yani durum uzayında bir döngü içerirler. Ölü işaretlemeler 322 tanedir ve bunlar 739, 72, 60, 435, 43038,... işaretlemeleridir. Çizelge 5.6'da ölü işaretlemeler sırasında düğümlerin durumu gösterilmiştir.

Görüldüğü gibi ölü işaretlemeler herhangi bir işlemin sonlandığı durumlardır. Bunlarda beklenen durumlardır.

Çizelge 5.6 : Davet Modeli Ölü İşaretlemeler

Düğüm	**739**	**435**	**60**	**43038**	**72**
Kullanıcı	sonlandı	sonlandı	sonlandı	sonlandı	sonlandı
İstekler	[]	[]	[]	[istekM]	[]
Sunucu	sonlandıS	sonlandıS	deneniyorS	tamamlandıS	deneniyorS
Cevaplar	[]	[]	[]	[1`r100++ 135`r101++ 1`r2xx]	[]
y	1`1	1`1	1`0	1`1	1`1
E	1`1000@500	1`500@0	1`1000@500	1`1000@500	1`1000@500
F	1`1000@500	1`500@0	1`1000@500	1`1000@500	1`1000@500

Canlı noktaları ve işaretlemeleri görebilmek için durum uzayı grafını çizdirebiliriz.



Şekil 5.4 : DAVETsiz Modelinin Durum Uzayı Grafı

Bunun için "State Space" aracındaki "SimToStateSpace" ve "StateSpaceToSim" seçeneklerini kullanarak simülatörde gördüğümüz işaretlemelerin durum uzayında hangi düğüme karşılık geldiğini, durum uzayında gördüğümüz düğümün simülatördeki işaretlemelerini görebiliriz. Kullanıcının istek gönderdiği işaretleme 1 numaralı düğümdür,çünkü DAVETsiz işlemi DAVET işleminden sonra oturumun yönetilmesi için kullanıldığından ilk durumun istek gönderme durumu olması gerekmektedir.

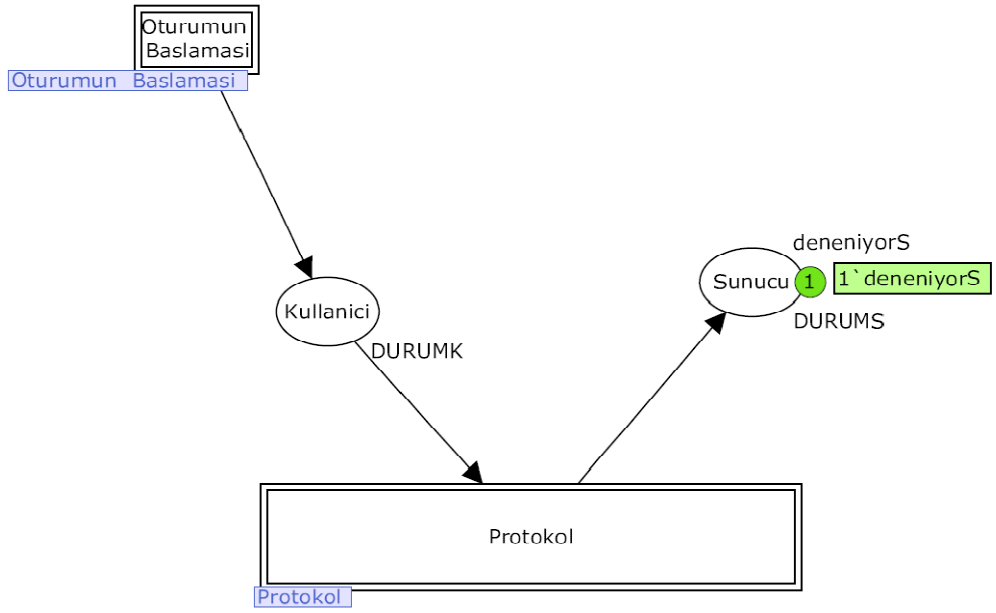
DAVET modelinde yapıldığı gibi sınırsız durum uzayı probleminden kaçınılmak için sınırlama yapılmıştır. "İstek Gönderildi" geçişi [a=0] ile sınırlandırılmıştır.

Modelin uygunluk özelliği incelenirse, "Cevap Alındı", "Cevap Kayıp", "E ve F Zamanlayıcıları", "İstek Kayıp", "Kullanıcı İletim Hatası", "Sunucu İletim Hatası" geçişlerinin "No Fairness" olduğunu görürüz. Bunun anlamı bu geçişlerin sınırsız kez etkin olabileceğini göstermektedir. Ancak biz bu geçişlere şartlar koyarak bunun önüne geçebildik. "İstek Alındı", "İstek Gönderildi", "J Zamanlayıcısı" ve "K Zamanlayıcısı" geçişlerinin ise sınırlı etkin olabileceğini buradan anlayabiliriz.

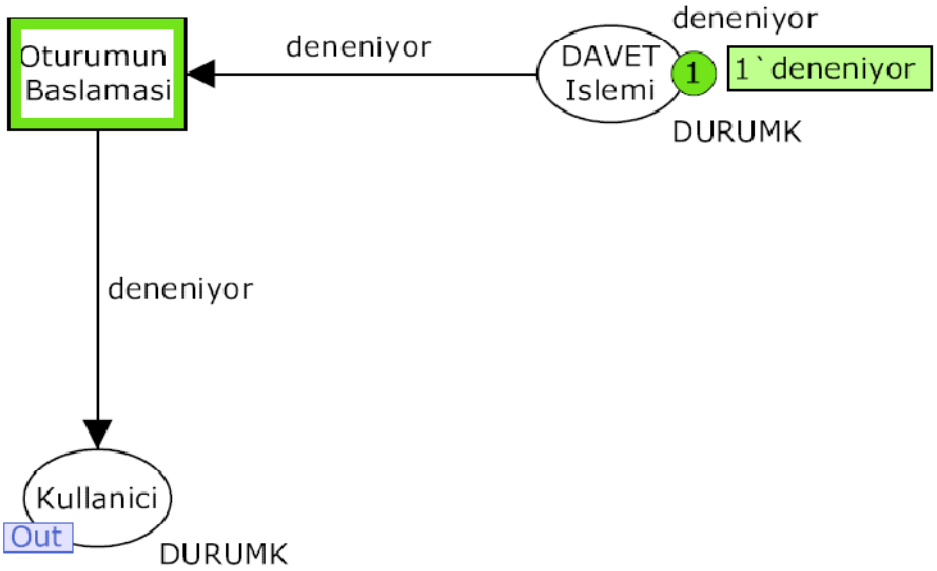
Modelin sınırlılık özellikleri incelenirse, EK B.4'deki rapora göre, "E", "F", "y", "Kullanıcı", "Sunucu" en fazla 1 en az 1 kez jeton olmalıdır. Istekler yerinde en fazla 1 en az 0, "Cevaplar" yerinde ise en fazla 139 en az 0 jeton olmalıdır. Örneğin Cevaplar yerinde 139 tane jeton şu şekilde bulunur: 1`msjyok++, 1`r100++, 137`r101++, 1`r2xx++,1`r3xx. r101 geçici cevabının bu kadar çok gelmesinin sebebi DAVET işleminde bir hareket beklenmesidir.

5.1.1 DAVETsiz Modelinin Performans Analizi

CPN Tools programının performans analizi yapabilmesi için modelimizin genel sistem modülü şekil 5.5'te verildiği gibidir.



Şekil 5.5 : Performans Analizi için DAVETsiz Modelinin En Üst Modülü



Şekil 5.6 : Performans Analizi için Modelin Oturumun Başlaması Modeli

Oturumun Başlaması modülünün amacı DAVET işleminden gelen bir etki olduğunu belirtmektir. Kullanıcı yerini *Deneniyor* durumuna geçirir. Performans analizi için bir takım monitörler tanımlanmalıdır.

Bizim modelimiz için kullandığımız monitörler şu şekildedir:

- Marking_size_Protokol'Kullanici_1 : Kullanıcı yerindeki jetonları sayar.
- Marking_size_Protokol'Istekler_1 : Istekler yerindeki jetonları sayar.
- Marking_size_Protokol'E_1 : E yerindeki jetonları sayar.
- Marking_size_Protokol'F_1 : F yerindeki jetonları sayar.
- Marking_size_Protokol'Sunucu_1 : Sunucu yerindeki jetonları sayar.
- Marking_size_Protokol'Cevaplar_1 : Cevaplar yerindeki jetonları sayar.
- Count_trans_occur_Protokol'K_Zamanlayıcısı_1 : "K Zamanlayıcısı" geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'Kullanici_İletim_Hatası_1: "Kullanıcı İletim Hatası" geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'İstek_Gönderildi_1 : "İstek Gönderildi" geçişinin oluşma sayısını sayar.

- Count_trans_occur_Protokol'Cevap_Alındı_1 : “Cevap Alındı” geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'E ve F_Zamanlayıcıları_1 : “E ve F Zamanlayıcıları” geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'Istek _Kayip_1 : “İstek Kayıp” geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'Istek_Alındı_1: “İstek Alındı” geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'Cevap_Gönderildi_1: “Cevap Gönderildi” geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'Cevap Kayıp_1: “Cevap Kayıp” geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'Sunucu_İletim_Hatası_1: “Sunucu İletim Hatası” geçişinin oluşma sayısını sayar.
- Count_trans_occur_Protokol'J_Zamanlayıcısı_1: J_Zamanlayıcısı geçişinin oluşma sayısını sayar.
- Group 1: : İsteklere karşı verilen cevap oranıdır. Arada mesaj kaybı olup olmadığına bakar.

50 adımlık benzetim sonuçlarının performans analizleri şu şekildedir.

Çizelge 5.7 : Zamanlı Performans Analizi Sonuçları

Timed statistics				
Name	**Count**	**Avrg**	**Min**	**Max**
Group_1	50	0.000000	0	8
Marking_size_Protokol'Cevaplar_1	48	0.000000	0	8
Marking_size_Protokol'E_1	2	0.000000	0	0
Marking_size_Protokol'F_1	2	0.000000	0	0
Marking_size_Protokol'Istekler_1	4	0.000000	0	1
Marking_size_Protokol'Kullanici_1	5	0.000000	0	1
Marking_size_Protokol'Sunucu_1	30	0.000000	1	1

Performans analizlerinden de anlaşılacağı gibi Cevap Kayıp, İstek Kayıp ve "Kullanıcı İletim Hatası" geçişleri birer kez, "Sunucu İletim Hatası" 4 kez ateşlenmiştir. Bu da bize 48 cevaptan 7 kez hatalı dönüş olduğunu göstermektedir. Ek B.6'te verilen analiz sonuçlarında Group monitöründe ortalama kaybolan mesajın 6-7 olduğunu da görebiliriz. DAVET modelinde olduğu gibi kullanıcı işleminin 5 isteğinden biri iletim hatasına uğramış ve işlem sonlanmıştır.

Çizelge 5.8: Zamansız Performans Analizi Sonuçları

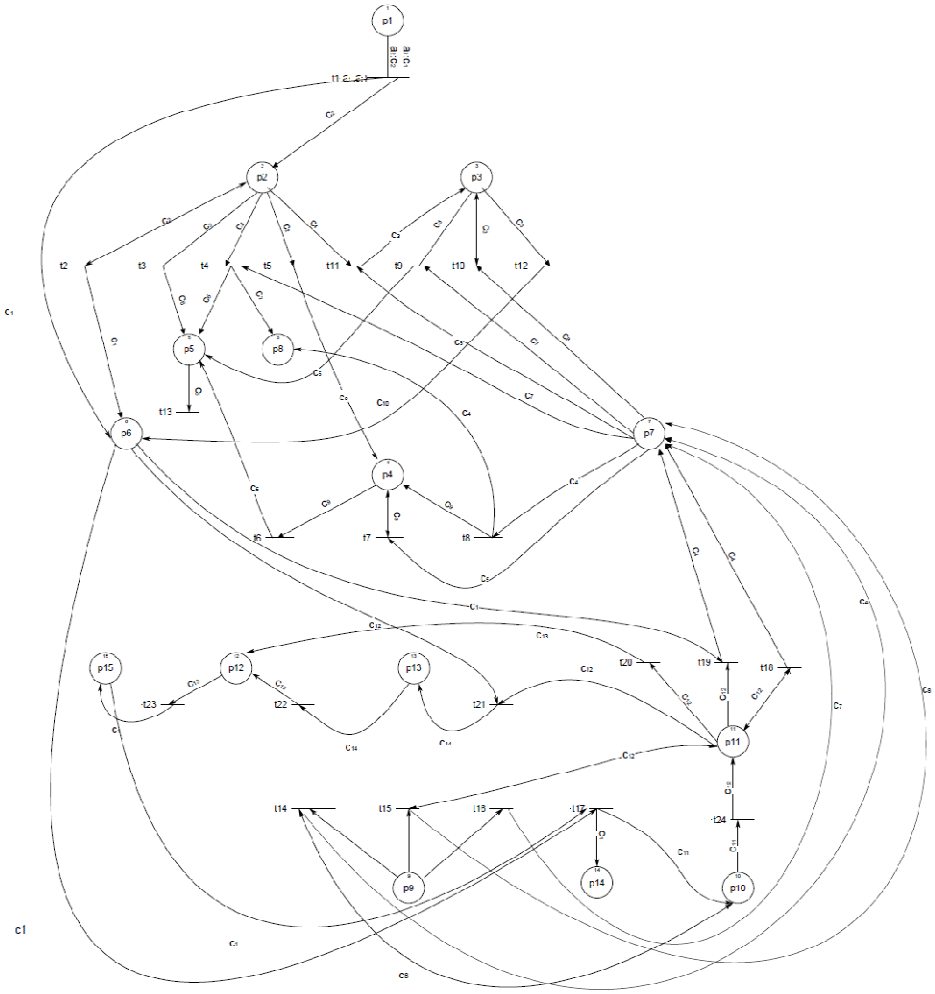
Untimed statistics					
Name	**Count**	**Sum**	**Avrg**	**Min**	**Max**
Count_trans_occur_Protokol'Cevap_Gönderildi_1	23	23	1.000000	1	1
Count_trans_occur_Protokol'Cevap_Kayıp_1	18	18	1.000000	1	1
Count_trans_occur_Protokol'Cevap__Alındı_1	1	1	1.000000	1	1
Count_trans_occur_Protokol'E_ve_F_Zamanlayıcıları_1	0	0	0.000000	0	0
Count_trans_occur_Protokol'İstek_Alındı_1	0	0	0.000000	0	0
Count_trans_occur_Protokol'İstek_Gönderildi_1	0	0	0.000000	0	0
Count_trans_occur_Protokol'İstek__Kayıp_1	1	1	1.000000	1	1
Count_trans_occur_Protokol'J_Zamanlayıcısı_1	1	1	1.000000	1	1
Count_trans_occur_Protokol'K_Zamanlayıcısı_1	0	0	0.000000	0	0
Count_trans_occur_Protokol'Kullanıcı_İletim_Hatası_1	1	1	1.000000	1	1
Count_trans_occur_Protokol'Sunucu__İletim_Hatası_1	4	4	1.000000	1	1

5.3 Klasik Petri Ağı Grafiği Kullanılarak Oluşturulan Model ve Analizi

CPN Tools programının dışında herhangi bir araç kullanılırsa toplamda 15 yer, 24 geçişe, 14 tane yer rengi, 2 tane geçiş rengi içeren bir model elde edilir. Bu modelin analizi yazılan programla yapılmıştır. Görüldüğü gibi CPN Tools programı sayesinde oklara bir takım ifadeler koyarak daha az geçiş ve yere sahip modeller oluşturulabilir. Şekil 5.7'de klasik çizim ile DAVET işleminin zamanlı petri ağları ile modeli, Çizelge 5.9'da bu modelin düğümlerinin ve renklerinin anlamları gösterilmektedir.

