Ethernet Üzerinde Dinamik, Dağıtılmış Ve Güvenilir Endüstriyel Haberleşme Protokolları: Genel Tasarım İşçerçevesi, Gerçekleştirim Ve Deneysel Çalışma

Proje No: 109E261

Doç. Dr. Ece Schmidt

Mart 2013
Ankara

ÖNSÖZ

TÜBİTAK tarafından Kariyer Programı çerçevesinde desteklenen bu projede, endüstriyel sistemler için standart paylaşımlı Ethernet üzerinde gerçek zaman garantileri sağlayan haberleşme protokolları geliştirimi için yeni ve pek çok üstünlükleri olan bir yaklaşımla formal bir işçerçevesi oluşturulmuştur. Bu işçerçevesine uygun olarak D^3RIP (Dynamic Distributed Dependable Real-time Industrial Protocol) (Dinamik Dağıtık Güvenilir Gerçek Zamanlı Endüstriyel İletişim Protokolu (D^2G^2EP)) protokol yığıtı geliştirilmış, gerçekleştirilmiş ve başarısı incelenmiştir.

Proje Orta Doğu Teknik Üniversitesi Elektrik Elektronik Mühendisliği Bölümü (ODTU-EEMB) tarafından labaratuvar alanı sağlanarak destesteklenmiştir.

Projede ODTÜ-EEMB öğrencisi olan toplam 4 yarı zamanlı bursiyer (bir doktora üç yüksek lisans) yanısıra bir BİDEB burslusu ve bir de gönüllü yüksek lisans öğrencisi çalışmıştır. Bu öğrencilerin tezleri tamamen proje kapsamında yaptıkları çalışmalardan oluşmaktadır. Proje kapsamında Almanya'da Erlangen Üniversitesi ve Çankaya Üniversitesi'nden araştırmacılarla ortak çalışmalar yapılmıştır. Proje çıktıları dergi makalesi ve bildiri olarak yayınlanmış ve halen yayınlanma sürecindedir.

Proje yöneticisi kariyerinin gelişmesine çok önemli katkıları olan bu proje desteği için TÜBİTAK'a teşekkür eder.

İçindekiler

Şekil Listesi

Tablo Listesi

ÖZET

Günümüz otomasyon sistemlerinde kontrol uygulamaları pek çok sayısal cihaz üzerinde dağıtılmış gömülü sistemler olarak gerçekleştirilmektedir. Bu cihazların koordinasyonu ve haberleşmesi endüstriyel haberleşme ağları üzerinden yapılmaktadır. Kontrol uygulamalarının gereklerine göre periyodik veya rasgele mesajlar gönderilmekte ve bu mesajlar gerçek zamanlı garantilere ihtiyaç duyabilmektedir. Buna ek olarak otomasyon sistemlerinin uzaktan idaresi ve çalışmalarının izlenmesi için gerçek zamanlı olmayan mesajların da ağ üzerinde taşınması gereklidir. Endüstriyel ağlar üzerlerinde çalışan kontrol uygulamaları insan hayatına doğrudan etki edebilecek özelliklerdedirler. Bu ağların her koşulda tanımlandıkları şekilde çalışmaları ve gerçek zamanlı garantileri sağlamaları gerekmektedir. Bu nedenle bu ağlar için protokolların formal olarak geliştirilebilmesi, incelenebilmesi ve çalışmalarının doğrulanabilmesi önemlidir. Literatürdeki endüstriyel ağ protokolları sistematik bir yaklaşım olmadan uygulama ihtiyaçları doğrultusunda ev ofis ağları için olan yaklaşımlara ekler ve değişiklikler yaparak, kontrol uygulamalarının davranışı hakkındaki bilgilerden faydalanmadan geliştirilmişlerdir. Pek çok var olan protokol için güvenirlik desteği bulunmamaktadır. Günümüzde ev ofis ağlarında çok yaygın olarak kullanılan fakat gerçek zamanlı haberleşmeyi desteklemeyen Standart Ethernet teknolojisinin endüstriyel haberleşme ağlarında kullanılması önemli bir araştırma konusudur.

Bu projede yukarıdaki gereklere uygun olarak, gerçek zamanlı paylaşımlı ortam haberleşme protokolları için formal bir yaklaşımla genel bir işçerçevesi oluşturulmuş ve bu işçerçevesine uygun olarak bir protokol yığıtı geliştirilmiş ve gerçekleştirilmiştir.

Projede geliştirilen formal işçerçevesi ve buna uygun olarak yapılan protokol gerçekleştirimi tamamen dağıtılmış bir mimariye sahiptir. Endüstriyel kontrol sistemlerinde gönderilen mesajların kontrol sistem modeline göre önceden belirlenmesi mümkündür. Bu özellikten yararlanarak ağ kaynakları anlık gerçek zamanlı haberleşme ihtiyaçlarına göre dinamik bir biçimde düğümlere dağıtılmakta ve artan ağ kapasitesi gerçek zamanlı olmayan trafik için kullanılmaktadır. Son olarak mimaride olası hata durumlarına karşı geri toplama mekanizmaları da mevcuttur. Bu özelliklerinden dolayı geliştirdiğimiz formal işçerçevesine ve bu işçerçevesine göre gerçekleştirilen üç katmanlı protokol yığıtına D^3RIP (Dynamic Distributed Dependable Real-time Industrial Protocol) adı verilmiştir.

Projemizin çıktısı olan formal işçerçevesi bu katmanların ve katman arayüzlerinin Timed IO Automata (TIOA) yöntemi ile ayrık olarak modellenmesi ve bu yönetmin sağladığı çeşitli özellikler ve operasyonlar kullanılarak bütün katman yığıtı modelinin elde edilmesine dayalıdır. Analitik olarak oluşturulan TIOA modelleri daha sonra UPPAAL yazılım aracıyla da benzetim yoluyla doğrulanmıştır. Bu aşamada TIOA modellerini UPPAAL modellerine algoritmik çeviren bir yöntem geliştirilmıştır.

Formal işçerçevesine uygun olarak tamamen standart Ethernet donanımı üzerinde, farklı platformlara taşınabilir bir D^3RIP Katman mimarisi gerçeklemesi yapılmış ve başarımı deneylerle incelenmiştir.

Proje kapsamında 1 dergi makalesi, 2 konferans bildirisi yayınlanmıştır. İki dergi makalesi hakem değerlendirmesindedir. Proje kapsamında üç yüksek lisans tezi tamamlanmıştır. 2 yüksek lisans ve bir doktora tezi 2013 bahar dönemi sonunda tamamlanacaktır.

ABSTRACT

The control applications of today's automation systems, are distributed over many devices which communicate and coordinate via industrial communication networks. The control applications generate periodic or sporadic messages which are mostly real-time. In addition, there are non-real-time messages for remote administration and work monitoring of automated systems. Networked control applications have the ability to directly affect people's lives and require defined operations in all conditions and real time service guarantees. Hence, formal development and verification of industrial network protocols are important. Industrial network protocols in the literature are developed without a systematic approach by modifying the solutions for home and office networks without using the behavioral information of control applications. Furthermore, many industrial communication protocols do not support dependability. Today, Standard Ethernet, is very commonly used in home and office networks but does not support real time communications. Ethernet for industrial communication is an important research topic.

In this project, in accordance with the above requirements, a general framework for real time shared medium communication protocols is created with a formal approach and a protocol stack is developed and implemented according to this framework.

The formal framework developed in this project and the protocol stack implementation have a fully distributed architecture. In industrial control systems, it is possible to predetermine the transmitted control messages according to control system model. Taking the advantage of thisproperty, network resources are dynamically distributed to nodes according to the needs of instant real time communications and the increased network capacity is used for non-real-time traffic. Finally, the architecture features roll back mechanisms against potential error conditions. Hence, the developed formal framework and accordingly implemented three-layered protocol stack are called D^3RIP (Dynamic Distributed Dependable Real-time Industrial Protocol).

The formal framework is based on individual modelling of the layers and layer interfaces via Timed IO Automata (TIOA) method and composing the stack model Analytically generated TIOA models are later verified by the software tool UPPAAL via simulation. In this step, a method which converts TIOA models to UPPAAL models is developed.

A D^3RIP implementation that is portable on different platforms, is implemented over standard Ethernet hardware according to the formal framework and experimentally evaluated. Within the scope of this project, 1 journal and 2 conference papers are published. 2 submitted journal papers are under review. 3 M.Sc. thesis are completed and 2 M.Sc. and 1 Ph.D. thesis will be completed by the end of 2013 spring term.

1 Giriş

Günümüz otomasyon sistemlerinde kontrol uygulamaları pek çok sayısal cihaz üzerinde dağıtılmış gömülü sistemler olarak gerçekleştirilmektedir. Bu cihazların koordinasyonu ve haberleşmesi endüstriyel haberleşme ağları üzerinden yapılmaktadır. Kontrol uygulamalarının gereklerine göre periyodik veya rasgele mesajlar gönderilmekte ve bu mesajlar gerçek zamanlı garantilere ihtiyaç duyabilmektedir. Buna ek olarak otomasyon sistemlerinin uzaktan idaresi ve çalışmalarının izlenmesi için gerçek zamanlı olmayan mesajların da ağ üzerinde taşınması gereklidir. Günümüzde Ethernet teknolojisinin endüstriyel haberleşme ağlarında kullanılması endüstriyel ve akademik ortamlarda önemli bir araştırma konusudur. Standart Ethernet düşük maliyetli ve hızlı olması nedeni ile ev ofis ağlarında çok yaygın olarak kullanılmaktadır. Buna karşın Ethernet üzerinden ortama deterministik erişim olmaması nedeni ile otomasyon sistemleri için gerçek zamanlı iletişim garantilerini sağlamamaktadır.

Bu projenin konusu endüstriyel uygulamalar için gerçek zamanlı paylaşımlı ortam Ethernet haberleşmesidir. Projenin başlıca iki amacı vardır:

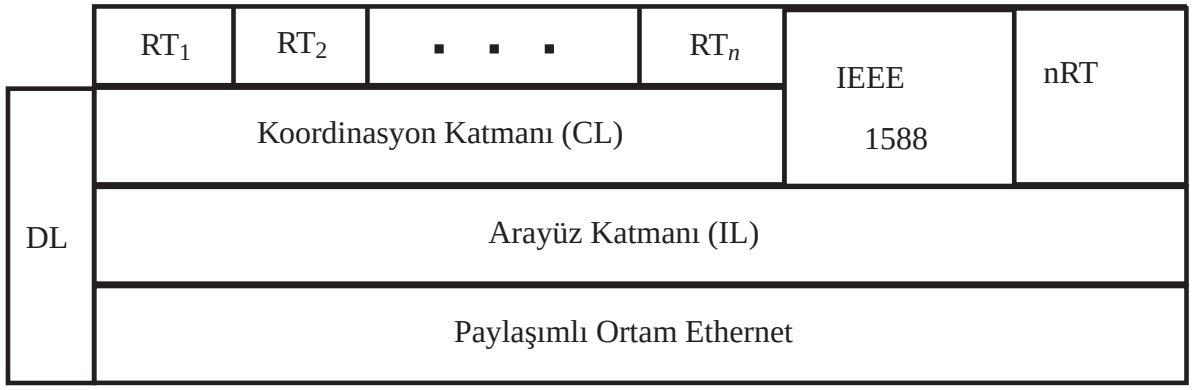
1. Gerçek zamanlı paylaşımlı ortam haberleşme protokolları için formal bir yaklaşımla genel bir işçerçevesi oluşturmak
2. Bu modele uygun olarak bir protokol yığıtı geliştirmek, gerçekleştirmek ve deneysel olarak başarımını ölçmek

Projede geliştirilen formal işçerçevesi ve buna uygun olarak yapılan protokol gerçekleştirimi tamamen *dağıtılmış* bir mimariye sahiptir. Bu mimaride ağ kaynakları anlık gerçek zamanlı haberleşme ihtiyaçlarına göre *dinamik* bir biçimde düğümlere dağıtılmakta ve artan ağ kapasitesi gerçek zamanlı olmayan trafik için kullanılmaktadır. Son olarak mimaride olası hata durumlarına karşı geri toplama mekanizmaları da mevcuttur. Bu özelliklerinden dolayı geliştirdiğimiz formal işçerçevesine göre gerçekleştirilen protokol yığıtına D^3RIP (Dynamic Distributed Dependable Real-time Industrial Protocol) (Dinamik Dağıtık Güvenilir Gerçek Zamanlı Endüstriyel İletişim Protokolu (D^2G^2EP)) adı verilmiştir.

Proje sürecinde geliştirilen iki katmanlı gerçek zamanlı endüstriyel haberleşme mimarisi, D^2RIP (Dynamic Distributed Real-time Industrial communication Protocol) (Dinamik Dağıtık Gerçek Zamanlı Endüstriyel İletişim Protokolu (D^2GEP)) bu mimarinin güvenirlik katmanı ile birlikte son hali, D^3RIP (Dynamic Distributed Dependable Real-time Industrial communication Protocol) (Dinamik Dağıtık Güvenilir Gerçek Zamanlı Endüstriyel İletişim Protokolu (D^2G^2EP)) olarak adlandırılmaktadır.

D^3RIP Katman Mimarisi paylaşımlı ortama çarpışma olmadan erişimi sağlayan bir Arayüz Katmanı (AK) (Interface Layer-IL), her düğümde uygulamanın haberleşme modeline göre güncel zaman diliminde hangi düğümün mesaj göndereceğini dağıtık bir hesaplamayla bütün düğümlerde tutarlı olarak belirleyen bir Koordinasyon Katmanı (KK) (Coordination Layer-CL) ve düğümlerdeki tutarlılık gereklerinden faydalanarak olabilecek hataları tespit eden ve bütün düğümleri hatasız bir zaman dilimine geri döndürebilen bir Güvenirlik Katmanından (GK) (Dependability Layer-DL) oluşmaktadır. Bu mimarinin bir parçası olmayan ve gerçek zamanlı olmayan uygulama olarak kabul edilen bir eşzamanlama protokolu ile zaman dilimli çalışma elde edilmektedir. Mimarinin yapısı Şekil 1'de görülmektedir. Şekilde DL: Güvenirlik Katmanı (Dependability Layer), RT_i: Gerçek zamanlı uygulama *i*, nRT: Gerçek zamanlı olmayan

uygulamalar kısaltmaları kullanılmıştır.



Şekil 1: D^{3}RIPMimarisi.

Projemizin çıktısı olan formal işçerçevesi bu katmanların ve katman arayüzlerinin Timed IO Automata (TIOA) yöntemi ile ayrık olarak modellenmesi ve bu yönetmin sağladığı çeşitli özellikler ve operasyonlar kullanılarak bütün katman yığıtı modelinin elde edilmesine dayalıdır. Analitik olarak oluşturulan TIOA modelleri daha sonra UPPAAL doğrulama yazılım aracıyla da benzetim yoluyla doğrulanmıştır.

Endüstriyel ağlar üzerlerinde çalışan kontrol uygulamaları insan hayatına doğrudan etki edebilecek özelliklerdedirler. Bu nedenle endüstriyel ağların her koşulda tanımlandıkları şekilde çalışmaları ve gerçek zamanlı garantileri sağlamaları gerekmektedir. Bu nedenle bu ağlar için protokolların formal olarak geliştirilebilmesi, incelenebilmesi ve çalışmalarının doğrulanabilmesi önemlidir. Literatürdeki endüstriyel ağ protokolları sistematik bir yaklaşım olmadan uygulama ihtiyaçları doğrultusunda ev ofis ağları için olan yaklaşımlara ekler ve değişiklikler yaparak, kontrol uygulamalarının davranışı hakkındaki bilgilerden faydalanmadan geliştirilmişlerdir. Pek çok var olan protokol için güvenirlik desteği bulunmamaktadır.

D^{3}RIP iş çerçevesi ve protokol yığıtının literatüre ve endüstriyel haberleşme protokolları teknolojisine yaptığı katkılar aşağıda sıralanmıştır:

1) ENDÜSTRİYEL UYGULAMALARIN TRAFİK ÖZELLİKLERİNE GÖRE TASARIM: Endüstriyel kontrol sistemlerinde gönderilen mesajlar ya periyodiktir ya da mesajların gönderileceği sistem modeline göre önceden bilinebilir. İlk defa endüstriyel kontrol sistemleri için haberleşme ağlarında taşınan trafiğin özelliklerini gözönüne alan bir protokol tasarımı yapılmıştır. Endüstriyel kontrol sistemi uygulamalarının sert gerçek zamanlı ihtiyaçlarını karşılamak için literatürdeki yaklaşımlarda en kötü duruma göre varsayım yapılmakta (bütün mesajların aynı anda gönderlmesi gibi) ve buna uygun kapasite tahsisi yapılmaktadır. Bu en kötü durumun gerçekleşmesinin mümkün olup olmadığı incelenmemektedir. Bu sebeple kapasite tahsisi verimsiz olarak yapılmaktadır. D^{3}RIP gerçek zamanlı trafik için usta yamak yaklaşımı kullanmadan tamamen dağıtılmış olarak dinamik kaynak tahsisi ile kaynakları daha verimli kullanan ilk mimaridir. Endüstriyel kontrol sistemleri ağları eskiden beri süregelen fieldbus veriyollarına uygun tasarlanmış ve kablolanmıştır. Bu sebeple anahtarlı yapılar yerine paylaşımlı ortam üzerinde gerçek zamanlı garantiler sağlamak önemli bir kazanımdır. D^{3}RIP bu beklentiyi karşılamaktadır.

2) FORMAL MODELLEME VE İŞÇERÇEVESİ: İlk defa endüstriyel haberleşme protokollarının tasarımı için genel, sistematik formal bir yaklaşım geliştirilmıştir [SCHMIDT (2012)].

3) İŞÇERÇEVESİ DOĞRULAMASI: TIOA modellerinin ispat yoluyla analitik doğrulamasının yanı sıra, proje önerisinde yer almayan ek bir çalışmayla, bu modellerin UPPAAL doğrulama yazılım aracıyla da doğrulaması yapılmıştır. UPPAAL genellikle kontrol ve haberleşme protokolü uygulamaları gibi zaman kritik uygulamaların analiz ve doğrulanmasında kullanılan gerçek zamanlı bir modelleme aracıdır [LARSEN et al. (1994)]. Yine proje önerisinde yer almayan ek bir çalışmayla ilk defa proje kapsamında TIOA modellerini UPPAAL modellerine otomatik olarak çeviren bir algoritma geliştirilmıştir [KARTAL et al. (2013)].

4) GÜVENİRLİK: Güvenirlik desteği için ev ofis uygulamalarında kullanılan mesajlara sıra numarası, zaman bilgisi ekleme ve alındı mesajları gönderme gibi yöntemler kullanılmakta endüstriyel haberleşme ağlarına özel bir yaklaşım izlenmemektedir. D^3RIP, kontrol uygulamalarındaki determinizm ve modellenebilirlikten faydalanan ilk güvenirlik desteği yaklaşımını içermektedir [KARTAL et al. (2013a)].

5) STANDART BİLEŞENLER, AÇIK KAYNAKLI İŞLETİM SİSTEMİYLE YÜKSEK łınebreak BAŞARIMLI TAŞINABİLİR gerçekleşTİRİM: Projede geliştirilen işçerçevesine uygun olarak standart (Components Off The Shelf-COTS) ağ donanımları ya da anahtarlar olmadan, zahmetli ve uygulamaya özelleşmiş kurulum gerektirmeyen, aynı zamanda gerçek zamanlı trafiği garantili ileten bir protokol gerçekleştirimi yapılmıştır [KAYA et al. (2013)]. Bu gerçekleştirimde tamamen açık kaynak yazılımlar ve gerçek zamanlı iletişim sistemi (Linux-Lubuntu) kullanılmış, gerçek zamanlı başarımı artırmak üzere pek çok işletimn sistemi ayarları yapılmıştır. Sonunda elde edilen protokol yığıtı gerçekleştirimi tamamen taşınabilir ve bir USB hafıza çubuğundan boot edilerek çalıştırılabilir olmuştur. Herhangi bir özel donanıma ya da sürücüye ihtiyaç duymamaktadır. Bu gerçekleştirim hem bilgisayar hem de PLC (Programmable Logic Controller-Programlanabilir Mantrık Kontrolcusu) cihazları üzerinde test edilmiştir.

Raporun geri kalanının organizasyonu şu şekildedir:

Bölüm 2, projenin konusu olan gerçek zamanlı endüstriyel Ethernet protokollarının gereksinimlerini tanımlamakta ve literatürdeki protokolların karşılaştırmalı olarak tartışmasını yapmaktadır. Bunu takiben kullanılan formal modelleme yöntemi TIOA ve doğrulama aracı UPPAAL hakkında ön bilgiler sunulmaktadır. Literatürdeki güvenirlik yaklaşımları da bu bölümde özetlenmektedir.