5.3.1 DAVET İşleminin Modeli

Klasik petri ağı grafiği kullanılarak oluşturulan model Şekil 5.7'de verildiği gibidir.



Şekil 5.7: DAVET İşleminin Klasik Renkli Petri Ağları Modeli

Çizelge 5.9: Modelde Kullanılan Düğümlerin ve Renklerin Anlamları

İşlemler	
Yerler	**Anlamları**
p1	Kullanıcı
p2	Kullanıcı çağırma durumunda
p3	Kullanıcı ilerleme durumunda.
p4	Kullanıcı tamamlandı durumunda
p5	Kullanıcı sonlandı durumunda
p6	Kullanıcı istek gönderdi, sunucu mesaj aldı.
p7	Sunucu istek gönderdi, kullanıcı mesaj aldı.
p8	Kullanıcı istekleri
p9	Sunucu
p10	Sunucu ilerleme durumunda
p11	Sunucu tamamlandı durumunda
p12	Sunucu sonlandı durumunda
p13	Sunucu onaylandı durumunda
p14	Sunucu istekleri
p15	Yeni DAVET isteği
Geçişler	**Anlamları**
t1	DAVET gönderildi
t2	A zamanlayıcısı
t3	B zamanlayıcısı
t4	Çağırma durumuna geçti
t5	300-699 mesajı geldi
t6	D zamanlayıcısı
t7	Yeniden 300-699 mesajı geldi
t8	hata
t9	r2xx mesajı geldi
t10	Yeniden r1xx mesajı geldi
t11	r1xx mesajı geldi
t12	r3xx mesajı geldi
t13	İşlem sona erdi
t14	Sunucu işlemi için r1xx mesajı geldi
t15	İletim hatası
t16	Sunucu işlemi için r2xx mesajı geldi
t17	Davet gönderildi
t18	G zamanlayıcısı
t19	Yeniden davet gönderildi
t20	H zamanlayıcısı
t21	ALINDI mesajı geldi
t22	I zamanlayıcısı
t23	İşlemde bozulma oldu
t24	Sunucu işlemi için 300-699 mesajı geldi

Çizelge 5.9: Modelde Kullanılan Düğümlerin ve Renklerin Anlamları (Devam)

Geçiş Renkleri	
a_1	DAVET isteği geldi
a_2	Çağırma durumuna geçti
Yer Renkleri	
c_1	DAVET
c_2	çağırma
c_3	ilerleme
c_4	hata
c_5	r3xx
c_6	sonlandı
c_7	200OK
c_8	r1xx
c_9	tamamlandı
c_{10}	ALINDI
c_{11}	ilerlemeS
c_{12}	tamamlandıS
c_{13}	sonlandıS
c_{14}	onaylandıS

5.3.2 Yazılım

Program Bloodshed Software'in çıkardığı açık kaynak bir derleyici olan dev c++ ile yazılmış ve derlenmiştir. Programa giriş olarak başlangıç işaretlemesi, geçiş, yer, geçiş rengi ve jeton rengi sayıları, geçişlere verilen zaman aşımları, girdi ve çıktı matrisleri verilir. Matrisler metin dosyalarına alt alta ve her matris arasında bir satır boşluk bırakılarak yazılmalıdır. Program çıkış olarak durum uzayını verir. Bu durum uzayının sonunda öncelikle oluşabilecek tüm durumlar sonra geçişlerin verilen zaman aşımı olduktan sonraki durumları yer alır.

Ek B.7'de Şekil 5.7'deki model için verilen giriş bilgileri bulunmaktadır.

5.3.3 DAVET İşleminin Modelinin Analizi

Programın çıktıları şu şekildedir:

Hesaplama süresi :3.44900 saniyedir.

Birbirinden farkli 22 tane durum vardir bunlardan 14 tanesi zamanlı işaretlemedir.

Çıktılar ekte verilmiştir.

Kullanılan bilgisayarların hızı analiz süresini değiştirmektedir. Bu analiz Intel Core Solo 1.4 GHz islemcili, 4GB RAM'li bir bilgisayarda yapılmıştır ve analiz süresi 350ms sürmüstür. Sistemin karmaşıklığı, geçiş ve yer sayısı artıkça analiz süresi artacaktır.

6. SONUÇLAR VE ÖNERİLER

Internet teknolojisi yaygınlaştıkça, IP üzerinden yapılan konuşmalar artmakta ve daha sonraki yıllarda artmaya devam edecektir. Bu nedenle Oturum Başlatma Protokolü gibi H.323, SAP(Standard announcement protocol) ve SDP(Streaming Download Project) gibi işaretleşme protokolleri, RTP(Real Time Transport Protocol), RTCP(Real Time Control Protocol) gibi veri aktarım protokollerinin kullanımı hızla artmaktadır.

Bu çalışmada, zamanlanmış renkli petri ağlarıyla Oturum Başlatma Protokolü'nün işlemleri modellenmiş ve CPN Tools programı ile durum uzayı ve performans analizleri yapılmıştır. Bu program renkli ve/veya zamanlanmış renkli petri ağlarında geniş kullanıma sahiptir. Ayrıca bu program kullanılmadan Oturum Başlatma Protokolü'nün bir işleminin modeli çıkarılmış ve bir program yazılarak bu modelin analizi yapılmıştır. Böylece CPN Tools programı gibi kullanımı zor bir program sadeleştirilmiş ve daha kullanışlı bir hale getirilmiştir.

Bölüm 2'de CPN Tools programın zamanlanmış petri ağları benzetiminde kullandığı formüller ve örneklendirilmiştir. Daha sonra Bölüm 3'te anlatılan ve durum makineleri verilen Oturum Başlatma Protokolü'nün işlemlerinin modelleri çıkarılmış ve analizleri yapılmıştır. Modeller istenen tüm şartları sağlamaktadır, bu şartlar işlemlerin durum makinelerinin akışına göre belirlenir. Durum uzayı analizi sonucunda hangi durumda hangi işlemin ne şekilde davrandığı açıkça belirtilmiştir.

Bu çalışmada, zamanlayıcılar arasındaki ilişki, analizlere dayandırılmış ve işlemlerin sıralı olarak gerçekleştirdiği olaylar elde edilmiştir. Bu zamanlayıcıların aralarındaki ilişkiyi açıklayabilmek için çalışma ortamının güvensiz olduğu varsayılmıştır.

OBP zamanlanmış renkli petri ağları ile ilk kez bu tez çalışmasında modellenmiş ve benzetimleri yapılmıştır.

Daha sonraki zamanlarda önerilen modellerden ve analizlerinden Oturum Başlatma Protokolü'nün güvenliği ölçülebilir ve bunların daha sağlam bir yapıya getirmesi sağlanabilir. Bu açıdan bu tez çalışmasında önerdiğim model bu tür protokol modellenmesi çalışmalarında temel olarak kullanılabilir.

Modeller beklendiği gibi sonuçlar verse de kullanıcı ya da sunucu işlemi sona erdiğinde modelde ölü noktalar oluşmaktadır. Aslında bu beklenen bir davranıştır, fakat daha sonra bunların ölü nokta olarak gösterilmemesi içinde çözüm bulunabilir.

CPN Tools programı modelde oluşabilecek durumlar çok fazla olduğunda hata vermektedir. Bu nedenle sadece bizim amacımıza yönelik basit bir program yazılmıştır. Daha sonra bu programa görsel grafiksel özellikler kazandırılarak daha kullanışlı hale getirilebilir.

İleri zamanlarda bu çalışmasında önerilen modeller örnek alınarak, yukarıda sayılan OBP benzeri protokollerin de petri ağı analizleri yapılabilir.

KAYNAKLAR

[1] **Ahson, S. and Ilyas M.,** 2009. SIP Handbook: Services, Technologies, and Security of Session Initiation Protocol, *CRC Press*, USA.

[2] **Bago, M., Peric, N. and Marijan, S.,** 2008. Modeling Wire Train Bus Communication Using Timed Colored Petri Nets, *SICE Annual Conference, 2008*, Tokyo, pp. 2905-2910.

[3] **Beaudouin-Lafon, M., Mackay,W. E., Andersen,P., Janecek, P., Jensen,M., Lassen, M., Lund,K., Mortensen, K., Munck, S., Ratzer, A., Ravn, K., Christensen S. and Jensen, K.,** 2001. CPN/Tools: A Post-WIMP Interface for Editing and Simulating Coloured Petri Nets, *Applications and theory of Petri nets 2001,* Vol. 2075, pp. 71-80.

[4] **Ramchandani, C.,**1974. Analysis of asynchronous concurrent systems by timed Petri nets. Massachusetts Inst. Technol., *Project MAC*, Tech. Rep. 120,

[5] **Çayır, S.**, 2004. Petri Ağlarında Değişmez Analizi, *Yüksek Lisans Tezi,* İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.

[6] **David, R. and Alla, H.,** 2004. Discrete, Continuous, and Hybrid Petri Nets, *Springer*

[7] **Deva, M.,**2004. İnternet Protokolü Üzerinden Ses Aktarımı, *Yüksek Lisans Tezi,* İ.T.Ü. Fen Bilimleri Enstitüsü, İstanbul.

[8] **Ding, L.G. and Liu, L.**,2008. Modelling and Analysis of the INVITE Transaction of the Session Initiation Protocol Using Coloured Petri Nets, *29th Int. Conf. on Applications and Theory of Petri Nets and Other Models of Concurrency, LNCS, vol 5062, pp. 132-151, Springer.*

[9] **Ergan, Z.H.,** 2004. Zamanlandırılmış Petri Ağlarında Sistem Çıkmazı Analizi, *Yüksek lisans tezi*, Anadolu Üniversitesi, Fen Bilimleri Enstitüsü.

[10] **Gehlot, V. and Hayrapetyan, A.,** 2006. A CPN Model of a SIP-Based Dynamic Discovery Protocol for Webservices in a Mobile Environment, *7th Workshop and Tutorial on Practical Use of CPNs and the CPN Tools*, Uni. of Aarhus, Denmark.

[11] **Huang,Y., Chung, T. and Lin, J.,** 2006. A Timed Coloured Petri Net Supervisor for Urban Traffic Networks Computational Engineering in Systems Applications*, IMACS Multiconference on Volume 2*, pp 2151 – 2156

[12] **Jensen, K., Kristensen, L.M. and Wells,L.,** 2007. Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems, *International Journal on Software Tools for Technology Transfer (STTT).* Vol. 9, no. 3S, pp. 213-254.

[13] **Jensen, K. and Kristensen, L.M.,** 2009. Coloured Petri Nets: Modeling and Validation of Concurrent System, *Springer Verlag.*

[14] **Jensen, K.,** 1993. An Introduction to the Theoretical Aspects of Coloured Petri Nets, Lecture Notes In Computer Science; Vol. 803, *Springer-Verlag London*, UK.

[15] **Liu, L.** 2009. Verification of SIP İşlem Using Coloured Petri Nets, *In Proc. Thirty-Second Australasian Computer Science Conference (ACSC 2009),* Wellington, New Zealand. CRPIT, 91. Mans, B., Ed. ACS. 63-72.

[16] **Peng, Y., Yuan, Z. and Wang, J.** 2007. Petri Net Model of Session Initiation Protocol and Its Verification, *Int. Conf. on Wireless Communications, Networking and Mobile Computing,* pp. 1861-1864, IEEE.

[17] **Petri**, **C.A.** 1962, Kommunikation mit Automaten, *Doctoral thesis*, Technical University Darmstadt, Germany.

[18] **Ramchandani, C.**, 1973,"Analysis of Asynchronous Concurrent Systems by Petri Nets", *Dissertation Thesis at M.I.T., Department of Electrical Engineering,*

[19] **Ratzer, A.V., Wells, L.,Lassen, H.M., Laursen, M., Qvortrup, J.F., Stissing, M.S., Westergaard, M., Christensen, S., Jensen, K.,** 2003. CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets, *İlerlemes of the 24th International Conference on Applications and Theory of Petri Nets (ICATPN 2003),* Eindhoven, The Netherlands, pages 450-462. Volume 2679.

[20] **Rosenberg, J., et al.** 2002. RFC 3261: SIP: Session Initiation Protocol. IETF, http://www.faqs.org/rfcs/rfc3261.html.

[21] **Sparks, R.,** 2007. draft-sparks-sip-invfix-00: Correct işlem handling for 200 responses to Session Initiation Protocol INVITE requests. *Internet Engineering Task Force (2007), http://tools.ietf.org/id/draft-sparks-sip-invfix-00.txt*

[22] **Ubiquity Software**, 2004, Understanding SIP: Today's Hottest Communications Protocol Comes of Age

[23] **Url-1** **<***http://www.daimi.au.dk/designCPN/.* > alındığı tarih 04.04.2010

[24] **Url-2** <*http://www.daimi.au.dk/PetriNets/tools/db.html.*> alındığı tarih 04.04.2010

[25] **Url-3** **<** *http://www.endersys.com*> alındığı tarih 27.09.2007

[26] **Url-4** <*http://www.yasinkaplan.com/tr/docs/SIP.pdf*> alındığı tarih 30.03.2010

[27] **Url-5 <***http://en.wikipedia.org/wiki/Session_Initiation_Protocol*> alındığı tarih 30.03.2010

[28] **Wan, H., Su, G., Ma, H.,** 2007. SIP for mobile networks and security model. *İlerlemes of International Conference on Wireless Communications, Networking and Mobile Computing (WiCom 2007),* pp. 1809-1812.

[29] **Wang, J.,** 1998. *TIMED PETRI NETS Theory and Application*, Kluwer Academic Publishers.

[30] **Wells, L.,** 2002, Performance Analysis Using Coloured Petri Nets,*Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, Page: 217

[31] **Yılmaz, A.,** 2009. Zamanlanmış Petri Ağlarında Erişilebilirlik Analizi, *Yüksek lisans tezi*, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü.

[32] **Yorulmaz, S.,** 2009. Renkli Petri Ağlarında Erişilebilirlik Ağacı ve P Değişmezleri Analizi, *Yüksek lisans tezi*, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü.

[33] **Yurttakal, S.,** 2009. Zamanlandırılmış Petri Ağlarında Ateşlenebilirlik Problemlerinin İncelenmesi, *Yüksek lisans tezi*, İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü.

[34]**Zuberek, W.M.,"**Timed Petri Nets and Preliminary Performance Evaluation",IEEE,*Institute of Computer Science Technical University of Warsaw,* 1980

[35] **Zuberek, W.M.,** "Timed Petri nets : definitions, properties,and applications", *Microelectron.Reliab.,* 31, 4, 627-644, 1991

[36]**Zuberek, W.M.,** "M-Timed Petri Nets, Priorities, Preemptions, and Performance Evaluation of Systems", *Lecture Notes in Computer Science*, Vol. 222, Advances in Petri Nets, 478-499, 1985.

[37] **Zuberek,W.,M., I. Rada,** 2001. Modeling and Analysis of Distributed State Space Generation for Timed Petri Nets. *Annual Simulation Symposium* :pp. 93-98

EKLER

EK A.1 : OBP'nin Davet İçeren Kullanıcı İşleminin Sağlaması Gereken Durumlar
EK A.2 : OBP'nin Davet İçermeyen Kullanıcı İşleminin Sağlaması Gereken Durumlar
EK A.3 : OBP'nin Davet İçeren Sunucu İşleminin Sağlaması Gereken Durumlar
EK A.4 : OBP'nin Davet İçermeyen Sunucu İşleminin Sağlaması Gereken Durumlar
EK A.5 : CPN Tools Programının Arayüzü ve Modelleme Dili
EK A.6 : CPN Tools Programının Elemanları Durum Uzayı Aracı
EK A.7 : CPN Tools Programının Performans Analizi
EK B.1 : OBP'nin DAVET İşleminin Modelinin Durum Uzayı Raporu
EK B.2 : OBP'nin DAVET İşleminin Modelinin Performans Analizi İçin Kullanılan Monitörleri
EK B.3 : DAVET İşleminin Modelinin Performans Analizi Sonrası Tutulan Günlük Dosyaları
EK B.4 : OBP'nin DAVETsiz İşleminin Modelinin Durum Uzayı Raporu
EK B.5 : OBP'nin DAVETsiz İşleminin Modelinin Performans Analizi İçin Kullanılan Monitörleri
EK B.6: DAVETsiz İşleminin Modelinin Performans Analizi Sonrası Tutulan Günlük Dosyaları
EK B.7: DAVET İşleminin Genel Petri Ağı Modelinin Analizi için Programa Verilen Girişler
EK C.1 : OBP'nin DAVET İşleminin Modeli
EK C.2 : OBP'nin DAVETsiz İşleminin Modeli

Ek A.1

İşlem *Çağırma* durumunda olduğu zaman aşağıdaki 6 olaydan biri oluşur:

- *Zamanlayıcı A* ateşlenir. İşlem zamanlayıcıyı 2*T1 değeri yeniden başlatır ve DAVET isteğini yeniden iletir. 2*T1 değerinden sonra *Zamanlayıcı* A değeri ateşlenirse, istek yeniden iletilmek zorundadır. Bu işlem isteğin her iletim ile iki katına çıkan aralıklarda yeniden iletilmesi devam eder.
- *Zamanlayıcı* B ateşlenir işlem *Sonlandı* durumuna girer.
- OBP iletim katmanında ağ üzerinden DAVET isteği gönderilemezse, iletim katmanı tarafından bir iletim hatası raporlanır. İşlem kendi TU'suna haber verir ve *Sonlandı* durumuna girer.
- 1xx geçici cevabı alınır. İşlem cevabı kendi TU'suna gönderir ve *İlerleme* durumuna girer ve diğer cevapları bekler.
- Son başarılı cevap 2xx alınır, DAVET isteğinin kabul edildiği sunucuya gönderilir. İşlem cevabı kendi TU'suna gönderir, *İlerleme* durumuna girer ve diğer cevapları bekler.
- Başarısız son cevap (300-699) cevabı alınır, DAVET isteğinin kabul edilmediği sunucuya gönderilir. İşlem cevabı kendi TU'suna gönderir, ALINDI isteği yaratır ve OBP iletim katmanına koyar ardından *Tamamlandı* durumuna girer.

İşlem *İlerleme* durumunda olduğu zaman aşağıdaki olaydan biri oluşur:

- 1xx geçici cevabı alır ve bu durumda kalır.
- Son final başarılı cevabı alır ve *Sonlandı* durumuna girer.
- Son başarılı olmayan cevabı alır ve ALINDI'yı hazırlayıp gönderdikten sonra *Tamamlandı* durumuna girer.

Tamamlandı durumunun amacı güvensiz ortamda sunucu tarafından yeniden iletilen 300*699 cevabını absorbe etmektir. Bu duruma girildiğinde,

- *Zamanlayıcı* D başlatılır. Zamanlayıcı sona erdiğinde işlem *Sonlandı* durumuna girer.
- Eğer 300-699 mesajı *Zamanlayıcı* D bitmeden gelirse, işlem yaratılır ve ALINDI mesajı göndererek aynı durumda kalır.
- OBP iletim katmanı ALINDI gönderirken iletimde bir hata meydana gelirse, TU haberdar edilir ve işlem *Sonlandı* durumuna girer.

Kullanıcı işlemi *Sonlandı* durumuna girdiğinde TU tarafından direk sonlandırılır.

Ek A.2

İşlem *Deneniyor* durumunda olduğu zaman aşağıdaki olaydan biri oluşur:

- Zamanlayıcı E ateşlenir. İşlem zamanlayıcıyı yeniden başlatır ve isteği yeniden iletir.
- *Zamanlayıcı* F ateşlenir ve işlem *Sonlandı* durumuna girer.
- OBP iletim katmanında ağ üzerinden istek gönderilemezse iletim katmanı tarafından bir iletim hatası raporlanır. İşlem kendi TU'suna haber verir ve *Sonlandı* durumuna girer.
- 1xx geçici cevabı alınır. İşlem cevabı kendi TU'suna gönderir ve *İlerleme* durumuna girer ve diğer cevapları bekler.
- Son başarılı cevap 2xx alınır. İşlem cevabı kendi TU'suna gönderir ve *Tamamlandı* durumuna girer.

İşlem *İlerleme* durumunda olduğu zaman aşağıdaki olaydan biri oluşur:

- 1xx geçici cevabı alır ve bu durumda kalır.
- *Zamanlayıcı* E ateşlenir. İşlem zamanlayıcıyı yeniden başlatır ve isteği yeniden iletir. E zamanlayıcısı T2 değeri ile ilk durumuna döner.
- Zamanlayıcı F ateşlenir işlem *Sonlandı* durumuna girer.
- 1xx geçici cevabı alınır. İşlem cevabı kendi TU'suna gönderir ve *İlerleme* durumuna girer ve diğer cevapları bekler.
- Son başarılı cevap 2xx alınır. İşlem cevabı kendi TU'suna gönderir ve *Tamamlandı* durumuna girer.

Tamamlandı durumuna girildiğinde,

- Zamanlayıcı K başlatılır. Zamanlayıcı sona erdiğinde işlem *Sonlandı* durumuna girer. K zamanlayıcısı T4 zamanında ateşlenebilecek şekilde ayarlanır. Bu güvenli sistemler için 0 s'dir. *Tamamlandı* durumunda ek cevapların yeniden iletimi saklanır. T4 bu sunucu ve kullanıcı arasındaki mesajların temizlenme süresini gösterir. T4'ün varsayılan değeri 5 s'dir.
- Eğer 200-699 mesajı Zamanlayıcı K bitmeden gelirse, işlem yaratılır ve ALINDI mesajı göndererek aynı durumda kalır.

Kullanıcı işlemi *Sonlandı* durumuna girdiğinde TU tarafından direk sonlandırılır.

Ek A.3

Sunucu işlemi *İlerleme* durumda iken,

- TU'dan OBP iletim katmanı tarafından 101-199 geçici cevapları geçmez ve aynı durumda kalır.
- Kullanıcı işlemi tarafından yeniden iletilen DAVET isteğini alır, ardından sunucu işlemi geçici cevabını kendi TU'sundan almadan önce yeniden iletir ve aynı durumda kalır.
- Kendi TU'sunda başarısız son 300*699 cevabını alır ve cevabı göndermeden önce *Tamamlandı* durumuna girer.
- OBP iletim katmanından iletim hatası alır ve işlem *Tamamlandı* durumuna girer.
- Kendi TU'sundan son başarılı 2xx cevabını alır ve işlem cevabı aldıktan sonra *Sonlandı* durumuna girer.

Önce *Tamamlandı* durumuna girilir. Zamanlayıcı H 64*T1 değeri ile başlar. Bu durumda aşağıdaki 5 olay gerçekleşir.

- Yeniden iletilen DAVET isteği alınır. İşlem kendi TU'sudan *İlerleme* durumdayken 300-699 cevabını yeniden iletir ve bu durumda kalır.
- Zamanlayıcı G ateşlenir. İşlem 300*699 cevabını yeniden gönderir, zamanlayıcı MIN(2*T1, T2) ile ayarlanır ve aynı durumda kalır.
- Zamanlayıcı H ateşlenir. İşlem *Sonlandı* durumuna geçer.
- OBP iletim katmanında bir cevap gönderiminde iletim hatası oluşur ve *Sonlandı* durumuna geçer.
- Kullanıcıdan bir ALINDI alınır. İşlem *Onaylandı* durumuna geçer.

Onaylandı durumuna girildiğinde, *Zamanlayıcı* I başlar, işlem 300-699 cevabının yeniden iletimi tarafından tetiklenen ALINDI'lar için bekler. Zamanlayıcı I ateşlenirse, *Sonlandı* durumuna girilir ve sunucu işlemi kendi TU'su tarafından sonlandırılır.

Ek A.4

Sunucu İşlemi *Deneniyor* durumda iken,

- Kendi TU'sunda 1xx cevabını alır ve *İlerleme* durumuna girer.
- Kendi TU'sunda son 200*699 cevabını alır ve *Tamamlandı* durumuna girer.
- Kendi TU'sundan son başarılı 2xx cevabını alır ve işlem cevabı aldıktan sonra *Sonlandı* durumuna girer.

Sunucu işlemi *İlerleme* durumda iken,

- Yeniden iletilen cevabı alınır ve bu durumda kalır.
- Kendi TU'sunda 1xx cevabını alır ve bu durumda kalır.
- OBP iletim katmanında bir cevap gönderiminde iletim hatası oluşur ve *Sonlandı* durumuna geçer.
- Kendi TU'sundan son başarılı 2xx cevabını alır ve işlem cevabı aldıktan sonra *Sonlandı* durumuna girer.

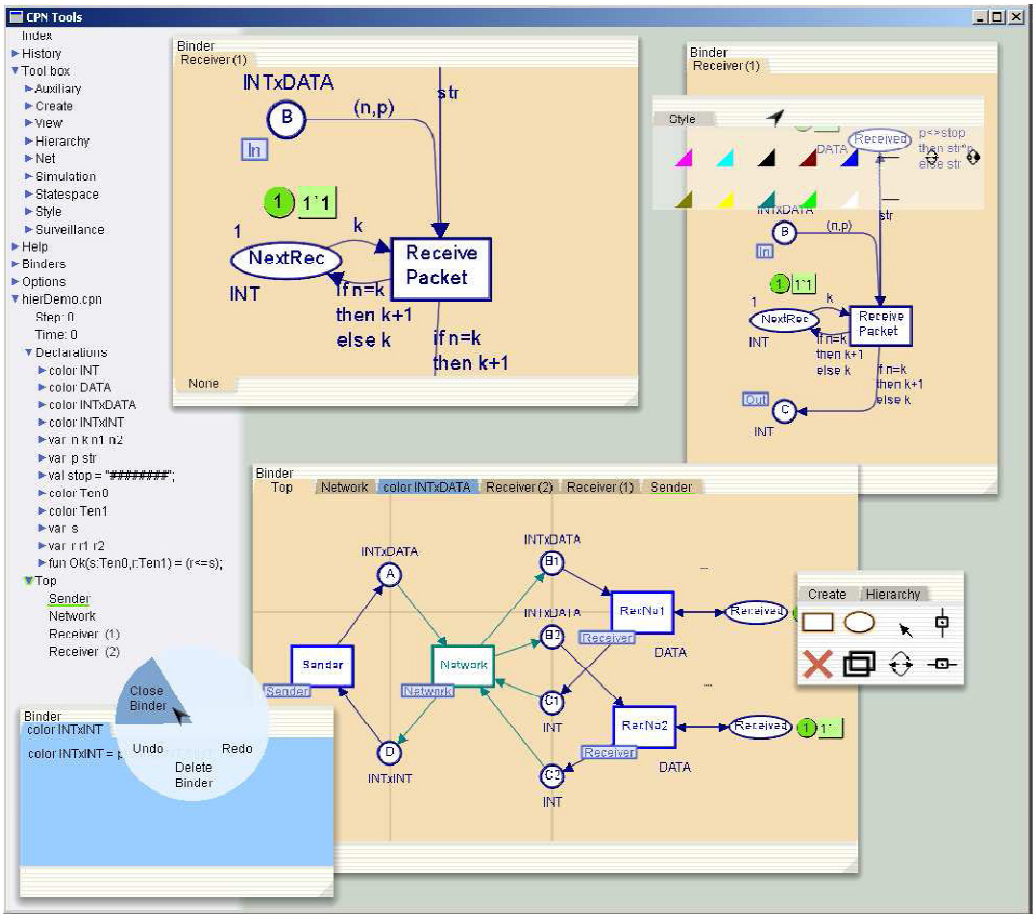
Önce *Tamamlandı* durumuna girilir. *Zamanlayıcı* J başlar. Bu durumda aşağıdaki 5 olay gerçekleşir.