Gereç ve Yöntem (Bölüm 3) altında öncelikle proje yönetimi, ortak çalışmalar ve işbirlikleri ile birlikte proje çıktıları Bölüm 3.1'de anlatılmaktadır. Proje çıktıları olan yayınlanmış, hakem değerlendirmesinde olan ve sunulmak üzere hazırlanmış makaleler, bildiriler ve tamamlanmış tezler TÜBİTAK proje takip sistemine rapor dönemi çıktıları olarak ayrıca yüklenmiştir. Buna ek olarak projede geliştirilen simülatör yazılımı kaynak kodları, UPPAAL yazılımında kullanılan modeller, protokol gerçekleştirimi son sürümü, protokol çalışmasını ve laboratuvar ortamını gösteren demo videosu ve ilgili diğer materyale ulaşılabilecek link aşağıdadır:

http://www.eee.metu.edu.tr/~eguran/D3RIP.html

Bölüm 3.2'de D^2RIP İş Çerçevesi TIOA Modeli ve Bölüm 3.3'da bu modele uygun olarak yapılan ilk sürüm gerçekleştirme ve proje çıktıları arasında yer alan yığıt simülatör yazılımı

özetlenmektedir. Bu işler gelişme raporlarında detaylı olarak anlatılmıştır. Son altı aylık dönemde daha önce yapılan D^2RIP gerçekleştirimi gerçek zaman tepki zamanlarını hızlandırmak amacıyla optimize edilmiştir. Aynı zamanda bu son altı aylık dönemde olan bir teknolojik gelişmeyle IEEE 1588 donanım zaman damgalı ucuz standart ağ arayüz kartları ve bu kartlar için sürücü yazılımları piyasaya çıkmıştır. Bu imkan değerlendirilmiş ve eşzamanlama hassasiyeti artırılmıştır. Buna ek olarak işletim sisteminde de çok alt düzeyde ayarlar yapılarak çalışma hızlandırılmıştır. Bu iyileştirmeyle D^2RIP gerçekleştiriminin son sürümü Bölüm 3.4'de detaylı olarak anlatılmaktadır. Bu yeni sürümün daha önce gerçekleştirilen ilk sürüme göre farklılıkları ve üstünlükleri Ekler bölümünde tablolanmıştır. D^2RIP başarımının nicel olarak ölçülmesi için başarım ölçütleri Bölüm 3.5'de tanımlanmaktadır. Yapılan deneylerde kullanılan örnek uygulama Bölüm 3.6'de açıklanmaktadır.

Bölüm 3.7 proje önerisinde önceden öngörülmemiş bir çıktı olan TIOA Modellerinin Algoritmik olarak UPPAAL Modellerine Dönüştürülmesi algoritmasını anlatmaktadır. TIOA işçerçevesine güvenirlik katmanı eklenmesi Bölüm 3.8'da ve bu katmanın gerçeklenmesi Bölüm 3.11'de anlatılmaktadır. Bu çalışmaların sonucunda D^2RIP TIOA modeli revize edilmiştir, bu yeni model Bölüm 3.9'da anlatılmaktadır.

Güvenirlik katmanı ile birlikte bütün D^2RIP mimarisinin UPPAAL aracı ile doğrulanması Bölüm 3.10'de ve bu mimariye uygun bir Güvenirlik Katmanı gerçeklemesinin yapılmasının ardından yığıt yapısına entegre edilmesi Bölüm 3.11'de anlatılmaktadır.

Bulgular (Bölüm 4) altında D^2RIP/D^3RIP yığıtlarının deneysel başarım incelemesi ve sonuçları anlatılmaktadır. Öncelikle Bölüm 3.6'de açıklanan örnek uygulamaya dayalı deney ortamı Bölüm 4.1'de tasvir edilmektedir. İyileştirilmış D^2RIP gerçekleştiriminin zamanlama başarımı Bölüm 4.2'de, D^2RIP deneysel inceleme sonuçları Bölüm 4.3'de anlatılmaktadır. D^3RIP formal modellemesinin ve UPPAAL doğrulamasının sonuçları Bölüm 4.4'de sunulmaktadır. Bölüm 4.4'de sunulan D^3RIP deneysel incelemesi ile Bulgular tamamlanmaktadır. Son olarak Simülatör yazılımının protokolun gerçeklenmesi ile ne kadar yakın sonuçlar verdiği Bölüm 4.5'de incelenmektedir.

Proje sonuçları Bölüm 5'de özetlenmektedir. Bu bölümde proje planındaki iş paketleri yapılırken karşılaşılan zorluklar da tartışılmaktadır. Son olarak bu bölümde projenin tamamlanmasını takiben konuyla ilgili yapılacak devam niteliğinde işler tartışılmaktadır.

Ekler Bölümünün içeriği aşağıda sıralanmıştır:

- Erlangen Üniversitesine yapılan ziyaret hakkında oradaki araştırma ekibi lideri Prof. Dr. Moor'dan gelen mektup
- Geliştirilmış D^2RIP sürümünün ilk sürüme göre farklarının tablolanması
- Hakem değerlendirmesinde makale: Dynamic Distributed Realtime Industrial Communication Protocol (D^2RIP): Architecture, Implementation and Experimental Evaluation, *Elsevier Computer Standards and Interfaces* Dergisine sunuldu, hakem değerlendirmesi altında.
- Hakem değerlendirmesinde makale: Modeling and Formal Verification of TIOA Based Distributed Real-Time Systems, *IEEE Transactions on Software Engıneering* Dergisine sunuldu, hakem değerlendirmesi altında.

- Sunulmaya hazır makale: Dependability Design for a Distributed Real-time Protocol Family, *IEEE Transactions on Parallel and Distributed* Systems Dergisine sunulacaktır.

Bu eklerdeki dökümanlar TÜBİTAK Proje Takip Sistemine bu sonuç raporu ile birlikte ayrı bir döküman olarak yüklenmiştir.

TÜBİTAK ARDEB'e gönderilen 2 DVD'nin içerikleri aşağıda sıralanmıştır:

- Proje raporu ve ekleri (DVD1)
- Protokol çalışmasını ve laboratuvar ortamını tanıtan video (DVD1)
- Protokol Simülatörü kaynak kodları (DVD1)
- UPPAAL dosyaları (DVD1)
- Güvenirlik katmanı kaynak kodları (DVD1)
- USB hafıza çubuğundan çalıştırılabilecek tüm D^2RIP, ayarları yapılmış lubuntu işletim sistemi ve gerekli Ethernet sürücülerinin imaj dosyası ve bu dosyayı USB hafıza çubuğuna yükleyecek program (DVD2)

Bu içeriğin bir bölümü yukarıda belirtilen web sayfasında da bulunabilir.

2 Genel Bilgiler

2.1 Gerçek Zamanlı Endüstriyel Ethernet Protokolları: Gereksinimler ve Literatür araştırması

Ağla haberleşen endüstriyel kontrol sistemleri (CIM-Computer Integrated Manufacturıng); kontrol edilecek olan sistem ve işlevleri bir haberleşme ağı üzerinden koordine edilen algılayıcılar (sensors), kontrolcular (programlanabilir mantık kontrolcuları-Programmable Logic Controllers -PLCs), endüstriyel kişisel bilgisayarlar -Industrial PCs -IPCs) ve eyleyicilerden (actuators) oluşur [BaAn07]. Endüstriyel kontrol uygulamalarının giderek karmaşık ve büyük ölçekli hale gelmeleri ve endüstriyel kontrol cihazlarının bilgisayar ve ağ erişimi destekli olarak üretilmeleri ağla haberleşen endüstriyel kontrol sistemlerini önemli bir endüstriyel ve akademik araştırma konusu haline getirmiştir. Bu sistemler için son yirmi yıl içinde farklı endüstriyel haberleşme ağları geliştirilmıştir [BAILLIEUL (2007), MOYNE (2007), DECOTIGNIE (2005), DECOTIGNIE (2009)].

Endüstriyel haberleşme ağlarında farklı amaçlarla ve farklı özelliklerde mesajlar taşınması gerekmektedir. Bunlar aşağıdaki gibi gruplanabilir [MOYNE (2007)] (Şekil 2):

T1) Algılayıcılar (sensors) -kontrolcular - eyleyiciler (actuators) arasında cihaz düzeyinde veri alışverişi: Sürekli ya da örneklenmiş veriler çoğunlukla periyodik olarak ve zaman kısıtları ile gönderilir. Örnek: servo sürücüsündeki hız algılayıcısından toplanan veri kontrolcuya gönderilir, kontrolcudan eyleyiciye çıkış akımı değeri gönderilir.

T2) Gözetim kontrolu (supervisory control) düzeyindeki mesajlar: Kontrolcuların ve kontrol edilen sistemlerin hiyerarşik organizasyonlarına uygun olarak hiyerarşide farklı düzeydeki sistem bileşenlerinin arasında koordinasyon amacı ile haberleşme gerekir. Çoğunlukla olay tabanlı ve deterministik tepki zamanları gerektiren veriler gönderilir. Sistemin davranışı ayrık zamanlarda (discrete time) değiştiği için sistemin bir sonraki durumu ve bu durumda gönderilecek mesajlar sistemin içinde olduğu durum ve sistemin dinamik modeli kullanılarak önceden bilinebilir. Örnek: İki makineyi kontrol eden bir kontrolcu sistem modeline uygun olarak birinci makinenin işleminin bitmesi olayı gerçekleştiğinde ikinci makineye çalışmaya başlaması için mesaj gönderir. Bu mesajın gönderileceği sistem modelinde birinci makinenin işleminin bitmesi olayı ile birlikte belirtilmiştir.

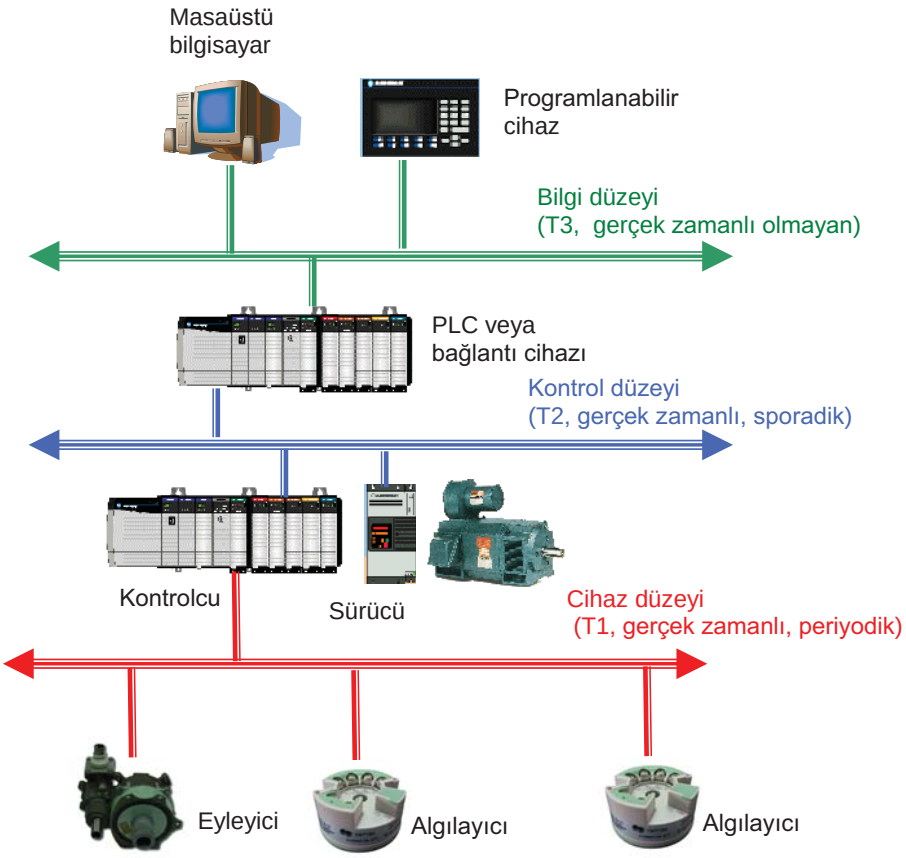
T3) Tanılayıcı (diagnostik) veriler ve uzaktan idare amaçlı uygulamalar: Çoğunlukla gerçek zamanlı olmayan ve olay tabanlı haberleşme olarak taşınır.