- Yeniden iletilen cevap alınır ve bu durumda kalır.
- Zamanlayıcı J ateşlenir.ve işlem *Sonlandı* durumuna geçer.
- OBP iletim katmanında bir cevap gönderiminde iletim hatası oluşur ve işlem *Sonlandı* durumuna geçer.

Sonlandı durumuna girilir ve sunucu işlemi kendi TU'su tarafından sonlandırılır.

Ek A.5

Şekil A.1'de CPN Tools programının arayüzü gösterilmiştir.



Şekil A.1 : CPN Tools Programının Arayüzü

Soldaki dikdörtgensel alan içeriği göstermektedir. Bu alan, tanımlamaların ve modelin parçalarının yapıldığı, araç kutucuğunu(Toolbox) içerir. Araç kutuğu ise temel elemanların yaratıldığı, kopyalandığı yerdir. Ek olarak sağ tarafta modelin çizilebileceği geniş bir alan bulunmaktadır.

Model oluşturulmak istendiğinde sağ tarafta bulunan “Tool Box” girişinden Create tool bölümü kullanılır. Bu araç yerleri, geçişleri ve oklar gibi petri ağları elemanlarını yaratmak için kullanılır. CPN tools programı otomatik olarak sayfa içerisinde yerleştirmeye yardımcı olur.

Şekil A.2 : CPN Tools Programının Elemanları Yaratma Aracı

Oluşturma (Create) aracı ağın yapısını yaratmak için kullanılmaktadır. Soldan sağa ve yukardan aşağı olarak kullanım amaçları şu şekildedir: Bir geçiş oluşturur, bir yer oluşturur, bir ok oluşturur, bir dikey yönlendirici oluşturur, bir elemanı siler, bir elemanı kopyalar, okun yönünü değiştirir, yatay bir yönlendirici oluşturur.

Çalışma ortamı, birçok sayfayı aynı anda yönetmekle ilgili her türlü kolaylığı sağlanmıştır. Çalışma ortamı bütün ekranı kaplar ve bağlayıcı adı verilen bir takım pencereler sahiptir. Bağlayıcılar her ayrı ekranın bulunduğu sayfaları içerir. Bu sayfalar, renkli petri ağlarının, tanımlamaların ve araçların bir kümesinin görünüşünü sağlar. Her sayfa bir sekmeye sahiptir, bu sekmeler karşılıklı iletişim için kullanılmaktadır.

Her özelliğin belli bir yeri vardır. Örneğin yere ait renk seti sağ alt tarafa yazılır.

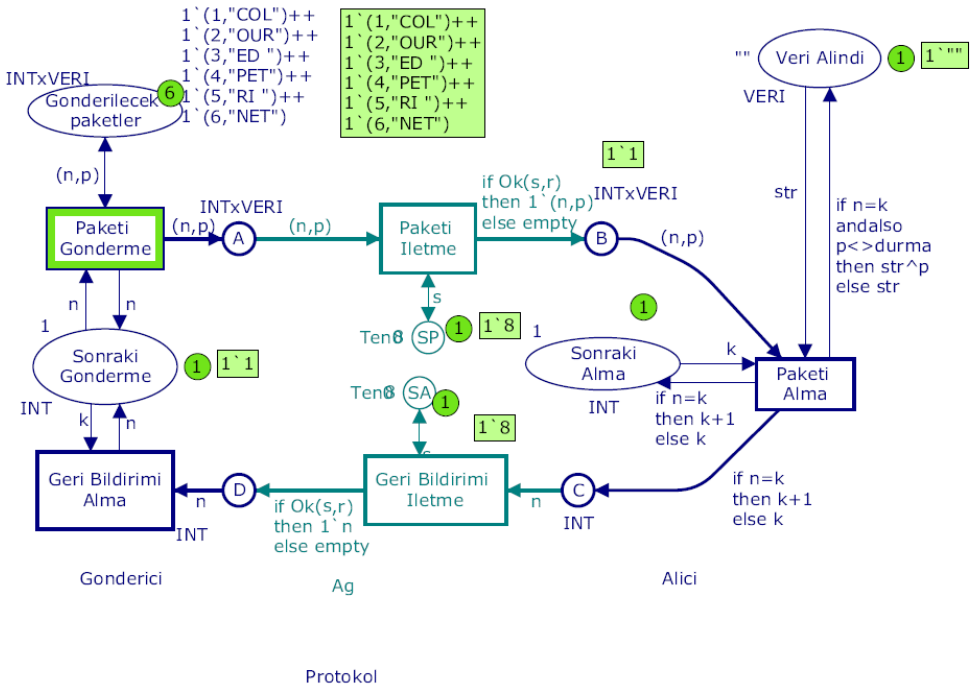
Şekil A.3’de her modelde kullanılabilen benzetim araçları gösterilmektedir. İstenilen araç fare yardımıyla modelin üzerine getirilerek çalıştırılır.

Şekil A.3 : CPN Tools Programının Benzetim Aracı

Araçlar sırayla şu amaçlar için kullanılır: İlk işaretlemeye dönülür, devam eden otomatik Benzetimu durdurur, elle seçilen tek bir geçişi çalıştırır, rastgele seçilen bir geçişi çalıştırır, belirtilen sayı kadar rastgele seçilen bir elemanla otomatik Benzetim adımı gerçekleştirir. Her adım ekranda gösterilir, model kilitlenene kadar bütün benzetim adımlarını gerçekleştirir, CPN ML ifadesini değerlendirir.

Renkli Petri Ağının Modelleme Dili

Örnek olarak basit bir protokolü alınabilir. Gönderici alıcıya birçok veri paketi göndermektedir. Modelin güvensiz ortamlar için geliştirildiğini yani paketlerin kaybolabileceğini, ansızın gelebileceği varsayılsın. Protokol bir sıra numarası, geri bildirim ve yeniden iletim özelliklerini kullansın. Protokol durma-bekleme yöntemini yani geribildirim alınmadan veri paketinin gönderilmesini kullanıldığını düşünülsün.



Şekil A.4 : CPN Tools Programı ile Oluşturulan Basit Bir Protokol Modeli

Şekil A.4'de görüldüğü gibi modelimizde 8 tane yer, 5 geçiş ve yönlendirilmiş oklar bulunmaktadır. Bu oklara ML dili ile yazılmış tanımlamalar eklenmiştir. Yerler ve geçişler genel olarak düğüm adını alır. Bir ok bir yerle bir geçiş arasında olmalıdır. Modelin durumu yerler ile gösterilir. Veri jetonlar ile taşınır ve her yer en az bir tane jeton ile işaretlenmiştir.

Her yerin bitişiğinde jetonların renklerini yani veri değerlerini tanımlayan ifadeler vardır. Bunlar renk kümesi(colour set) adını alır. CPN ML dilinde "colset" anahtar sözcüğü kullanılır.

Örneğin; colset INT = int;

INT renk seti tam sayı değerler alabilir anlamına gelmektedir ve protokoldeki sıra numarasını tutar. VERI renk seti ise metin değerleri içermektedir ve taşınan veri paketlerinin modellenmesi için kullanılır. INT x VERI renk seti iki ifadeye sahiptir.

İlk ifade tamsayı olduğunu, ikincisi metin olduğunu gösterir. Bu renk seti sıra numarası içeren veri paketlerinin modellemek için kullanılır ve aşağıdaki şekilde yazılır.

colset VERI = string;

colset INT x VERI = product INT * VERI

Her yerin bitişimde bulunan diğer bir ifade başlangıç işaretlemesidir. Genellikle yerin sağ üst köşesine yazılır. Örneğin "Sonraki Gönderme" yerinin başlangıç işaretlemesi 1 rengi ile tek bir jeton içerir. Buda bize alıcının öncelikle sıra numarası 1 olan paketi beklediğini gösterir.

++ operatörü çok kümeli ifadeleri alır ve bunların toplamını gönderir. 'operatörü ise sol tarafındaki değişken sayısı kadar sağ tarafındaki değişkenin görüldüğünü ifade eder.

Anlık işaretleme yerin bitişinde küçük daire ile gösterilmektedir. Başlangıçta anlık işaretleme M_o ile gösterilen başlangıç işaretlemesine eşittir.

Bu 5 geçiş sistemdeki olayları gösterir. Bir geçiş gerçekleştiği zaman, jetonları giriş yerinden çıkış yerine iletir. Jeton renkleri de oklardaki anlamlara göre giriş yerlerinden çıkış yerlerine gönderilir.

Bir geçiş ve yer birbirine çift yönlü oklar ile bağlı olabilir. Bu hem giriş yeri hem çıkış yeri olarak kullanılır. “Gönderilecek Paketler” ve “Sonraki Gönderme” yerleri ve “Paketi Gönderme” geçişi birbirine çift yönlü oklar ile bağlıdır.

Ok ifadeleri CPN ML dili ile yazılmış ve belirtilmiş değerler, operatörler, fonksiyonlar ve değişmezler ile yazılmaktadır. Bir ok ifadesi çok kümeli jeton renkleri ile değerlendirilir. Örneğin n ve (n,p) ifadeleri “Gönderilecek Paketler” geçişine bağlıdır ve şu şekilde tanımlanır.

var n : INT;

var d : VERI;

n değeri INT renk setinin değerleri, d değeri VERI renk setinin değerleri ile sınırlandırılmıştır. Örneğin;

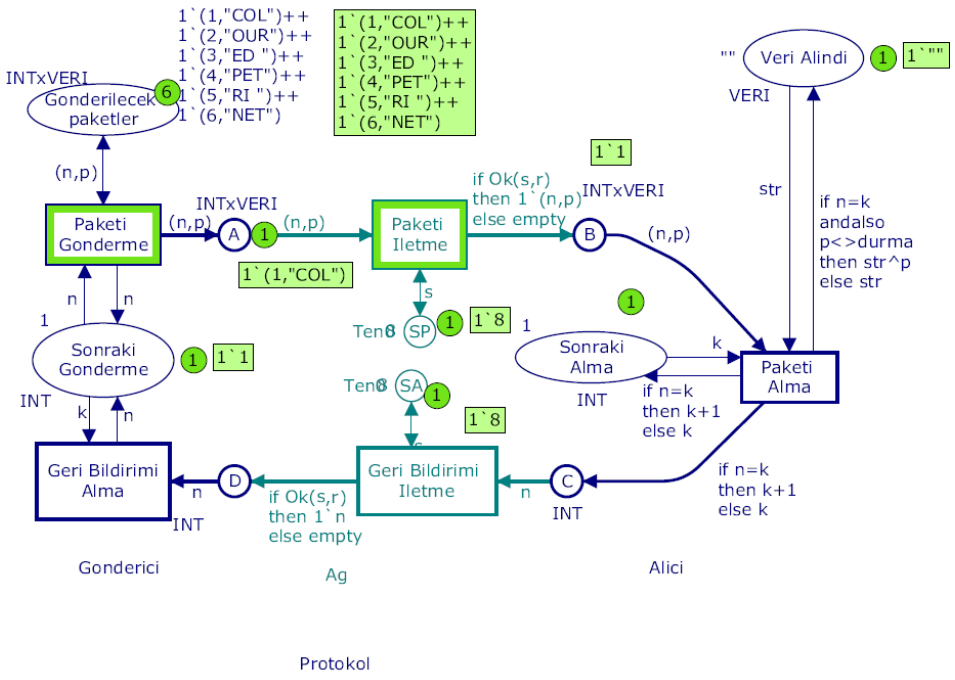
n →1‘3

(n,d) →1‘(3,"CPN")

ile gösterilebilir.

“Paketi Gönderme” geçişi diğer geçişlerden farklı olarak daha kalın gösterilmiştir. Bunun nedeni başlangıçta etkin olabilecek tek geçiş bu geçiş olmasıdır. Bu geçiş gerçekleştiğinde “Gönderilecek Paketler” ve “Sonraki Gönderme” giriş yerlerinden jetonu gönderir. “Gönderilecek Paketler” yerinin başlangıç işaretlemesi 1 rengi ile tek jeton içerir. Burada n değerinin 1 olduğu zorunluluğunu görebiliriz.

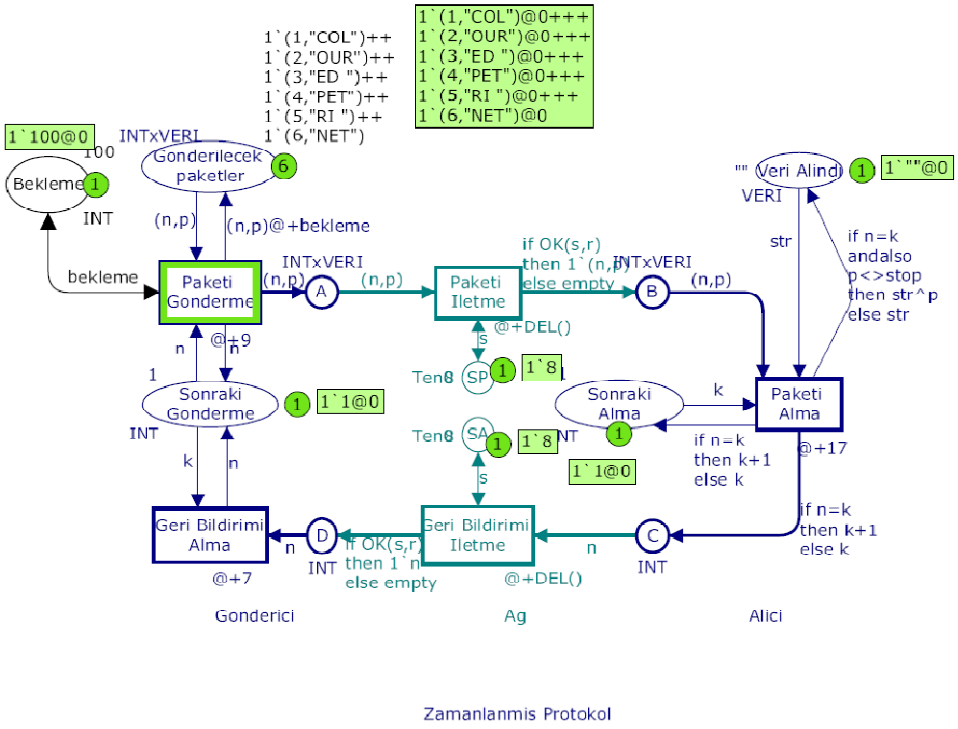
Şekil A.4’de “Paketi Gönderme” geçişi oluştuğu anda ulaşılan işaretlemeyi göstermektedir. “Gönderilecek Paketler” yerinden (1,"COL") rengi “Sonraki Gönderme” yerinden 1 rengi ile jeton çıkar ve A yerinde (1,"COL") rengi oluşur.