Bu trafik tipleri incelendiğinde endüstriyel haberleşme sistemlerinde dört önemli gereksinim belirlenebilir: 1) Gerçek zamanlı trafik iletimi: Mesajların yaratıldıktan sonra belirli bir son gönderilme zamanından (deadline) önce iletilmesi (T1, T2 trafik tipleri).

2) Eşzamanlı haberleşme: Gerçek zamanlı trafik iletimini sağlayabilmek için ağdaki düğümlerin ortak bir zaman tabanı olması (T1, T2 trafik tipleri).

3) Güvenirlik (dependibility): Endüstriyel kontrol uygulamalarında hata (fault) ve bozulma (failure) durumlarına karşı güvenirlik desteği (T1, T2 trafik tipleri).

4) Gerçek zamanlı olmayan trafik desteği: Bu mesajlar ağ üzerinden gerçek zamanlı mesajların başarımını düşürmeden iletilebilmesi (T3 trafik tipi).



Şekil 2: Endüstriyel haberleşme ağlarında farklı düzeyde iletişim.

Endüstriyel haberleşme ağlarında gerçek zamanlı garantileri sağlanması en kötü durumda mesajların gecikmeden gönderilmesini gerektirmektedir. Benzer şekilde güvenirlik desteği de istenen hata olasılığını sağlamak için en kötü durumlara göre olmalıdır. Bu nedenlerle endüstriyel ağlarda toplam kapasite ihtiyacı ve kapasite tahsisi bütün mesajların aynı anda gönderilmesi gibi çoğu zaman gerçekçi olmayan varsayımlara göre hesaplanır.

Endüstriyel ortamlarda kullanılmak için ilk haberleşme ağları olan CAN, LonWorks ve Profibus gibi alan veriyolları (field-bus) yaklaşık yirmi yıl önce kullanılmaya başlamışlardır [THOMESSE (2005)]. Bu veriyolları şirkete özel (proprietary), pahalı, geliştirmesi zor, birbirleri ile uyumlu olmayan ve değişen endüstriyel uygulamalara hızla uyum sağlayamayan teknolojilerdir. Buna karşılık basit, yüksek hızlı ve ev, ofis ortamlarındaki çok yaygın kullanımı nedeni ile ucuz olması Ethernet'i endüstriyel haberleşme için önemli bir aday yapmaktadır [DECOTIGNIE (2005), DECOTIGNIE (2009), MOYNE (2007), FELSER (2009)]. Bu önemli üstünlüklerine karşın standard Ethernet (Ethernet-IEEE 802.3) üzerinde aynı anda ağa gönderilen mesajlar çarpışmakta (collision) ve bundan sonra rasgele zamanlarda tekrar gönderilmektedir. Dolayısı ile standard Ethernet ağ erişimini deterministik olarak sağlayamamakta ve gerçek zamanlı güvenilir haberleşmeyi destekleyememektedir. Son yıllarda gerçek zamanlı Ethernet (Real-time Ethernet (RTE)) geliştirimi için akademik ve endüstriyel çevrelerde çalışmalar sürmektedir [DECOTIGNIE (2005), DECOTIGNIE (2009), FELSER (2009)].

2.1.1 Gereksinimler

[FELSER (2009)]'den derlenen gerçek zamanlı Ethernet için en önemli gereksinimler aşağıda sıralanmıştır.

Gerçek zamanlı veri iletimi
Mesaj iletim zamanı gönderen ve alan uygulamalar arasında ölçülmektedir. Yukarıda bahsedilen farklı düzeyde haberleşmeler için mesaj iletim zamanı gerekleri farklılık göstermektedir. İnsan operatörlerin dahil olduğu uygulamalar (süreç otomasyonu, bina denetimi) 100 ms civarında iletim zamanlarına ihtiyaç duyarken, PLCler ile çalışan süreç denetimi uygulamaları için 10 ms altı iletim zamanı ve pek çok cihazı koordine eden hareket denetimi uygulamaları için ise 1 ms altı gecikme zamanı gereklidir.

Eşzamanlama desteği
Endüstriyel ağlarda gerçek zamanlı tepki zamanı ağ düğümleri arasında ortak referans zaman için eşzamanlama protokoluna dayalıdır. Eşzamanlama hassasiyeti iki cihazın saatleri arasındaki en fazla sapma olarak tanımlanır [FELSER (2009)]. Bu hassasiyet mesajların önce ve sonrasında gerekli olabilecek koruma zamanlarının ve dolayısı ile gecikmelerin artmasına sebep olur. En geniş kabul gören eşzamanlama protokolu Ethernet üzerinde sorunsuz olarak dağıtılmış koşan IEEE 1588 [IEEE (2002), EIDSON (2006)] protokoludur.

IEEE 1588 Precision-time protocol (PTP) göre çalışır. Seçilmiş bir yönetici düğümle diğer düğümler arasında mesaj alışverişleriyle yönetilen düğümlerin yönetici düğümle aralarındaki saat farkını ve gecikme süresini hesaplayıp saaatlerini buna göre ayarlamaları sağlanır. IEEE 1588 dışında

EtherCAT [JANSEN (2004)] ve Sercos (IEC 61491) [SCHEMM (2004)] gibi protokolların kendi özel eşzamanlama mekanizmaları bulunmaktadır.

Gerçek zamanlı olmayan trafik desteği
Denetim süreçlerinin uzaktan yönetimi gibi uygulamalar için yüksek düzey yönetim ve tanılama verisinin standart HTTP ya da FTP trafiği ile TCP/UDP/IP üzerinden iletilmesi istenmektedir. Bu tür gerçek zamanlı olmayan (nRT) trafiğin gerçek zamanlı (RT) kontrol uygulaması trafiğine hiç etki etmeden taşınması gereklidir.

Uyumluluk
Etherneti çekici bir teknoloji kılan en önemli sebepler arasında ucuz donanımı ve yazılım arayüzü gelmektedir. Bunlardan yararlanabilmek ve COTS (Commercial Off-The-Shelf) bileşenlerle gerçekleştirim yapabilmek için endüstriyel Ethernet'in standart Ethernet ile uyumlu çalışması gereklidir. Buna ek olarak çok kullanılan HTTP ve FTP gibi uygulama protokollarının yanısıra IEEE 1588 gibi standart eşzamanlama protokollarının da desteklenebilmesi gerekmektedir.Geriye doğru uyumluluk gerekleri vardır. Bu nedenle bir protokolun bir kere kurulduktan sonra yıllarca çalışması beklenmektedir.Sonuç olarak bir endüstriyel Ethernet protokolunun yeni cihazların eklenmesine elverişli olması gereklidir.

Gerçek zamanlı trafik için dinamik kaynak ayırılması
Bir ağla haberleşen edüstriyel sistemin haberleşme gerekleri zamanla dinamik olarak değişebilir [SCHMIDT (2012)]. Örnek olarak öz-tetiklenen kontrol [NESIC et al. (2009)] kavramında cihaz seviyesinde haberleşm için önceden hesağlama zamanları ayrılmıştır. Buna ek olarak güncel dağıtık sistemlerde koordinasyon görevi gören üst seviye kontrolcular sadece gerekli olduğunda haberleşmektedirler. Buna göre endüstriyel Ethernet protokllarında RT bant genişliğinşn anlık

ihtiyaçlara göre cihazlara ayırılabilmesi gerekmektedir.

Güvenirlik
Endüstriyel kontrol sistemlerinde çalışan kritik güvenlik kısıtları olan uygulamalar için güvenirlik (dependability) önemli bir gereksinimdir [DEP (2007)]. Güvenirlik kavramının içinde kullanırlılık (availability), güvenlik (safety), bütünlük (integrity) ve kolay bakımlı olma (maintainability) gibi unsurlar da bulunmaktadır [AVIZIENIS et al. (2004)]. Ağla haberleşen dağıtılmış bir endüstriyel kontrol sisteminin güvenirliğinden bahsedebilmek için hem ağın hem kontrolcuların güvenirliğinin sağlanması gereklidir. Güvenilir bir endüstriyel haberleşme ağı tasarlarken güvenilir dağıtılmış eşzamanlama, mesajlarda gönderilen değerlerin birbirleri ve sistem durumu ile tutarlı olması önemlidir. Güvenirlik problemi Ethernet'in deterministik olmayan özelliklerinden dolayı RTE tabanlı çözümlerde daha öne çıkmaktadır [FELSER (2004)]. Güvenilir haberleşmenin doğru bilginin doğru yere doğru zamanda ve sırada gönderilmesini sağlaması gerekmektedir. Güvenirlik desteği çoğu zaman varsayılan en kötü duruma göre statik ek kapasite ayırımı ile yapılmaktadır [DECOTIGNIE (2009)]. Örnek olarak TDMA tabanlı bir protokol için kaybolan her mesajın tekrar zamanında gönderilebilmesi için düğümlere ayırılmış zaman dilimleri kadar ek zaman dilimi ayırılması gerekmekte ve kapasitenin ancak yarısı yararlı olarak kullanılabilmektedir.

2.1.2 Gerçek zamanlı Ethernet protokolları

Aşağıdaki bölümlerde Ethernet üzerinden gerçek zamanlı haberleşme garantileri sağlamak için literatürde yapılmış çalışmaların karşılaştırmalı kritik bir incelemesi bulunmaktadır. Literatürde Ethernet'e gerçek zamanlılık ekleyebilmek için belli başlı dört yaklaşım bulunmaktadır. Bunlar; Ethernet ağ arayüz kartı donanımının uyarlanması ile ortama erişim katmanının dolayısı ile çarpışmadan sonraki deterministik olmayan mesaj gönderme işlevinin değiştirilmesi, çarpışma olasılığının ve tepki zamanlarının azaltılması, noktadan noktaya bağlantılar ile anahtarlar kullanılarak paylaşımlı ortamdaki çarpışma olasılığının kaldırılması ve Ethernet ortama erişim katmanının üzerine deterministik ortama erişim sağlayan ek bir katman eklenerek çarpışmadan kaçınılması (collision avoidence)'dır.

Özelleşmiş donanım:
Ethercat [JANSEN (2004)], SERCOS III [SCHEMM (2004)] ve ProfiNet [PRO (2004)] özelleşmiş standart olmayan düğüm ve anahtar donanımı için tasarlanmışlardır. Ethercat ve Profinet'de eşzamanlı veri transferi IEEE 1588 ile sağlamakta, SERCOS III özel mesajlar kullanmaktadır. Bu üç protokol özel geliştirilmiş güvenlik protokolları ile desteklenmektedirler. Özel bir protokol olan TwinSafe protokolu EtherCat protokolunun alt katmanlarından bağımsız ayrı bir katman olarak çalışmaktadır. Cihazlar adreslenmekte ve veri güvenliği CRC ile sağlanmaktadır. SERCOS III Safety'de mesajlarda sıra numarası ve zaman bilgisi bulunmaktadır. Mesaj alıcısı cihazlar gönderen cihazlara alındı bilgisi göndermektedirler. Cihazlar adreslenmekte ve veri güvenliği HDLC kodlama ile sağlanmaktadır. PROFIsafe ProfiNet [PRO (2004)] için geliştirilmıştır. Mesajlara sıra numarası ve zaman bilgisi eklenmiştir. Cihazlar adreslenmekte ve veri güvenliği CRC ile sağlanmaktadır.

Garanti sağlamayan yaklaşımlar:
MODBUS/TCP [MOD (2002)] benzeri protokollar uyumlu olabilmek için standard Ethernet üzerinde TCP/IP yığıt limitleri ile çalışmaktadırlar. [KWEON (2003), KWEON et al. (2004)] trafik şekillendirme ile düşük gecikmeler amaçlamaktadır. Bu tür yaklaşımlarda mesajların

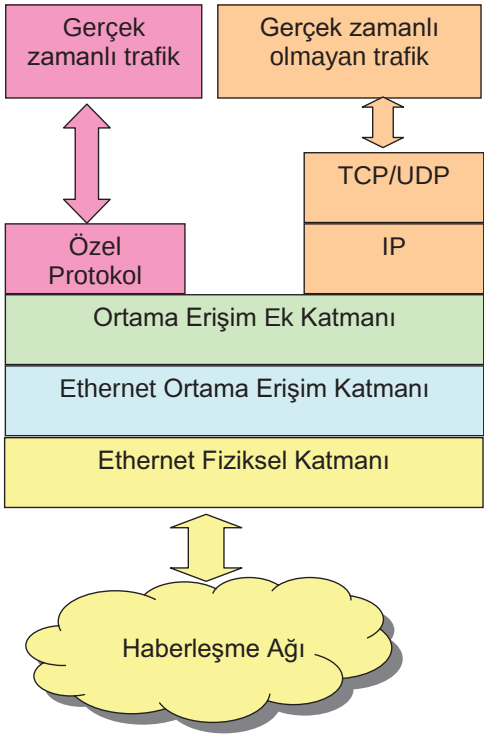
zamanında iletileceğine dair bir garanti yoktur [DECOTIGNIE (2009)].

Anahtarlanan Ethernet (IEEE 802.3x):
Standart paylaşımlı Ethernet'deki çarpışmadan kaynaklanan deterministik olmayan ağ erişimi problemine getirilen bir çözüm tam çift yönlü, anahtarlanan, noktadan noktaya Ethernet (IEEE 802.3x) olmuştur. Bu yapı ile paylaşımlı ağ erişimi dolayısı ile çarpışma problemi ortadan kalksa da ağa erişim problemini ağ içindeki kuyruklara taşınmıştır [DECOTIGNIE (2005), DECOTIGNIE (2009), FELSER (2005)]. Gerçek zamanlı haberleşme sağlamak için kuyruk çizelgeleme (scheduling) ve önceliklendirme (prioritization) yapabilen anahtarlar gerekmektedir [LIU (2007), SCHARBARG (2007)]. Paketlere öncelik tanınmasına ve bu önceliklere göre farklı servis verilmesini sağlayan 802.1p, 802.1Q gibi Ethernet protokolları ve protokol eklentileri önerilmiştir. Bu protokollar standart Ethernet'ten farklı olarak özelleşmiş anahtarlar gerektirmektedirler. Sonsuz arabellek varsayımı altında gerçek zamanlı trafik için çizelgelenebilirlik analizi yapılabilmekte olsa da gerçek koşullar sınırlı arabellek kullanımını gerektirmektedir. Sınırlı arabellek için yapılan analizler ise sadece olasılık ile belirlenen garantiler için olmaktadır [WANG et al. (2002)Wang, Song, Chen, , Sun]. Anahtarlanan Ethernet üzerinde gerçek zamanlı çalışma için önemli olan eşzamanlama işlevininin hassas olarak gerçekleştirilmesi de zorlaşmaktadır.

Ethernet/Ip (EIP) [EIP (2004)] TCP/IP üzerinde ve tam çift yönlü Ethernet üzerinde özel önceliklendirme mekanizması içeren anahtarlar ile çalışır. Ethernet/Ip protokolunun gerçek zaman garantileri bulunmamaktadır. Eşzamanlı veri transferi IEEE 1588 protokolu ile uyumlu özel mesajlar göndererek yapılmaktadır. Ayrıca mesaj göndericiler ve alıcılar arasındaki koordinasyon karşılıklı ping mesajları ile sağlanmaktadır. Mesajlarda zaman bilgisi bulunmaktadır. Cihazlar adreslenmekte ve veri güvenliği CRC ile sağlanmaktadır.

Endüstriyel ağların gelişimi veriyolu yapısı olan fieldbus teknolojisi ile başlamıştır. Bu nedenle pek çok sistem veriyolu topolojisi şeklinde kablolanmıştır. Bu sistemleri anahtarlanan Ethernet ile bağlamak cihazların kapsamlı şekilde yeniden kablolanmasını gerektirmektedir [FELSER (2009), MORAES et al. (2011)].

Standart Ethernet (IEEE 802.3) ek ortama erişim katmanı:
Paylaşımlı Ethernet'e çarpışmalardan kaçınmaya (collision avoidance) dayalı gerçek zamanlılık özellikleri ekleme amaçlı akademik ve endüstri kaynaklı bir çok protokol önerilmiştir. Bu protokollar çoğu IEEE 802.3 katmanının üzerine yeni bir ortama erişim katmanı ekleyerek çarpışmadan ve sonraki rasgele mesaj göndermeden kaçınmayı amaçlamaktadır. Gerçek zamanlı olan ve olmayan bütün trafik bu yeni katmandan geçmektedir. Bu katmanın üzerinde gerçek zamanlı trafiğin iletilmesinden sorumlu özel bir protokol olabildiği gibi bu işlevi TCP-UDP/IP katmanları da görebilir (Şekil 3).



Şekil 3: Ek ortama erişim katmanı.

Yeni ortama erişim katmanı yaklaşımları arasında zaman bölümlü çoklu erişim (Time Division Multiple Access -TDMA), usta-yamak (master slave) ve andaç geçirme (token passing) sayılabilir [DECOTIGNIE (2005), DECOTIGNIE (2009)]. Aşağıda bu yaklaşımlar incelenmektedir.

TDMA (Time Division Multiple Access): Zaman çoğu zaman eş uzunlukta dilimlere bölünmüştür. Her düğüme mesajını göndereceği bir ya da birden fazla zaman dilimi statik olarak ayırılır. Erişimin doğru şekilde sağlanabilmesi için düğümler arasında eşzamanlılık sağlanması gerekmektedir. Bu yöntem düğümlere güvenilir ağ erişimi sağlar. TDMA erişiminin en olumsuz yanı düşük verimli çalışmasıdır. Düğümler kendilerine ayırılan dilimlerde mesaj göndermezlerse kullanılmayan dilim başka bir düğüm tarafından kullanılamamaktadır. Buna ek olarak protokol yazılımında ve varsa anahtarlarda olacak gecikmelerin dilim uzunluğunu seçerken hesaba katılması gerekmektedir. Eğer mesajlar ağdaki hatalar nedeni ile kaybolurlarsa mesajları tekrar gönderebilmek için ek zaman dilimlerinin ayırılması gerekmektedir.

Usta-yamak (master-slave): Seçilen özel bir düğüm (usta-master) diğer düğümlere (yamak-slave) mesaj gönderip göndermeyeceklerini sıra ile sorar (polling-yoklama). Yamak düğümler sadece usta tarafından yoklandıklarında mesaj gönderebilirler. Bu teknik özellikle trafiğin düzenli olduğu, düşük sayıda düğümün olduğu ağlarda kullanılabilmektedir. Usta-yamak erişiminin verimi yoklama işleminin kaybettirdiği zamandan olumsuz olarak etkilenmektedir. Özellikle trafiğin çok değişken olduğu durumlarda düğümlerin yoklandıklarında gönderecek mesajlarının olmadığı durumlarda verim düşmektedir. Usta düğümün yamak düğümlerden cevap beklemesi ile kaybedilen zamanlar da verimi ayrıca düşürmektedir. Düğüm sayısı fazla olduğunda her düğümü yoklamak için kullanılacak zamanların toplamı (bir yoklama çevrimi) en az gecikme ile gönderilmesi gereken mesajın gecikmesinden fazla olabilir. Bu durumda mesaj-

lar kabul edilemeyecek kadar gecikebilirler. Yamak düğümlerdeki yazılımın hızı da yoklama işleminin hızını belirleyici etkenlerden biridir. Yazılım çok yavaş olursa ağ hızı önemini kaybetmekte ve ağın verimi bu yavaş yazılım nedeni ile azalmaktadır. Verim probleminin yanı sıra, usta-yamak haberleşmesi dağıtılmaya uygun bir yapı değildir ve usta düğümden kaynaklanan bir tek merkezli, tek noktadan hataya açık bir özellik taşımaktadır. Bu problemler nedeni ile usta-yamak yapıların hataya duyarlılığı fazladır. Usta-yamak düzeni bir usta bileşenin diğer bileşenlerden fazla bilgiye ve sorumluluğa sahip olmasını gerektirmekte ve dağıtılmış sistemlere aykırı bir yapı olmaktadır. Özelleşmiş donanımların ve öncelikli kuyruk desteği sağlayan anahtarların maliyetleri yüksektir.