Şekil A.5 : Protokol Örneğinin Bir Benzetim Adımından Sonraki İşaretlemesi

Şekil A.5'te protokole zaman eklenerek çıkarılan model görülmektedir. CPN Tools programında yerlere ve geçişlere zaman uygulamasından çok zaman jetonlarla ilişkilendirilmiştir. Yerlere zaman uygulaması yapılmaz ama geçişlerdeki zaman bilgisi ile yapmak istediğimiz her şeyi modelimize uygulayabiliriz. Geçişe gecikme eklenebilir ve bu gecikmeden sonra jeton hedefe ulaşır. Farklı çıkış oklarında farklı zamanlarda jetonların üretilmesi isteniyorsa, okların üzerindeki zaman tanımlamaları kullanılabilir.

Zamanlı ve zamansız renkli petri ağlarının arasındaki en önemli fark zamanlanmış renkli petri ağlarında jetonlara zaman vermektir. Jeton renginin yanı sıra zaman bilgisi de jetonla birlikte taşınabilmektedir. Jetonun zaman bilgisini taşıdığı yerdeki işaretleme de görünüşlerde çoklu kümeler ile gerçekleştirilir. Renkli petri ağı modeli model zamanını gösteren bütünsel bir saate sahiptir. Zaman bilgileri ve bütünsel saat değerleri ile birlikte yerlerdeki jeton dağılımı zamanlanmış işaretleme adını alır.



Şekil A.6 : CPN Tools Programı ile Oluşturulan Zamanlanmış Renkli Petri Ağları ile Protokol Modeli

colset INT = int timed;

colset VERI = string timed;

colset INTx VERI = product INT * VERI timed;

Zaman bilgisi gerçektir ve negatif olamaz. Zaman bilgisi bize o jetonun ne zaman kullanıma hazır olduğunu göstermektedir. Geçişin oluşmasıyla yerlerden bu zaman bilgisi kaldırılır. Bir yerdeki jetonlar, eğer yer zamanlanmış renk kümesi olarak tanımlandıysa taşınabilir. CPN ML dilinde renk seti “timed” anahtar sözcüğü ile yer zamanlanmış renk kümesi olarak tanımlanır. Şekilde gösterilmiştir.

Zamanlanmış renkli petri ağlarının çalıştırılması bütünsel bir saat tarafından kontrol edilir. Bu saat jetonun zaman bilgisinden daha büyük ya da eşit olmadıkça jeton uygun değildir.

Eğer etkin bir geçiş bulunmazsa ama saatin yüksek bir değeri varsa benzetimlik en az bir tane uygun geçiş olabilmesi için zamanı olabilecek en küçük değere getirir.

Gönderilecek paketlerin başlangıç işaretlemesi şu şekildedir:

1‘(1,"COL")@0 +++ 1‘(2,"OUR")@0 +++ 1‘(3,"ED ")@0 +++

1‘(4,"PET")@0 +++ 1‘(5,"RI ")@0 +++ 1‘(6,"NET")@0

Zaman bilgisi “@” işaretinden sonra yazılır. Burada bütün jetonlar 0 zaman bilgisi taşımaktadır. +++ operatörü 2 tane zamanlanmış kümeyi alır ve birim zamanını gönderir. Bütünsel saatin başlangıç değeri 0’dır. Eğer bir yer zamanlanmış renk kümesi ile tanımlanmışsa CPN Tools programı otomatik olarak başlangıç işaretlemelerine zaman bilgisini koyar.

Bekleme yeri “Paketi Gönderme” geçişinin paketi yeniden göndermeden önce ne kadar zaman beklemesi gerektiğini belirtmek için kullanılır. Geçişlere zaman alanı eklenmiştir. “Paketi Gönderme” geçişinin zaman alanı @+9 şeklindedir. Buradaki 9 sayısı bu geçişten çıkan bütün jetonlara 9 bitim saat gecikmesi ekleneceği anlamına gelmektedir. Jetonlar A ve “Bir Sonraki Gönderilecek” yerlerinde yaratılır ve

0 + 9 + 100 = 109

zaman bilgisini alırlar.

Başlangıçtaki 0 bütünsel saat tarafımdan verilen geçişin oluştuğu zamanı, 9 zaman gecikmesini, 100 ise çıkış okundaki zaman gecikmesini yani yeniden iletim süresini gösterir. Yani toplamda bu geçişin işlem süresi 109 birim zamandır.

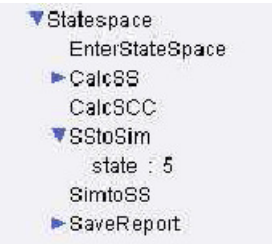
“Paketi İletme” geçişi @+DEL() zaman alanına sahiptir. Bu fonksiyon rastgele bir değer alır bu da iletim işleminin zamanının değişebilirliğini gösterir.

Geri Bildirimin iletim zamanı 25 ve 75 birim zaman aralığındadır.

“Sonraki Gönderme” yerindeki jeton bir zaman bilgisine sahiptir. Gönderici yeni bir “Paketi Gönderme” veya “Geri Bildirimi Alma” başlatacaksa ama zaten bunlar çalışıyor ise bu zaman bilgisi sayesinde göndericinin bütün işlemleri yapabilmesi sağlanır.

Birçok benzetim işleminden sonra zamanlanmış renkli petri ağları ölü işaretlemeye ulaşır.

Ek A.6



Şekil A.7 : CPN Tools Programının Elemanları Durum Uzayı Aracı

EnterStateSpace : Durum uzayı analizini oluşturabilmek için ilk önce bu araç çalıştırılmalıdır. Durum uzayı analizi benzetim gibi otomatik olarak çalışmaz.

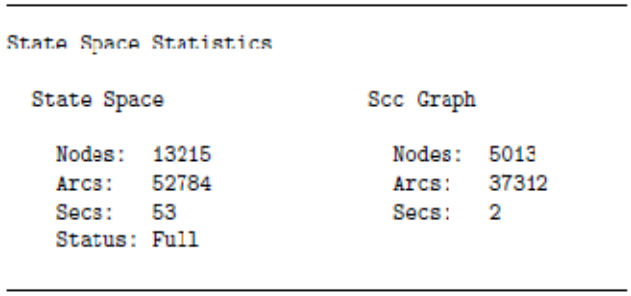
CalcSS : Durum uzayını oluşturur.

CalcSCC : Grafiksel durum uzayının bağlı elemanlarını bulur. Kullanıcı kaç tane durum analizi oluşturmak ya da ne kadar zaman geçirmek istediğini belirtebilir.

SStoSim: Durum uzayındaki bütün durumlar numaralandırılır, bu araç ile istenen durum uzayı verilerek simülatörde nasıl davrandığı görülebilir. Etkin geçişler, işaretlemeler simülatör ile görülebilir.

SimToSS : Şimdiki durumu durum analizine çevirebilir. Böylece parçalı durum analizi oluşturulabilir.

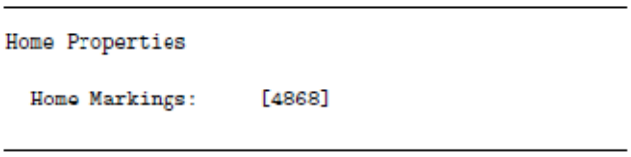
SaveReport: Bu araç ile durum uzayı raporları otomatik olarak kaydedilir. Bu rapor, durum uzayı oluşturmadaki istatistikleri, sınırlılık özelliklerini, canlılık özelliklerini, uygunluk özelliklerini, başlangıç özelliklerini göstermektedir.



Şekil A.8 : CPN Tools Programının Elemanları Durum Uzayı Raporu

Şekil A.8'de durum uzayı istatistikleri görülmektedir. Buradan durum uzayını oluşturulmasının 53 saniye sürdüğünü görebiliriz.

Başlangıç Özellikleri(Home Properties)

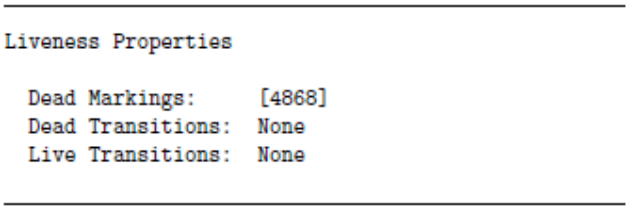


Şekil A.9 : CPN Tools Programının Durum Uzayı Başlangıç Özellikleri

Şekil başlangıç durum özelliklerini göstermektedir. Buradan sadece bir tane 4898 numaralı düğümde başlangıç işaretlemesinin olduğunu görebiliriz.

Canlılık Özellikleri (Liveness Properties)

Şekil A.12'den anlaşılacağı gibi, sadece bir tane ölü işaretleme 4868 numaralı düğümde oluşmuştur. Hiç zorlayıcı bir eleman etkin olmamıştır. Ayrıca etkin hale geçebilecek bir geçiş bulunmamaktadır. Ayrıca hiç ölü geçişin olmaması ulaşılamayan geçiş olmadığını göstermektedir.



Şekil A.10 : CPN Tools Programının Durum Uzayı Canlılık Özellikleri

Zamanlanmış modellerin durum uzayı analizleri daha büyüktür., çünkü zamanlanmış durum uzayındaki düğümler zamanlanmış işaretlemeleri ve bütünsel saat ile zaman bilgisini taşır.

Sınırlılık Özellikleri (Boundness properties)

Bütün erişilebilir işaretlemeler düşünüldüğünde bir yerde tutulan jetonların kaç tane olduğu gösterir.

Best Integers Bounds	Upper	Lower
PacketsToSend	6	6
DataReceived	1	1
NextSend, NextRec	1	1
A, B, C, D	3	0
Limit	3	0

Şekil A.11 : CPN Tools Programının Durum Uzayı Sınırlık Özellikleri

```
Best Upper Multi-set Bounds

  PacketsToSend       1`(1,"COL")++1`(2,"OUR")++
                      1`(3,"ED ")++1`(4,"PET")++
                      1`(5,"RI ")++1`(6,"NET")

  DataReceived        1`""++1`"COL"++1`"COLOUR"++
                      1`"COLOURED "++
                      1`"COLOURED PET"++
                      1`"COLOURED PETRI "++
                      1`"COLOURED PETRI NET"

  NextSend, Nextrec   1`1++1`2++1`3++1`4++1`5++
                      1`6++1`7

  A, B                3`(1,"COL")++3`(2,"OUR")++
                      3`(3,"ED ")++3`(4,"PET")++
                      3`(5,"RI ")++3`(6,"NET")

  C, D                3`2++3`3++3`4++3`5++3`6++3`7

  Limit               3`()

Best Lower Multi-set Bounds

  PacketsToSend       1`(1,"COL")++1`(2,"OUR")++
                      1`(3,"ED ")++1`(4,"PET")++
                      1`(5,"RI ")++1`(6,"NET")

  DataReceived        empty

  NextSend, NextRec   empty

  A, B, C, D          empty

  Limit               empty
```

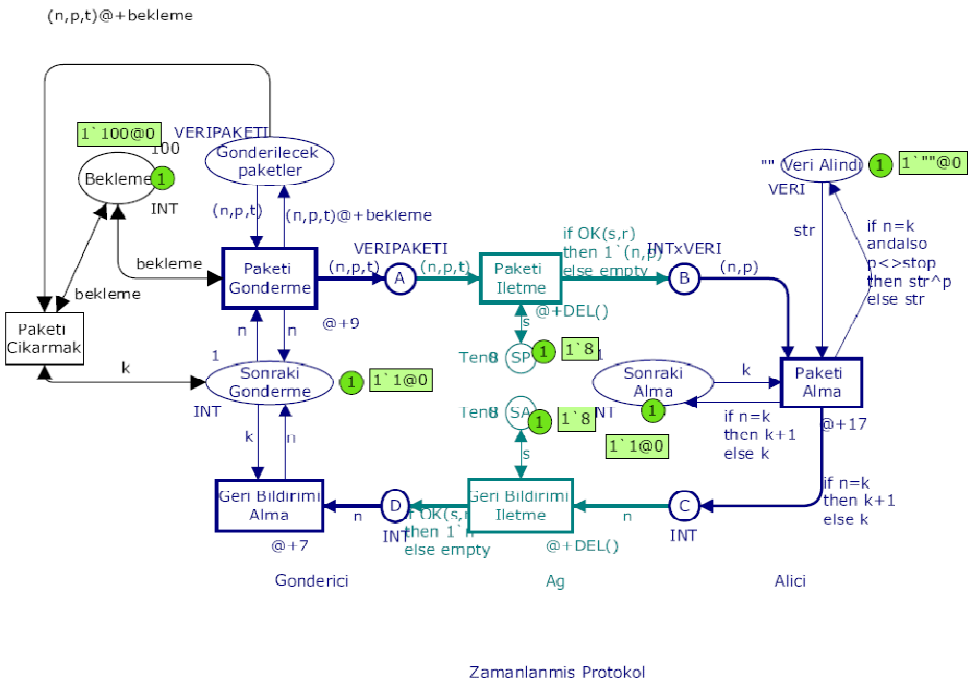
Şekil A.12 : CPN Tools Programının Durum Uzayı Jeton Renkleri ile Gösterilen Sınırlık Özellikleri

Şekil A.9'da tamsayı olarak sınırlamaları görebiliriz. Üst kısım(upper) bir yerdeki en büyük jeton sayısını göstermektedir. Alt kısım(lower) ise bir yerdeki en küçük jeton sayısını göstermektedir.

Şekil A.10'da sadece jetonları sayısı değil rengi de gösterilmiştir.

Ek A.7

Performans analizi için oluşturulan zamanlı protokol modeli şu şekildedir:



Şekil A.13 : Performans Analizi için Oluşturulan Zamanlı Protokol Modeli

Veri paketlerinin gönderici tarafından gönderilmek zorunda olduğunu varsayılsın. Bir sistemin performansını analiz ederken, iş yükünün daha fazla olduğu durumun analizi yapılır. Örneğin, zamanlı protokol örneğinde iş yükü veri paketleridir. Renkli petri ağları ile birçok iş yükü birlikte kullanılabilir. Benzetimin başında iş yükleri önceden belirlenmiştir.