Andaç geçirme (token passing): Bir düğüm ancak mesaj gönderme hakkını temsil eden andaç elindeyken mesaj gönderebilir. Bu hakkı kullandığında özel bir mesaj kullanarak andaçı bir sonraki düğüme iletir. Andaç tabanlı (token-based) yaklaşımlar andaç kaybolması, andaç dolaşım zamanının haberleşmeyi yavaşlatması, sisteme yeni bir bileşen ekleme zorlukları gibi dezavantajlara sahiptir.

RTE için endüstriyel çevrelerce paylaşımlı Ethernet'deki çarpışmalardan kaçınmaya (collision avoidance) dayalı bir grup standart oluşturulmuştur. Buna paralel olarak akademik çevrelerce de çözümler üretilmiştir. Bu standartlar ve çözümler yukarıda açıklanan yaklaşımları izlemekte ve tartışılan olumsuz yönlerini taşımaktadırlar.

Time Critical Control Network (TCNet) [TCN (2004)] standart Ethernet katmanı üzerinde andaç geçirme (token passing)'ye dayalı ek bir katman ile gerçekleştirilmiştir. Gerçek zamanlı olmayan trafiği düşük öncelikle taşıyabilmektedir. Eş zamanlılık andaçla taşınan özel mesajlarla yapılmaktadır. Güvenirlik her TCNet kartı için fazladan bir kart daha kullanarak sağlanmaktadır.

Powerlink (EPL) [EPL (2004)] usta-yamak (master-slave) biçiminde ortama erişim sağlayan ek katman ile standard Ethernet üzerinde gerçekleştirlmiştir. Usta-yamak yapısının verimsizliğinden dolayı EPL verimi %25 olarak hesaplanmıştır [DECOTIGNIE (2009)]. EPL protokolunda eşzamanlı veri transferi IEEE 1588 protokolu benzeri özel mesajlar göndererek yapılmaktadır. Gerçek zamanlı olan ve olmayan veriler ayrı gönderme pencerelerinde gönderilmektedir. Mesajlarda sıra numarası ve zaman bilgisi bulunmaktadır. Cihazlar adreslenmekte ve veri güvenliği CRC ile sağlanmaktadır.

Ethernet for Plant Automation (EPA) [EPA (2004)] statik TDMA kullanarak çalışır. TDMA çevirimi önceden gerçek zamanlı olan ve olmayan trafik için bölünmüştür.

Akademik alanda da çarpışmadan kaçınmaya dayalı çalışmalar yapılmaktadır. FTT-Ethernet [PEDREIRAS et al. (2005)] hem paylaşımlı hem de anahtarlanan Ethernet standart donanımı üzerine özel bir arayüz katmanı ekleyerek ve usta-yamak haberleşmesi kullanan bir yaklaşımdır. Sanal Andaç Geçirme (Virtual Token Passing -VTPE) Protocol [MORAES et al. (2011)] Ethernet binary exponential back off (BEB) algoritmasını kapatarak ve çarpışma anında gerçek zamanlı trafik üreten düğümlerin hemen tekrar göndermesini öneren bir akademik çalışmadır. Gerçek zamanlı düğümler arasında olabilecek çekişmeler andaç geçirmeyle çözülmektedir.

Bu çalışmamızın odak noktası Ethernet çalışma prensibini değiştirmeden uyumlu bileşenlerle çalışan ve garantili gerçek zamanlı başarım sağlayan paylaşımlı ortam protokollarıdır. Bu protokolların yukarıda tanımlanmış gerekler ve başarım ölçütlerine göre karşılaştırılmaları Tablo 1'de görülebilir. Tabloda, A/E:Akademik/Endüstriyel Öneri, RT Kap.: Gerçek zamanlı veri iletim kapasitesi, nRT Kap.: Gerçek zamanlı olmayan veri iletimi, EşZ: Eşzamanlama protokolu

ve hassasiyet kısaltmaları kullanılmıştır. Protokolların bilinen güvenirlik destekleri yukarıda belirtilmiştir. Literatürde bulunabilen bilgilere göre, bu destekler; ek donanım ile artıklık (redundancy) sağlamak ve veri hattı (data link layer) katmanında Ethetrnet için zaten standart olan CRC kullanımından ibarettir.

Tablo 1: Paylaşımlı ortam Endüstriyel Ethernet Protokolları.

A/E	**Protokol**	**Ortama Erişim**	**Gecikme**	**Düğüm Sayısı**	**RT Kap. (bps)**	**nRT Kap.**	**EşZ**
E	EPA [EPA (2004), FELSER (2009)]	TDMA	5ms, 100μs	32, 64	12.28M	85%	IEEE 1588, 10μs, 1μs
E	EPL [EPL (2004), FELSER (2009), DECOTIGNIE (2009)]	Usta yamak	400μs, 5.5ms	4, 150	15.2M, 32M	19.6%, 4.4%	IEEE 1588, 1s
E	Time Critical Control Network (TCNet) [TCN (2004), FELSER (2009)]	Andaç geçirme	2ms/ 20ms/ 200ms	24, 13	58.4M/ 51.2M/ 7.2M, 45.6M/ 40.8M/ 4.8M	0%, 20%	Yok
A	FTT-E [PEDREIRAS et al. (2005)]	Usta yamak	1ms	Belirtilmemiş	36M, 36%	11%	Yönetici düğümden periyodik eşzamanlama mesajı ile
A	VTPE [MORAES et al. (2011)]	Andaç geçirme	5.8ms	256	Ethernet kapasitesinin %40ının altında	Belirtilmemiş	Yok

2.2 Formal Modelleme Yöntemi

Formal modelleme aracı olarak zamanlanmış giriş çıkışlı otomat (Timed I/O Automata-TIOA) aracı kullanılmaktadır [KAYNAR et al. (2003)],[KAYNAR et al. (2003a)]. TIOA gerçek zamanlı sistem analiz ve modellemesi amacı ile oluşturulmuş matematiksel bir iş çerçevesidir.

TIOA modelinde, bir modelin davranışları kesikli (discrete) ya da sürekli (continuous) durum geçişleri ile tanımlanmaktadır. Bu modelleme tekniği ile iki model arasındaki ilişkiler giriş ve/veya çıkış aksiyonları ile sağlanmaktadır. Bu modelleme tekniği sınırlı durum otomata modeline uygun olduğu için seçilmiştir. TIOA modellemenin ayrıntıları aşağıda açıklanmaktadır.

Proje kapsamında geliştirdiği-miz Şekil 1'de görülen katmanlı mimari, bu katmanların davranışları ve katmanlar arası ilişkiler de TIOA modeline uygun olarak tanımlanmıştır. Bu katmanlar tasarlanırken, yalnızca katmanların kendi modellerinin oluşturulması yeterli olmamaktadır. Arayüz ve Güvenirlik katmalarının paylaşımlı erişim alanı (shared medium) ile olan arayüzünü tanımlayabilmek, ağ paketlerinin nasıl gönderilip alındığını da gösterebilmek amacı ile paylaşımlı ortam modeli de geliştirilmıştır.

Kesikli ve/veya sürekli zaman değişiklikleri ile tariflenen sistemlere zamanlı sistemler adı ver-

ilmektedir. Sistem davranışı kesikli ya da sürekli zaman değişiklikleri ve değişkenlerin aldığı değerler ile tanımlanan durumlar (state) ile belirtilmektedir. Sistemdeki her değişken (variable) bir durağan tip (statik) ve bir de zaman değişimlerini belirten dinamik tip (dynamic type) ile anlatılmaktadır.

Değişkenler, dinamik tiplerinin aldığı değer tipine göre ikiye ayrılmaktadır: dinamik tipi sürekli değerler alan değişkenler analog değişkenler, dinamik tipi kesikli değerler alan değişkenler ise kesikli (discrete) olarak adlandırılmaktadır.

TIOA modelinde sistem davranışı zaman anlarında sistem değişkenlerinin aldığı değerler ile tanımlanmaktadır. Modeldeki değişkenlerin her zaman anındaki aldığı değerler o değişkenin gezingesini (trajectory) τ oluşturmaktadır. Bir sistem modelindeki muhtemel bütün gezingeler $traj s(X)$ kümesini oluşturmaktadır.

Bir gezingenin aldığı ilk değer τ.fval ile gösterilmekte, son değer ise τ.lval ile gösterilmektedir. TIOA modelinde oluşturulan her gezingenin bir de limit zamanı vardır ve τ.ltime ile gösterilir. TIOA modelinde iki gezingeyi peşpeşe ekleyerek yeni bir gezinge oluşturulabilmektedir.

TIOA sistem modelinde modellenen sistemin dış dünya ile ilişkisi kesikli (discrete) aksiyonlar ile olmaktadır. Sistemdeki aksiyonlar sisteme giren (input) ve sistemden çıkan (output) aksiyonlar olarak ikiye ayrılmaktadır. Sistem tanımındaki aksiyonlar A kümesi ile gösterilmektedir.

TIOA model tanımındaki gezinge ve aksiyonlar kullanılarak hibrit (kesikli ve sürekli birlikte) diziler ($\tau_0 a_1 \tau_1 a_2 \cdots$) oluşturmak mümkün olmaktadır. Böylece sistemin bir gezinge sonrası aksiyonunu belirtmek mümkün kılınmıştır.

TIOA Tanımı

Zamanlı sistem davranışını gerçekleyebilmek amacı ile TIOA modeli, değişkenlerce tanımlanan durumlardan (state) oluşan durum makinaları şeklinde tanımlanmaktadır. Modeldeki durum değişiklikleri kesikli aksiyonlar (discrete actions) ya da sürekli gezingeler ile belirtilmektedir.

Bir TIOA modeli 8 elemanlı bir küme ile tanımlanmaktadır. Küme içeriği ve küme elemanlarının tanımları sembolik A otomatı üzerinden anlatılmaktadır.

$A = (X, Q, Q_0, I, O, H, D, T)$:

- X değişken kümesini (set of variables) belirtmektedir.
- $Q \subseteq val(X)$ durum kümesini (set of states) belirtmektedir.
- $Q_0 \subseteq Q$ boş olmayan başlangıç durumu(ları) kümesini (set of start states) belirtmektedir.
- I sistem modeline dışarıdan giren aksiyonlar kümesini (set of input actions) belirtmektedir.
- O sistem modelinden çıkan aksiyonlar kümesini (set of output actions) belirtmektedir
- H sistemin aksiyonlar kümesini belirtmektedir. Sisteme ilişkin bütün aksiyonlar kümesi $A = I \cup O \cup H$ ile belirtilmektedir.
- $D \subseteq Q \times A \times Q$ sistemdeki kesikli geçişleri (set of discrete transitions) belirtmekte ve $q \xrightarrow{a} q'$ şeklinde gösterilmektedir. Bir TIOA bir durumdan başka bir duruma bir aksiyon yoluyla geçmektedir.

- T gezingeler kümesini (set of trajectories) belirtmekte ve $\tau \in T$ şeklinde bütün gezingeleri içermektedir.

Bu çalışmamızda kullandığımız TIOA notasyonunda ortamla etkileşim sağlayan dış aksiyonlar (external actions) kümesini $E = I \cup O$ olarak tanımlanmaktadır. Benzer şekilde A tarafından yerel kontrol edilen aksiyonlar (locally controlled actions) kümesi de $L = O \cup H$ tanımlanacaktır.

TIOA Modeli ile Gerçek Zamanlı Sistem Oluşturma

Gerçek zamanlı sistemler oluştururken sistemin bütününü tek seferde çıkarmak yerine sistemin birleştirilebilir parçalarını tek tek oluşturup daha sonra bu parçaları birleştirerek sistemi oluşturmak, eğer mümkün ise, tasarım kolaylığı sağlamaktadır. TIOA modeli buna olanak sağlamaktadır. TIOA ile modellenmiş iki sistem ya da alt sistem birleştirilerek bütünleşik bir sistem oluşturmak mümkündür. İki TIOA modelinin birleşiminden sonra oluşan yeni model parametreleri ($A = A_1||A_2$) ve bu parametrelerin nasıl oluşturulduğu aşağıda anlatılmaktadır.

- $X = X_1 \cup X_2$
- $Q = \{x \in val(X) | x\lceil X_i \in Q_i, i = 1,2\}$, $x\lceil X_i$, x değiskeninin $i = 1,2$ değerleri için kısıtlanmasıdır.
- $Q_0 = \{x \in val(X) | x\lceil X_1 \in Q_{1,0} \wedge x\lceil X_2 \in Q_{2,0}\}$
- $I = (I_1 \cup I_2) - (O_1 \cup O_2)$, $O = O_1 \cup O_2$, $H = H_1 \cup H_2$
- $x, x' \in Q$ ve $a \in A$ için, $x \xrightarrow{a} x' \iff i = 1,2$ için, Ya $a \in A_i$ ve $x\lceil X_i \xrightarrow{a} x'\lceil X_i$ ya da $a \notin A_i$ ve $x\lceil X_i = x'\lceil X_i$
- $T = \{\tau \in trajs(X) | \tau \downarrow X_1 \in T_1 \wedge \tau \downarrow X_2 \in T_2\}$. $\tau \downarrow X_i$ τ gezingesini $i = 1,2$ için X_i'deki değişkenlere kısıtlamaktadır.

Yukarıda da görüldüğü gibi iki TIOA modelini birleştirmek iki modelin giriş ve çıkış aksiyonlarını senkronize ederken iki modelin iç aksiyonları bağımsız olarak işlemeye devam etmektedir.

TIOA yaklaşımı ile elde ettiğimiz modeller Bölüm 3.2, 3.9 ve 3.8'de tartışılmakta ve sunulmaktadır.

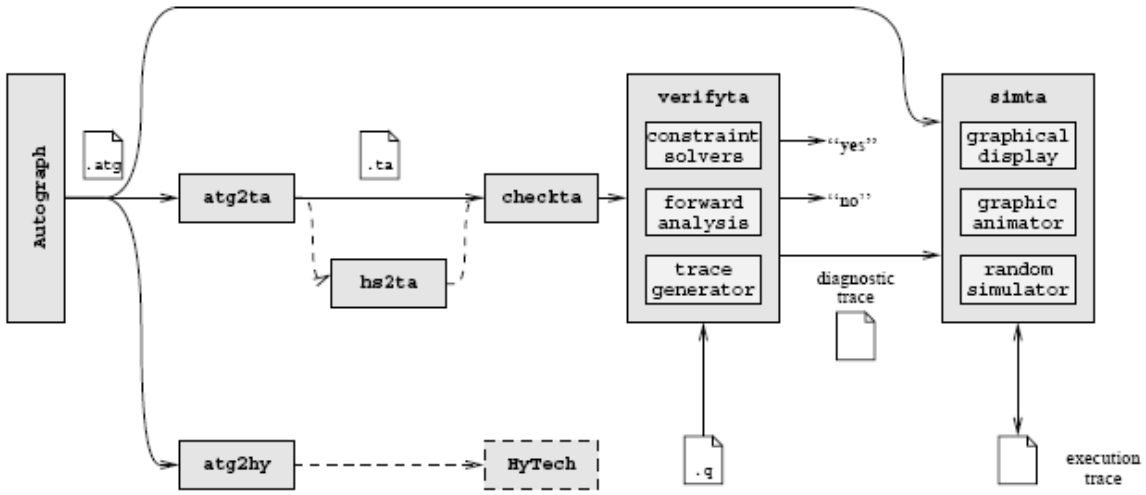
2.3 Doğrulama Aracı Olarak UPPAAL

Geliştirilmekte olan işçerçevesinin gerçek zamanlı çalışacak protokol ileleri geliştirmekte kullanılacak olması sebebi ile işçerçevesinin modelleme, analiz ve doğrulamasının yapılabilmesi için uygun bir ortam ihtiyacı bulunmaktadır. Bu amaca yönelik yapılan çalışmalar sonucunda Uppsala ve Aalborg üniversitelerinince ortaklaşa geliştirilen UPPAAL ortamının kullanılmasına karar verilmiştir.

UPPAAL genellikle kontrol ve haberleşme protokolü uygulamaları gibi zaman kritik uygulamaların analiz ve doğrulanmasında kullanılan gerçek zamanlı bir modelleme aracıdır [LARSEN et al. (1994)].

UPPAAL 3 temel bileşenden oluşmaktadır. Bunlar: tanımlama dili, simulatör ve model kontrolörüdür. UPPAAL tanımlama dili olarak Timed Automata kullanmaktadır. UPPAAL'ın

tanımlama dili Timed Automata'nın yanı sıra durum modellemesinde kullanılan değişkenler ile tamamlanmıştır. Simulatör ve model kontrolörü ise şekil 4'de verildiği gibi birlikte çalışarak sistemin davranışsal (behavioral) ve sintax yönünden doğruluğunu kontrol etmektedir. UPPAAL, kullanıcının modellenmiş sistemin kontrolünü yapabilmesi amacı ile bir takım kısıt (constraint) tanımlaması yapabilmesine olanak sağlamaktadır. Kullanıcı tarafından yazılan bu kısıtlar modelde oluşabilecek bütün durum geçiş ihtimalleri dikkate alınarak test edilmektedir. Sistemin doğrulaması da bu kısıtların geçerli olup/olmamasına bağlı olarak yapılmaktadır.



Şekil 4: UPPAAL Çalışma Prensibi

UPPAAL ile sistem modellemesi grafiksel bir arayüz üzerinden yapılmaktadır. Grafik arayüzü üzerinden modellenen sistem *atg2ta* derleyicisi vasıtası ile timed automata sentaksına uygun olarak yazılı hale çevrilir. Modelin kontrolü *verifyta* aracı ile gerçekleştirilir. Sentax olarak kontrol edilen model *simta* simülasyon aracı ile görsel olarak simule edilir. Kullanıcı bu sayede modellediği sistemdeki durum geçişlerini takip edebilme, hatalı olduğunu düşündüğü kısımlara müdehale edebilme fırsatı bulur.

Bir başka deyişle UPPAAL geliştirilen modelin gramer olarak doğru olup olmadığını otomatik olarak gerçekleştirebilmektedir. Sistemin davranışsal kontrolü, yani beklenen fonksiyonları doğru olarak yerine getirip getiremediğinin kontrolü için kullanıcı etkileşimine ihtiyaç duyulmaktadır. Kullanıcı, UPPAAL'ın doğrulama (verification) arayüzü üzerinden doğrulanmasını istediği fonksiyonları Timed Automata sentaxına uygun olarak girebilmektedir. Burada durum geçişleri, zamanlama, değişken değerleri gibi fonksiyonel işleyişin kontrolü yapılabilmektedir. UPPAAL girilen bu kıstaslar (constraint) UPPAAL tarafından olası bütün durum geçişleri taranarak kontrol edilir. Bu sayede verilen bir modelin hem sentaks olarak ve hem de davranışsal olarak kontrolü yapılabilmektedir. örnek bir UPPAAL uygulaması [WANG YI, (1997)]'de verilmektedir.

Projede geliştirilen işçerçevesi TIOA olarak modellenmiştir. TIOA'da sistem durumları (state) sistemin değişkenlerinin aldığı değerlere bağlı olarak belirlenmektedir. UPPAAL ortamının zamanlı otomat temelli bir yapı üzerine kurulması ve dış (external) değişkenler tanımlanabilmesi işçerçevesinin modellenmesinde büyük avantaj sağlamaktadır.

2.4 Güvenirlik Tanımları

Güvenilirlik, bir sistemin gereksinimlerini ispat edilebilir bir biçimde yerine getirmesi olarak tanımlanmaktadır. [AVIZIENIS et al. (2004), NELSON (1990)] Tanım itibariyle güvenilirlik aşağıda belirtilen öz-nitelikleri (attribute) kapsamaktadır: [AVIZIENIS et al. (2004), NELSON (1990)]

Hazır Olma (Availability): Sistemin doğru servis vermeye hazır olması.

İşleyişin Sürdürülebilmesi (Reliability): Sistemin ilerleyen zaman içerisinde doğru servis vermeye devam edebiliyor olması.

Güvenlik (Safety): Sistemin kullanıcı seviyesinde geri dönülemeyecek hatalara yol açmıyor olması.

İdame Edilebilirlik (Maintainability): Sistemin ihtiyaç duyulan / duyulabilecek bakım ve onarıma elverişli olması.

Entegre Olabilirlik (Integrity): Sistem durum değişikliklerinin tasarlanmış bir sıralamaya uygun olması, beklenmeyen durum değişikliklerinin bulunmaması.

Yukarıda belirtilen öz-nitelikler uygulama alanına göre farklı önemler taşımakla birlikte, bir sistemin güvenilir olabilmesi için bu öz-niteliklere sahip olması ve bu öz-niteliklere sahip olduğunun ispat edilebiliyor olması gereklidir. Bir sistemin güvenilir yapılabilmesi için yukarıda belirtilen öz-nitelikleri gözetecek şekilde, sistem güvenilirliğini tehdit eden unsurlara karşı tasarım ve işletim aşamalarında önlemler alınması gerekir. Sistem güvenilirliğini tehdit eden unsurlar 3 temel başlık altında incelenir. [AVIZIENIS et al. (2004), NELSON (1990), AVIZIENIS et al. (2001)] Bu başlıklar ve aralarındaki ilişki aşağıda anlatılmaktadır.