Örnekte, göndericinin veriyi göndermesi ve alıcının veriyi almasında harcanan zaman performans analizi için önem taşımaktadır. Bu nedenle ilk olayın oluştuğu andaki zaman kayıt altına alınmalıdır.

Renkli petri ağları modelinde, en kolay yol jeton değerlerini kaydetmektir. VERIPAKETI renk seti veri paketlerini modellemek için kullanılmaktadır. Veri paketinin jeton rengi, veri içeriğini, sıra numarasını ve verinin ulaşma zamanını içermektedir. Varış zamanı TOA renk seti ile gösterilmektedir.

colset TOA = int;

var t : TOA;

colset VERIPAKETI = product NO*VERI*TOA timed;

Birçok performans ölçümleri vardır. Örneğin kaç veri paketi alındı, bunların kaç tanesi birbirinin aynısı. Paketlerin gönderilme sayısını hesaplamak göndericideki bekleyen veri paketlerini göstermektedir. Ortalama ya da en büyük paket gecikmesini hesaplamak, veri paketlerinin zamanlı olarak alındığını gösterir.

Alıcı tarafından alınan veri paketlerinin sayısının hesaplanması, benzetim sırasında "Paketi Alma" geçişinin ne kadar sürede gerçekleştiği ile ilgilidir. Bunun için CPN Tools programında geçişlerin gerçekleme sayısı monitörü(count transition occurrence monitor) bulunmaktadır. ALINANPAKETLER adı verilen monitor veri paketlerinin sayısını içerir.

$n \neq k$ iken "Paketi Alma" geçişi oluştuğunda aynı iki veri paketi oluşur. Kapsamlı veri toplama monitörü renkli petri ağı modelinden nümerik veriyi toplamak için kullanılır. Bu monitörlerin davranışı kullanıcı tarafından ulaşılabilen fonksiyonlara bağlıdır. Bu monitor aynı paketlerin sayısını içerir ve ALINANAYNIPAKETLER adını alır.

Bir veri toplama monitöründe bir fonksiyon periyodik olarak çağrılmaktadır ve veri toplama işlemi başladığında bu fonksiyonun geri dönüş değeri 1 olur. Fonksiyon şu şekildedir:

fun pred (Protokol'Paketi_Alma

(1,{d,veri,k,n,t})) = true

Bir veri toplama monitöründe gözlem fonksiyonu modelden nümerik değerleri toplar.

fun obs (Protokol'Paketi_Alma (1, {d,veri,k,n,t})) = if n=k then 0 else 1

Yukarıdaki fonksiyon veriyi toplar ve aynı olan veri paketlerini hesaplar. Alıcı tarafından aynı paket alındığı zaman fonksiyon 1 döndürür. Eğer paket ilk kez alınıyorsa 0 döndürür. Bu fonksiyonun döndürdüğü değerler istatistikleri hesaplamaya yardımcı olur. Verilen toplanmasıyla benzetim boyunca kaç tane çift veri paketinin geldiği bulunabilir.

Her veri toplama monitörü iki ek fonksiyona sahiptir. Başlangıç fonksiyonu ve bitiş fonksiyonu. Başlangıç fonksiyonu başlangıç işaretlemesinde veri toplamak için kullanılır. Aynı şekilde bitiş fonksiyonu da son durumdan veri toplamak için kullanılır.

PAKETGECIKMESI isimli yeni bir veri toplama monitörü oluşturulmuştur. Bu monitor zorunlu elemanların gerçekleşmesiyle oluşan veriyi toplar. Ortalama ve maksimum paket gecikmelerini hesaplar. "Paketi Alma" geçişi oluştuğunda t değeri alınan verinin varış zamanına bağlıdır.

Eğer model zamansız ise yerlerdeki ortalama jeton sayısını hesaplamak için veri toplamak için en iyi yol budur.

Zamanlı modeller için yerlerdeki ortalama jeton sayısı hesaplanırken zaman bilgisi kullanılır. Paketi göndermek için ortalama zamanı hesaplarsak, zaman bilgisini veriye yerleştirmiş olur.

Performans ölçümleri benzetim boyunca veri toplama monitörleri ile toplanan veri için değerler hesaplanarak bulunur. Veri toplama monitörleri ile toplanan veriler veri toplama günlük dosyasında tutulur Bu dosya adımlar hakkında bilgi ve verinin toplandığı model zamanını içerir.

Çizelge A.1 GönderilecekPaketler Monitörü İçin Veri Toplama Günlük Dosyası

Data	counter	step	time
0	1	0	0
1	2	1	0
1	3	2	0
0	4	7	169
1	5	8	220
1	6	9	220

Çizelge A.1'de son satır "Gönderilecek Paketler" yerinde bir jeton olduğunu gösterir. Bu 9 adım sonra model zamanın 220 olduğunda ve bu yerdeki jetonlar 6. Kez hesaplandığında 1 jeton olduğunu gösterir. Veri toplama günlük dosyaları hazırlandıktan sonra grafik olarak çizdirilebilir.

Benzetim performans raporu benzetim boyunca veri toplama ile toplanan verileri hesaplar. Benzetim alıcıda 1000 paket aldıktan sonra durur.

Continuous-time statistics

Name	Count	Avrg	Min	Max	First Time
PacketsToSend	2911	1.57	0	7	0
WaitingForTransmission	4223	0.14	0	1	0

Discrete-parameter statistics

Name	Count	Sum	Avrg	Min	Max
PacketDelay	1000	273264	273.26	51	1275
ReceivedDuplicatePackets	1000	102	0.10	0	1
ReceivedPackets	1000	1000	1.00	1	1
Throughput	1	4756.99	4756.99	4756.99	4756.99

Şekil A.14 : Benzetim Performans Raporunun İstatistikleri

Üst kısım sürekli zamanlı istatistikleri gösterir. GONDERILENPAKET monitörü paketlerin gönderildiği ortalama zamanı 1.57 ve en büyük paket numarası 7dir. 2911 veri toplanmış yani, "Gönderilecek Paketler" yerinde 2911 kere jeton görülmüştür. Başlangıç zamanı 0'dır.

Alt kısım ayrık parametre istatistiklerini gösterir. PAKETGECIKMESI monitörünün istatistikleri 1000 veri paketi için paket gecikmesinin hesaplar. Ortalama paket gecikmesi 273.26 birim zaman, en küçük ve en büyük paket gecikmesi 51 ve 1275tir.

Çizelge A.2 Ortalama Paket Gecikmesi İçin Tutulan Günlük Dosyası

Data	counter	data	counter
255.41	1	311.17	6
203.79	2	206.15	7
213.70	3	254.84	8
212.62	4	271.66	9
255.54	5	215.25	10

Yukarıdaki değerler 10 kez benzetim yapıldıktan sonraki değerlerdir. Bu değerlerden bir güven aralığı oluşturulabilir.

EK B.1

OBP'nin DAVET işleminin modelinin durum uzayı raporu şu şekildedir:

Statistics

State Space
Nodes: 12687
Arcs: 67828
Secs: 300
Status: Partial

Scc Graph
Nodes: 9232
Arcs: 43403
Secs: 8

Boundedness Properties

Best Integer Bounds

	Upper	Lower
DAVETMODEL'A 1	1	1
DAVETMODEL'B 1	1	1
DAVETMODEL'Cevaplar 1	9	0
DAVETMODEL'DAVET_Gönderildi 1	1	1
DAVETMODEL'G 1	1	1
DAVETMODEL'H 1	1	1
DAVETMODEL'Istekler 1	5	0
DAVETMODEL'Kullanici 1	1	1
DAVETMODEL'Sunucu 1	1	1
DAVETMODEL'r3xx 1	1	1

Best Upper Multi-set Bounds

DAVETMODEL'A 1	1`500++,1`1000
DAVETMODEL'B 1	1`500++,1`1000
DAVETMODEL'Cevaplar 1	6`[]++,1`[r100]++,4`[r101]++, 4`[r2xx]++, 9`[r3xx]
DAVETMODEL'DAVET_Gonderildi 1	1`0++,1`1
DAVETMODEL'G 1	1`500++,1`1000
DAVETMODEL'H 1	1`500++, 1`1000++, 1`2000++, 1`3000++, 1`4000++, 1`5000++, 1`6000++,1`7000++,1`8000++, 1`9000

DAVETMODEL'Istekler 1	2`DAVET++,4`ALINDI
DAVETMODEL'Kullanici 1	1`cagirma++, 1`ilerleme++, 1`tamamlandi++, 1`sonlandi
DAVETMODEL'Sunucu 1	1`bos++,1`ilerlemeT++, 1`ilerlemeS++,1`onaylandiS++, 1`tamamlandiS++,1`sonlandiS
DAVETMODEL'r3xx 1	1`0

Best Lower Multi-set Bounds

DAVETMODEL'A 1	empty
DAVETMODEL'B 1	empty
DAVETMODEL'Cevaplar 1	empty
DAVETMODEL'DAVET_Gonderildi 1	empty
DAVETMODEL'G 1	empty
DAVETMODEL'H 1	empty
DAVETMODEL'Istekler 1	empty
DAVETMODEL'Kullanici 1	empty
DAVETMODEL'Sunucu 1	empty
DAVETMODEL'r3xx 1	1`0

Home Properties

Home Markings

Initial Marking is not a home marking

Liveness Properties

Dead Markings

6933 [9999,9998,9997,9996,9995,...]

Dead Transition Instances-None

Live Transition Instance-None

Fairness Properties

DAVETMODEL'A_ve_B_zamanlayicileri 1	No Fairness
DAVETMODEL'Cevap_Alindi 1	No Fairness
DAVETMODEL'Cevap_Gonderildi 1	Impartial
DAVETMODEL'G_ve_H_zamanlayicilari 1	No Fairness
DAVETMODEL'Istek 1	No Fairness
DAVETMODEL'Istek_Alindi 1	No Fairness
DAVETMODEL'Istek_Gonderildi 1	No Fairness
DAVETMODEL'Kullanici_Iletim_Hatasi 1	No Fairness
DAVETMODEL'Sunucu_Iletim_Hatasi 1	No Fairness
DAVETMODEL'Zamanlayici_D 1	No Fairness
DAVETMODEL'Zamanlayici_I 1	Just
DAVETMODEL'cevapQ 1	No Fairness

EK B.2

- Marking_size_Protokol'Kullanici_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Mesajın ulaşması
      - Mesajın ulaşması(transition)
    - ✓ Protokol
      - A_ve_B_zamanlayıcıları(transition)
      - Cevap_Alındı(transition)
      - Istek_Gönderildi(transition)
      - Kullanıcı(place)
      - Kullanıcı_İletim_Hatası(transition)
      - Zamanlayıcı_D(transition)
- Marking_size_Protokol'Sunucu_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - Cevap_Gönderildi(transition)
      - G_ve_H_zamanlayıcıları(transition)
      - İstek_Alındı(transition)
      - Sunucu(place)
      - Sunucu_İletim_Hatası(transition)
      - Zamanlayıcı_I(transition)
- Marking_size_Protokol'Cevaplar_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - Cevap_Alındı(transition)
      - Cevap_Gönderildi(transition)
      - Cevap_Kayıp(transition)
      - Cevaplar(place)
      - G_ve_H_zamanlayıcıları(transition)
      - İstek_Alındı(transition)
      - Sunucu_İletim_Hatası(transition)
- Marking_size_Protokol'Istekler_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - A_ve_B_zamanlayıcıları(transition)
      - Cevap_Alındı(transition)
      - İstek_Alındı(transition)

        - Istek_Gönderildi(transition)
        - Istek_Kayıp(transition)
        - İstekler(place)
        - Kullanıcı_İletim_Hatası(transition)
- Marking_size_Protokol'DAVET _Gonderildi_1
    - Type: Marking size
    - Nodes ordered by pages
        - ✓ Protokol
            - A_ve_B_zamanlayıcıları(transition)
            - DAVET_Gönderildi(transition)
            - Istek_Gönderildi(transition)
- Marking_size_Protokol'A_1
    - Type: Marking size
    - Nodes ordered by pages
        - ✓ Protokol
            - A (place)
            - A_ve_B_zamanlayıcıları(transition)
- Marking_size_Protokol'B_1
- Type: Marking size
    - Nodes ordered by pages
        - ✓ Protokol
            - B (place)
            - A_ve_B_zamanlayıcıları(transition)
- Marking_size_Protokol'r3xx_1
    - Type: Marking size
    - Nodes ordered by pages
        - ✓ Mesajın ulaşması
        - ✓ Protokol
- Marking_size_Protokol'r3xx_1
    - Type: Marking size
    - Nodes ordered by pages
        - ✓ Protokol
            - R3xx (place)
            - G_ve_H_zamanlayıcıları(transition)
- Marking_size_Protokol'G_1
    - Type: Marking size
    - Nodes ordered by pages
        - ✓ Protokol
            - G (place)
            - G_ve_H_zamanlayıcıları(transition)
- Marking_size_Protokol'G_1
    - Type: Marking size
    - Nodes ordered by pages
        - ✓ Mesajın ulaşması
        - ✓ Protokol

        - G (place)
        - G_ve_H_zamanlayıcıları(transition)
- Marking_size_Protokol'H_1
    - Type: Marking size
    - Nodes ordered by pages
        - ✓ Mesajın ulaşması
        - ✓ Protokol
            - H(place)
            - G_ve_H_zamanlayıcıları(transition)
- Count_trans_occur_Protokol'Istek_Kayip_1
    - Type: Marking size
    - Nodes ordered by pages
        - ✓ Protokol
            - Istek_Kayip(transition)
- Count_trans_occur_Protokol'Cevap _Kayip_1
    - Type: Marking size
    - Nodes ordered by pages
        - ✓ Protokol
            - Cevap _Kayip(transition)
- Count_trans_occur_Protokol'G ve H _zamanlayicilari_1
    - Type: Marking size
    - Nodes ordered by pages
        - ✓ Protokol
            - G ve H _zamanlayicilari (transition)
- Count_trans_occur_Protokol'Cevap_Gonderildi_1
    - Type: Marking size
    - Nodes ordered by pages
        - ✓ Protokol
            - Cevap_Gonderildi (transition)
- Count_trans_occur_Protokol'Istek_Alindi_1
    - Type: Marking size
    - Nodes ordered by pages
        - ✓ Protokol
            - Istek_Alindi (transition)
- Count_trans_occur_Protokol'Zamanlayici_I_1
    - Type: Marking size
    - Nodes ordered by pages
        - ✓ Protokol
            - Zamanlayici_I (transition)
- Count_trans_occur_Protokol'Sunucu _Iletim_Hatasi_1
    - Type: Marking size
    - Nodes ordered by pages
        - ✓ Protokol
            - Sunucu _Iletim_Hatasi (transition)
- Count_trans_occur_Protokol'Istek_Gonderildi_1

- Type: Marking size
- Nodes ordered by pages
  - ✓ Protokol
    - Istek_Gonderildi (transition)

- Count_trans_occur_Protokol'Kullanici_Iletim_Hatasi_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - Kullanici_Iletim_Hatasi (transition)
- Count_trans_occur_Protokol'Cevap_Alindi_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - Cevap_Alindi (transition)
- Count_trans_occur_Protokol'Zamanlayici_D_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - Zamanlayici_D (transition)
- Count_trans_occur_Protokol'A ve B_zamanlayıcıları_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - A ve B_zamanlayıcıları (transition)
- Group 1
  - Type: Data collection
  - Nodes ordered by pages
    - ✓ Protokol
      - Istekler (place)
      - Cevaplar (place)
  - Predicate

```
fun pred (Protokol'Cevaplar_1_mark : CEVAPQ ms,
          Protokol'Istekler_1_mark : ISTEK ms) =
 true
```

  - Observer

```
fun obs (Protokol'Cevaplar_1_mark : CEVAPQ ms,
          Protokol'Istekler_1_mark : ISTEK ms) =
 (size Protokol'Istekler_1_mark) +
(size Protokol'Cevaplar_1_mark );
```

  - Init Function

```
fun init (Protokol'Cevaplar_1_mark : CEVAPQ ms,
          Protokol'Istekler_1_mark : ISTEK ms) =
 NONE
```

  - Stop

```
fun stop (Protokol'Cevaplar_1_mark : CEVAPQ ms,
          Protokol'Istekler_1_mark : ISTEK ms) =  NONE
```

EK B.3

- Marking_size_Protokol'Kullanıcı_1 :

#data	counter	step	time	#data	counter	step	time
0	1	0	0	1	5	9	0
1	2	2	0	1	6	15	1500
1	3	3	0	1	7	19	1500
1	4	6	0	1	8	50	16500

- Marking_size_Protokol'Sunucu_1: Sunucu yerindeki jetonları sayar.

#data	counter	step	time	#data	counter	step	time	#data	counter	step	time
1	1	0	0	1	9	17	1500	1	17	35	9500
1	2	1	0	1	10	21	2500	1	18	37	10500
1	3	7	0	1	11	23	3500	1	19	39	11500
1	4	8	0	1	12	25	4500	1	20	41	12500
1	5	11	500	1	13	27	5500	1	21	43	13500
1	6	13	1500	1	14	29	6500	1	22	45	14500
1	7	14	1500	1	15	31	7500	1	23	47	15500
1	8	16	1500	1	16	33	8500	1	24	49	16500
								1	25	50	16500

- Marking_size_Protokol'Cevaplar_1:

#data	counter	step	time	#data	counter	step	time	#data	counter	step	time
0	1	0	0	0	17	22	2500	0	33	38	10500
1	2	1	0	1	18	23	3500	1	34	39	11500
0	3	4	0	0	19	24	3500	0	35	40	11500
1	4	7	0	1	20	25	4500	1	36	41	12500
1	5	8	0	0	21	26	4500	0	37	42	12500
1	6	9	0	1	22	27	5500	1	38	43	13500
0	7	10	0	0	23	28	5500	0	39	44	13500
1	8	11	500	1	24	29	6500	1	40	45	14500
0	9	12	500	0	25	30	6500	0	41	46	14500

1	10	13	1500	1	26	31	7500	1	42	47	15500
1	11	14	1500	0	27	32	7500	0	43	48	15500
1	12	15	1500	1	28	33	8500	1	44	49	16500
2	13	16	1500	0	29	34	8500	0	45	50	16500
1	14	18	1500	1	30	35	9500	0	46	50	16500
0	15	20	1500	0	31	36	9500				
1	16	21	2500	1	32	37	10500				

- Making_size_Protokol'Istekler_1:

#data	counter	step	time	#data	counter	step	time	#data	counter	step	time
0	1	0	0	1	4	6	0	1	7	15	1500
1	2	3	0	0	5	7	0	0	8	16	1500
0	3	5	0	0	6	9	0	0	9	50	16500

- Marking_size_Protokol'DAVET _Gonderildi_1:

#data	counter	step	time	#data	counter	step	time
1	1	0	0	1	3	6	0
1	2	3	0	1	4	50	16500

- Marking_size_Protokol'A_1:

#data	counter	step	time	#data	counter	step	time
1	1	0	0	1	3	50	16500
1	2	6	0				

- Marking_size_Protokol'B_1 :

#data	counter	step	time	#data	counter	step	time
1	1	0	0	1	3	50	16500
1	2	6	0				

- Marking_size_Protokol'r3xx_1 :

#data	counter	step	time	#data	counter	step	time	#data	counter	step	time
1	1	0	0	1	8	27	5500	1	15	41	12500
1	2	1	0	1	9	29	6500	1	16	43	13500
1	3	11	500	1	10	31	7500	1	17	45	14500
1	4	13	1500	1	11	33	8500	1	18	47	15500
1	5	21	2500	1	12	35	9500	1	19	49	16500
1	6	23	3500	1	13	37	10500	1	20	50	16500
1	7	25	4500	1	14	39	11500				

- Marking_size_Protokol'G_1:

#data	counter	step	time	#data	counter	step	time	#data	counter	step	time
1	1	0	0	1	8	27	5500	1	15	41	12500
1	2	1	0	1	9	29	6500	1	16	43	13500
1	3	11	500	1	10	31	7500	1	17	45	14500
1	4	13	1500	1	11	33	8500	1	18	47	15500
1	5	21	2500	1	12	35	9500	1	19	49	16500
1	6	23	3500	1	13	37	10500	1	20	50	16500
1	7	25	4500	1	14	39	11500				

- Marking_size_Protokol'H_1:

#data	counter	step	time	#data	counter	step	time	#data	counter	step	time
1	1	0	0	1	8	27	5500	1	15	41	12500
1	2	1	0	1	9	29	6500	1	16	43	13500
1	3	11	500	1	10	31	7500	1	17	45	14500
1	4	13	1500	1	11	33	8500	1	18	47	15500
1	5	21	2500	1	12	35	9500	1	19	49	16500
1	6	23	3500	1	13	37	10500	1	20	50	16500
1	7	25	4500	1	14	39	11500				

- Cout_trans_occur_Protokol'Istek_Kayip_1:

#data	counter	step	time
1	1	5	0

- Count_trans_occur_Protokol'Cevap_Kayip_1:

#data	counter	step	time	#data	counter	step	time	#data	counter	step	time
1	1	4	0	1	8	26	4500	1	15	40	11500
1	2	10	0	1	9	28	5500	1	16	42	12500
1	3	12	500	1	10	30	6500	1	17	44	13500
1	4	18	1500	1	11	32	7500	1	18	46	14500
1	5	20	1500	1	12	34	8500	1	19	48	15500
1	6	22	2500	1	13	36	9500	1	20	50	16500
1	7	24	3500	1	14	38	10500				

- Count_trans_occur_Protokol'G ve H_zamanlayicilari_1:

#data counter step time #data counter step time #data counter step time

1	1	1	0	1	7	27	5500	1	13	39	11500
1	2	11	500	1	8	29	6500	1	14	41	12500
1	3	13	1500	1	9	31	7500	1	15	43	13500
1	4	21	2500	1	10	33	8500	1	16	45	14500
1	5	23	3500	1	11	35	9500	1	17	47	15500
1	6	25	4500	1	12	37	10500	1	18	49	16500

- Count_trans_occur_Protokol'Cevap_Gonderildi_1:

#data counter step time

1	1	8	0
1	2	14	1500

- Count_trans_occur_Protokol'Istek_Alindi_1:

#data counter step time

1	1	7	0
1	2	16	1500

- Count_trans_occur_Protokol'Zamanlayici_I_1:

#data counter step time

1	1	17	1500

- Count_trans_occur_Protokol'Sunucu _Iletim_Hatasi_1: -
- Count_trans_occur_Protokol'Istek_Gonderildi_1:

#data counter step time

1	1	3	0

- Count_trans_occur_Protokol'Kullanici_Iletim_Hatasi_1: -
- Count_trans_occur_Protokol'Cevap_Alindi_1 :

#data counter step time

1	1	9	0
1	2	15	1500

- Count_trans_occur_Protokol'Zamanlayici_D_1:

#data counter step time

1	1	19	1500

- Count_trans_occur_Protokol'A ve B_zamanlayıcıları_1:

#data	counter	step	time
1	1	6	0

- Group 1:

#data	counter	step	time	#data	counter	step	time	#data	counter	step	time
1	1	1	0	1	18	18	1500	1	35	35	9500
1	2	2	0	1	19	19	1500	0	36	36	9500
2	3	3	0	0	20	20	1500	1	37	37	10500
1	4	4	0	1	21	21	2500	0	38	38	10500
0	5	5	0	0	22	22	2500	1	39	39	11500
1	6	6	0	1	23	23	3500	0	40	40	11500
1	7	7	0	0	24	24	3500	1	41	41	12500
1	8	8	0	1	25	25	4500	0	42	42	12500
1	9	9	0	0	26	26	4500	1	43	43	13500
0	10	10	0	1	27	27	5500	0	44	44	13500
1	11	11	500	0	28	28	5500	1	45	45	14500
0	12	12	500	1	29	29	6500	0	46	46	14500
1	13	13	1500	0	30	30	6500	1	47	47	15500
1	14	14	1500	1	31	31	7500	0	48	48	15500
2	15	15	1500	0	32	32	7500	1	49	49	16500
2	16	16	1500	1	33	33	8500	0	50	50	16500
2	17	17	1500	0	34	34	8500				

EK B.4

CPN Tools state space report for:
Report generated: Sat Apr 10 10:03:51 2010

Statistics

State Space
Nodes: 43038
Arcs: 279558
Secs: 300
Status: Partial

Scc Graph
Nodes: 36618
Arcs: 264586
Secs: 75

Boundedness Properties

Best Integer Bounds	Upper	Lower
model'Cevaplar 1	139	0
model'E 1	1	1
model'F 1	1	1
model'Istekler 1	1	0
model'Kullanici 1	1	1
model'Sunucu 1	1	1
model'y 1	1	1

Best Upper Multi-set Bounds

model'Cevaplar 1	1`msjyok++, 1`r100++,137`r101++,1`r2xx++,1`r3xx
model'E 1	1`500++,1`1000
model'F 1	1`500++,1`1000
model'Istekler 1	1`[istekM]++,1`[istekM,istekM]
model'Kullanici 1	1`deneniyor++,1`ilerleme++,1`tamamlandi++, 1`sonlandi
model'Sunucu1	1`deneniyorS++, 1`ilerlemeT++,1`ilerlemeS++, 1`tamamlandiS++, 1`sonlandiS,
model'y 1	1`0++,1`1

Best Lower Multi-set Bounds

model'Cevaplar 1	empty
model'E 1	empty
model'F 1	empty
model'Istekler 1	empty
model'Kullanici 1	empty
model'Sunucu 1	empty
model'y 1	empty

Home Properties

Home Markings

Initial Marking is not a home marking

Liveness Properties

Dead Markings

322 [739,72,60,435,43038,...]

Dead Transition Instances-None

Live Transition Instances- None

Fairness Properties

---

model'Cevap_Alindi 1	No Fairness
model'Cevap_Gonderildi 1	Impartial
model'Cevap_Kayip 1	No Fairness
model'E_ve_F_Zamanlayicilari 1	No Fairness
model'Istek_Alindi 1	Fair
model'Istek_Gonderildi 1	Fair
model'Istek_Kayip 1	No Fairness
model'J_Zamanlayicisi 1	Fair
model'K_Zamanlayicisi 1	Fair
model'Kullanici_Iletim_Hatasi 1	No Fairness
model'Sunucu_Iletim_Hatasi 1	No Fairness

EK B.5

- Marking_size_Protokol'Kullanici_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Oturumun Başlaması
      - Oturumun Başlaması (transition)
    - ✓ Protokol
      - E_ve_F _zamanlayıcıları(transition)
      - Cevap_Alındı(transition)
      - Istek_Gönderildi(transition)
      - Kullanıcı(place)
      - Kullanıcı_İletim_Hatası(transition)
      - K Zamanlayıcısı(transition)
- Marking_size_Protokol'Sunucu_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - Cevap_Gönderildi(transition)
      - İstek_Alındı(transition)
      - Sunucu(place)
      - Sunucu_İletim_Hatası(transition)
      - J zamanlayıcısı(transition)
- Marking_size_Protokol'Cevaplar_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - Cevap_Alındı(transition)
      - Cevap_Gönderildi(transition)
      - Cevap_Kayıp(transition)
      - Cevaplar(place)
      - Cevap_Alındı(transition)
      - Sunucu_İletim_Hatası(transition)
- Marking_size_Protokol'Istekler_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - İstek_Alındı(transition)
      - Istek_Gönderildi(transition)
      - Istek_Kayıp(transition)
      - İstekler(place)

- Marking_size_Protokol'E_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - E (place)
      - E_ve_F_zamanlayıcıları(transition)
- Marking_size_Protokol'F_1

  Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - F (place)
      - E_ve_F_zamanlayıcıları(transition)
- Count_trans_occur_Protokol'Istek_Kayip_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - Istek_Kayip(transition)
- Count_trans_occur_Protokol'Cevap _Kayip_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - Cevap _Kayip(transition)
- Count_trans_occur_Protokol'E ve F _zamanlayicilari_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - E ve F _zamanlayicilari (transition)
- Count_trans_occur_Protokol'Cevap_Gonderildi_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - Cevap_Gonderildi (transition)
- Count_trans_occur_Protokol'Istek_Alindi_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - Istek_Alindi (transition)
- Count_trans_occur_Protokol'J_Zamanlayicisi_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - Zamanlayici_I (transition)

- Count_trans_occur_Protokol'Sunucu _Iletim_Hatasi_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - Sunucu _Iletim_Hatasi (transition)
- Count_trans_occur_Protokol'Istek_Gonderildi_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - Istek_Gonderildi (transition)
- Count_trans_occur_Protokol'Kullanici_Iletim_Hatasi_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - Kullanici_Iletim_Hatasi (transition)
- Count_trans_occur_Protokol'Cevap_Alindi_1
  - Type: Marking size
  - Nodes ordered by pages
    - ✓ Protokol
      - Cevap_Alindi (transition)

- Group 1
  - Type: Data collection
  - Nodes ordered by pages
    - ✓ Protokol
      - Istekler (place)
      - Cevaplar (place)
  - Predicate

```
fun pred (Protokol'Cevaplar_1_mark : CEVAPQ ms,
          Protokol'Istekler_1_mark : ISTEK ms) =
 true
```

  - Observer

```
fun obs (Protokol'Cevaplar_1_mark : CEVAPQ ms,
         Protokol'Istekler_1_mark : ISTEK ms) =
 (size Protokol'Istekler_1_mark) +
(size Protokol'Cevaplar_1_mark );
```

  - Init Function

```
fun init (Protokol'Cevaplar_1_mark : CEVAPQ ms,
          Protokol'Istekler_1_mark : ISTEK ms) =
 NONE
```

  - Stop

```
fun stop (Protokol'Cevaplar_1_mark : CEVAPQ ms,
          Protokol'Istekler_1_mark : ISTEK ms) =
 NONE
```

EK B.6

- Marking_size_Protokol'Kullanici_1 :

#data	counter	step	time	#data	counter	step	time
0	1	0	0	1	4	7	0
1	2	4	0	1	5	50	0
1	3	6	0				

- Marking_size_Protokol'Istekler_1 :

#data	counter	step	time	#data	counter	step	time
1	1	0	0	0	3	50	0
0	2	1	0				

- Marking_size_Protokol'E_1 :

#data	counter	step	time
0	1	0	0
0	2	50	0

- Marking_size_Protokol'F_1 :

#data	counter	step	time
0	1	0	0
0	2	50	0

- Marking_size_Protokol'Sunucu_1 :

#data	counter	step	time	#data	counter	step	time	#data	counter	step	time
1	1	0	0	1	11	17	0	1	21	35	0
1	2	3	0	1	12	18	0	1	22	37	0
1	3	5	0	1	13	21	0	1	23	42	0
1	4	8	0	1	14	22	0	1	24	44	0
1	5	9	0	1	15	25	0	1	25	45	0
1	6	10	0	1	16	27	0	1	26	46	0
1	7	11	0	1	17	29	0	1	27	48	0
1	8	12	0	1	18	30	0	1	28	49	0
1	9	13	0	1	19	32	0	1	29	50	0

- Marking_size_Protokol'Cevaplar_1 :

#data	counter	step	time	#data	counter	step	time	#data	counter	step	time
1	1	0	0	5	17	20	0	3	33	36	0
0	2	2	0	6	18	21	0	4	34	37	0
1	3	3	0	7	19	22	0	3	35	38	0
2	4	5	0	6	20	23	0	2	36	39	0
1	5	6	0	5	21	24	0	1	37	40	0
2	6	8	0	6	22	25	0	0	38	41	0
3	7	9	0	5	23	26	0	1	39	42	0
4	8	10	0	6	24	27	0	0	40	43	0
5	9	11	0	5	25	28	0	1	41	44	0
6	10	12	0	6	26	29	0	2	42	45	0
7	11	13	0	5	27	30	0	1	43	46	0
8	12	14	0	4	28	31	0	0	44	47	0
7	13	15	0	5	29	32	0	1	45	48	0
6	14	16	0	4	30	33	0	0	46	49	0
7	15	17	0	3	31	34	0	1	47	50	0
6	16	19	0	4	32	35	0	1	48	50	0

- Count_trans_occur_Protokol'Kullanici_Iletim_Hatasi_1:

#data	counter	step	time
1	1	7	0

- Count_trans_occur_Protokol'Cevap _Alindi_1 :

#data	counter	step	time
1	1	6	0

- Count_trans_occur_Protokol'Istek _Kayip_1 :

#data	counter	step	time
1	1	1	0

- Count_trans_occur_Protokol'Cevap_Gonderildi_1:

#data	counter	step	time	#data	counter	step	time	#data	counter	step	time
1	1	3	0	1	9	14	0	1	17	35	0
1	2	5	0	1	10	17	0	1	18	37	0
1	3	8	0	1	11	21	0	1	19	42	0
1	4	9	0	1	12	22	0	1	20	44	0

1	5	10	0	1	13	25	0	1	21	45	0
1	6	11	0	1	14	27	0	1	22	48	0
1	7	12	0	1	15	29	0	1	23	50	0
1	8	13	0	1	16	32	0				

- Count_trans_occur_Protokol'Cevap Kayip_1:

#data	counter	step	time	#data	counter	step	time	#data	counter	step	time
1	1	2	0	1	7	24	0	1	13	38	0
1	2	15	0	1	8	26	0	1	14	39	0
1	3	16	0	1	9	28	0	1	15	40	0
1	4	19	0	1	10	31	0	1	16	41	0
1	5	20	0	1	11	34	0	1	17	43	0
1	6	23	0	1	12	36	0	1	18	47	0

- Count_trans_occur_Protokol'Sunucu _Iletim_Hatasi_1:

#data	counter	step	time	#data	counter	step	time
1	1	30	0	1	3	46	0
1	2	33	0	1	4	49	0

- Count_trans_occur_Protokol'J_Zamanlayicisi_1:

#data	counter	step	time
1	1	18	0

- Group 1:

#data	counter	step	time	#data	counter	step	time	#data	counter	step	time
1	1	1	0	7	18	18	0	4	35	35	0
0	2	2	0	6	19	19	0	3	36	36	0
1	3	3	0	5	20	20	0	4	37	37	0
1	4	4	0	6	21	21	0	3	38	38	0
2	5	5	0	7	22	22	0	2	39	39	0
1	6	6	0	6	23	23	0	1	40	40	0
1	7	7	0	5	24	24	0	0	41	41	0
2	8	8	0	6	25	25	0	1	42	42	0
3	9	9	0	5	26	26	0	0	43	43	0
4	10	10	0	6	27	27	0	1	44	44	0
5	11	11	0	5	28	28	0	2	45	45	0
6	12	12	0	6	29	29	0	1	46	46	0
7	13	13	0	5	30	30	0	0	47	47	0
8	14	14	0	4	31	31	0	1	48	48	0
7	15	15	0	5	32	32	0	0	49	49	0
6	16	16	0	4	33	33	0	1	50	50	0
7	17	17	0	3	34	34	0				

EK B.7

elemansayilari:
24
2
15
14

nmatrisleri:
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
…..

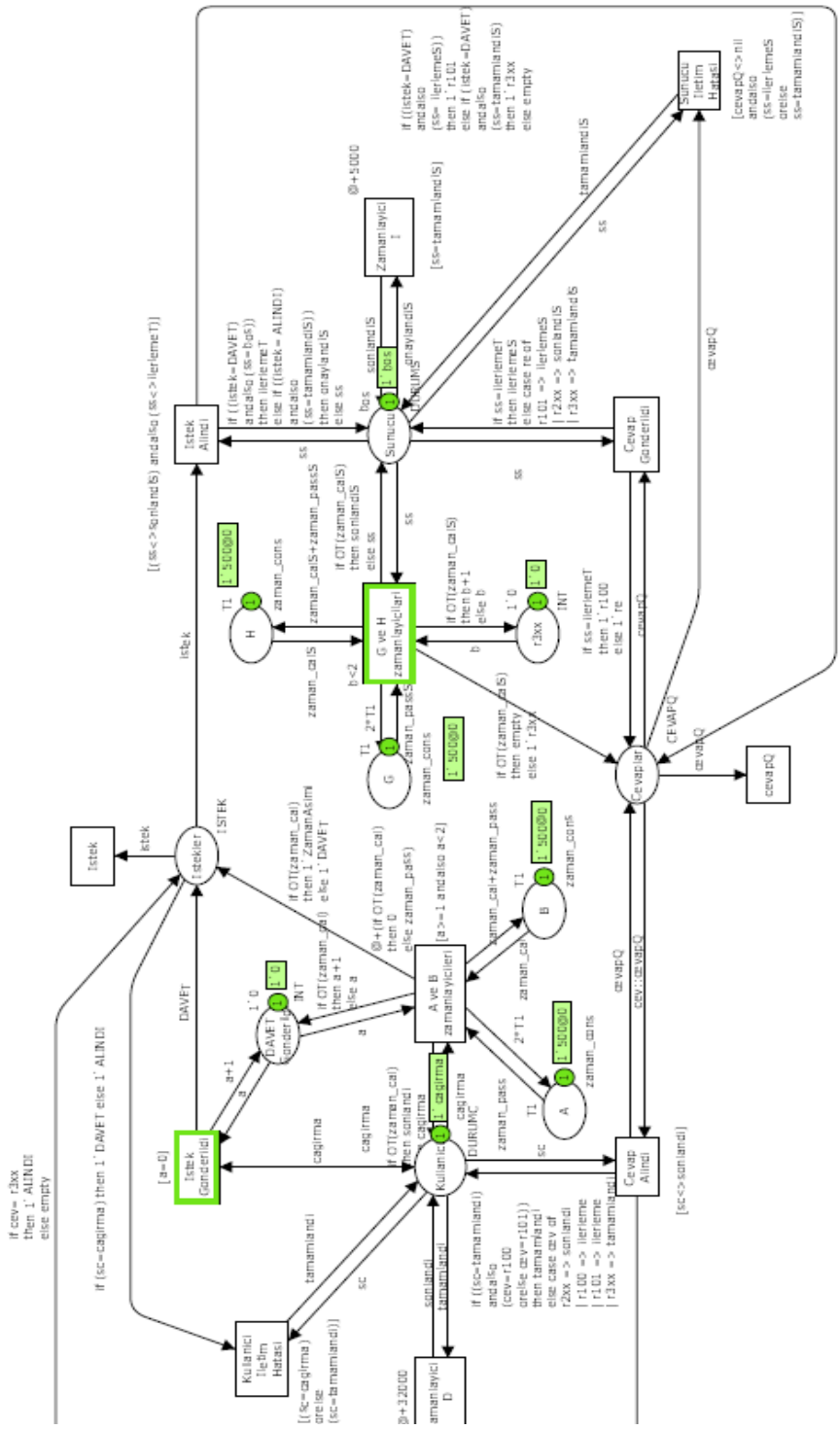
omatrisleri:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
…

zaman:
0
0
0
0
0
0
0
0
0
0
0
0
0

Mo:
1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

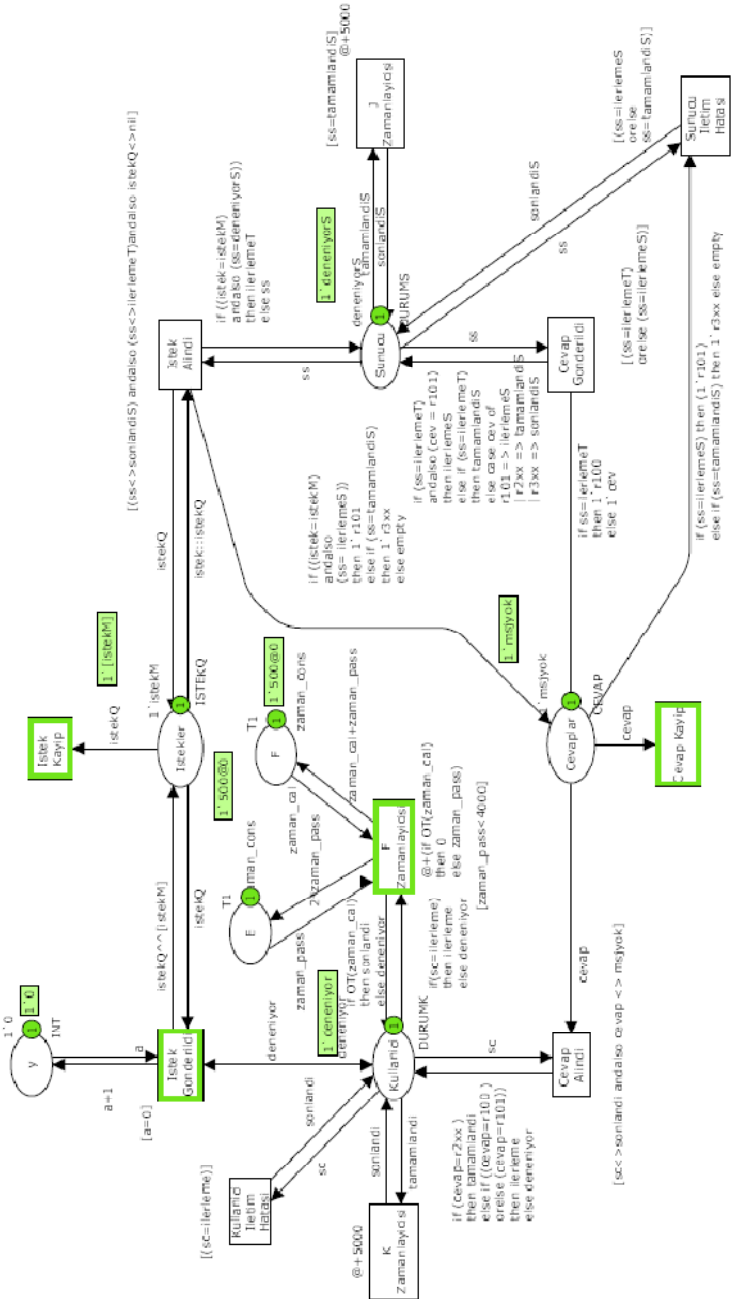
EK C.1

OBP'nin DAVET işleminin modeli aşağıdaki gibidir.



EK C.2

OBP'nin DAVETsiz işleminin modeli aşağıdaki gibidir.



ÖZGEÇMİŞ

Ad Soyad: Safiye KIZMAZ

Doğum Yeri ve Tarihi: İzmit / 16.10.1984

Adres: Esenkent mah. 4 Eylül Sok. No:29 B Blok D:8 Maltepe/İSTANBUL

Lisans Üniversite: Yıldız Teknik Üniversitesi

Yayın Listesi:

- Kızmaz S., Kırcı M., Verification of Session Initiation Protocol Using Timed Colored Petri Net. *The 6th International Conference on Wireless Communications, Networking and Mobile Computing*, September 23-25, 2010. Chengdu, China. (Kabul edilmiştir.)