Güvenilirlik Tehditleri

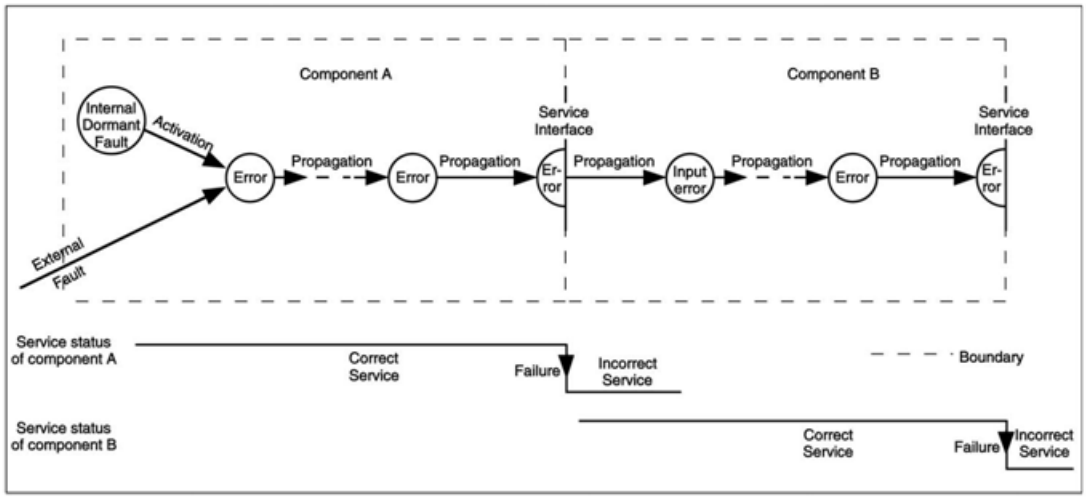
Güvenilirlik tehditleri oluşma şekilleri ve etki düzeylerine bağlı olarak 3 ana grupta toplanmıştır. Bunlar:

Bileşen Seviyesi Hatalar (Faults): Oluştuğu bileşenin sistem işleyişinde kullanılmamış olması nedeniyle henüz sistem seviyesi bir hataya yol açmamış hatalar.

Sistem Seviyesi Hatalar (Errors): Etkileri henüz kullanıcı seviyesinde hissedilmemekle birlikte sistemin bir veya daha fazla işlevinde sorun oluşturmuş hatalar.

Kullanıcı Seviyesi Hatalar (Failures): Etkileri sistem kullanıcısı veya kullanıcıları (operatör, başka sistemler vb.) seviyesinde görülmüş hatalar.

Yukarıda belirtilen hata grupları arasındaki neden sonuç ilişkisi Şekil 5'de gösterilmektedir.



Şekil 5: Güvenilirlik Tehditleri [AVIZIENIS et al. (2004)]

Şekil 5'de de görüldüğü gibi bileşen seviyesinde oluşup çözülemeyen hataların kullanıcı seviyesine kadar ilerlemesi sonucunda sistem doğru işleyişini yerine getiremez hale gelmektedir. Sistemin doğru işlevini yeniden kazandırılabilmesi için hata durumlarının ortadan kaldırılması gerekir.

Sistemde çeşitli seviyelerde oluşan hataların giderilmesi ve güvenilirliğin tahsis edilebilmesi için yürütülen faaliyetler Güvenilirlik Faaliyetleri olarak adlandırılır. [AVIZIENIS et al. (2004), NELSON (1990)] Güvenilirlik Faaliyetleri'ne ilişkin genel tanımlar aşağıda verilmektedir.

Güvenilirlik Faaliyetleri

Güvenilirlik faaliyetleri hatalara etki etme zamanlarına göre 4 grupta toplanmıştır. [AVIZIENIS et al. (2004), NELSON (1990), MEYER (1999)] Bu faaliyetler:

Bileşen Seviyesi Hata Önleme (Fault Prevention) Faalyetleri: Tasarım aşamasında, sistemde oluşabilecek hataları önlemeye yönelik yürütülen faaliyetler. Tasarlanmış bir sistemde görülen hataların kayıtlarının tutularak sistem tasarım sürecinde modifikasyonlar yapmak [CHILLAREGE et al. (1992), PAULK et al. (1993)] en sık rastlanan örneklerindendir.

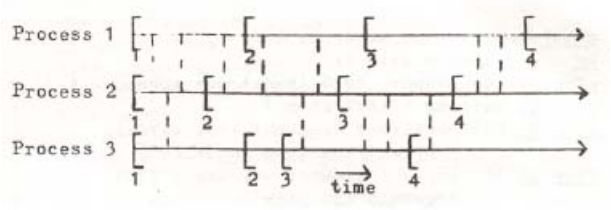
Bileşen Seviyesi Hata Ayıklama (Fault Removal) Faaliyetleri: Hata bulma, sınıflandırma ve doğrulama aşamalarından oluşan bu faaliyetler sistem tasarımı aşamasında çıkan hataların ortadan kaldırılmasını amaçlamaktadır. Sistem doğrulama (verification) hata ayıklama öncesinde hatanın varlığının teyidi ve hata ayıklama sonrasında sistem gereksinimlerinin sağlandığının gösterilebilmesi için uygulanan bir yöntemdir.

Bileşen Seviyesi Hata Öngörme (Fault Forecasting) Faaliyetleri: Tasarımı tamamlanmışolan sistemin işletmeye alındıktan sonra gözlenerek kullanıcı seviyesi hatalara yol açan durum değişikliklerinin (state changes) belirlenmesi faaliyetleridir.

Bileşen Seviyesi Hata Etkilerini Yok Etme (Fault Tolerance) Faaliyetleri: Sistem işleyişi sırasında oluşabilecek hataların algılanması ve giderilmesi faaliyetleridir. [AVIZIENIS (1967)] Sistem hatalarının etkilerinin yok edilmesi faaliyetlerine genel olarak "sistem kurtarılması

(system recovery)" adı verilir. [AVIZIENIS et al. (2004)] Sistem işleyişinde ilklendirme (reset) zorunluluğu olmadan sistemin kurtarılması için sıklıkla kullanılan yöntemlerden bir tanesi Kurtarma Noktası Oluşturma ve Geri Dönme (checkpointing and rollback) yöntemidir. [HORNING et al. (1974), RANDELL (1975)]

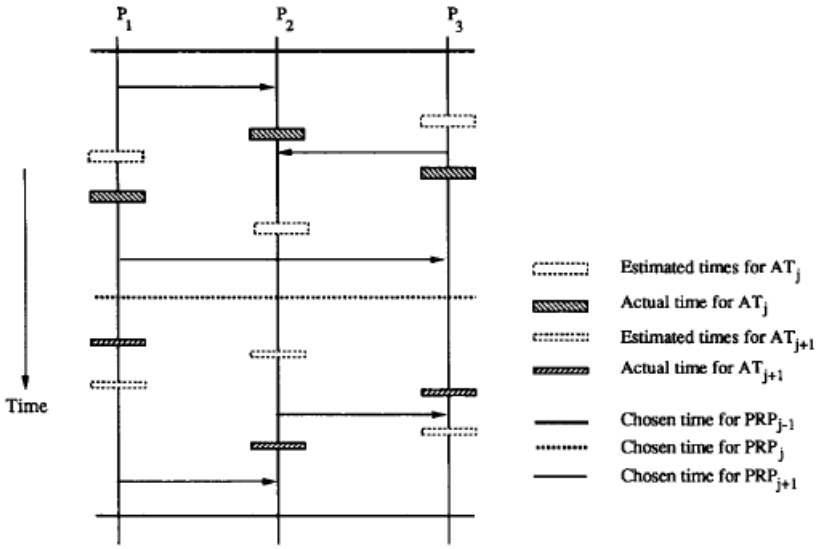
Dağıtık (distributed) sistemlerde bu yöntem uygulanırken, sistemde dağıtık olarak bulunan işlevler, oluşabilecek hata durumlarına karşı bulundukları durumları (state) kaydetmektedirler. Herhangi bir hata olması durumunda sistem tarafından işletilen hata kurtarma senaryosu kapsamında işlevler önceden kaydettikleri durumlara geri dönmektedir. Her ne kadar ilk bakışta uygulaması kolay bir yöntem olarak görünse de kayıtlı durumlara geri dönme (rollback) mekanizması bazı durumlarda, peşpeşe sıralanan geri dönmelere yol açarak sistemin ilk durumuna dönmesine, bir başka deyişle sistemin ilklendirilmesine yol açmaktadır. Domino etkisi (domino effect) olarak bilinen bu durum Şekil 6'de gösterilmektedir.



Şekil 6: Domino Etkisi [RANDELL (1975)]

Şekil 6'de kesikli dik çizgiler işlevler arası haberleşmeyi göstermektedir. Şekilde verilen üç işlev, kendi içlerinde periyodik olarak kurtarma noktaları tanımlamaktadır. Örneğin, 3 numaralı işlevde (Process 3), 4. kurtarama bloğu tanımlandıktan hemen sonra hata olması durumunu ele alalım. Bu durum 3 numaralı işlevin 4 numaralı kurtarma noktasına geri dönmesine neden olmaktadır. 3 numaralı işlev, hata oluştuğu an ile 4 numaralı kurtarma bloğu arasında diğer iki işlev ile mesaj alışverişi yaptığı için, işlev 4 numaralı kurtarma noktasına geri döndüğünde diğer iki işlev de 3 numaralı işlev ile uyumlu bir duruma (state) geçebilmek için bir önceki kurtarma noktasına geri dönmek durumunda kalacaktır. Aynı mantık ile bu geri dönme mekanizması, peşi sıra devrilen domino taşları gibi sistemin ilk durumuna geri dönmesine neden olacaktır.

Domino etkisinin durdurulabilmesi amacıyla [SHIN (1984)] ve [RAMANATHAN (1993)]'de işlevler arasında kullanılan haberleşme mekanizmaları önerilmektedir. Her iki makalede de önerilen haberleşme mekanizmaları sistemdeki mesaj trafiğine ek yük getirmekle beraber, [RAMANATHAN (1993)]'de bahsi geçen haberleşme mekanizmasının daha az bir yük getirdiği görülmektedir. Belirtilen haberleşme mekanizması, haberleşme yükünü azaltabilmek amacıyla işlevler arasında ortak referans zaman kullanımını önermektedir. Makalede önerilen kontrol noktası tanımlama ve geri dönme işleyişi Şekil 7'te verilmektedir.

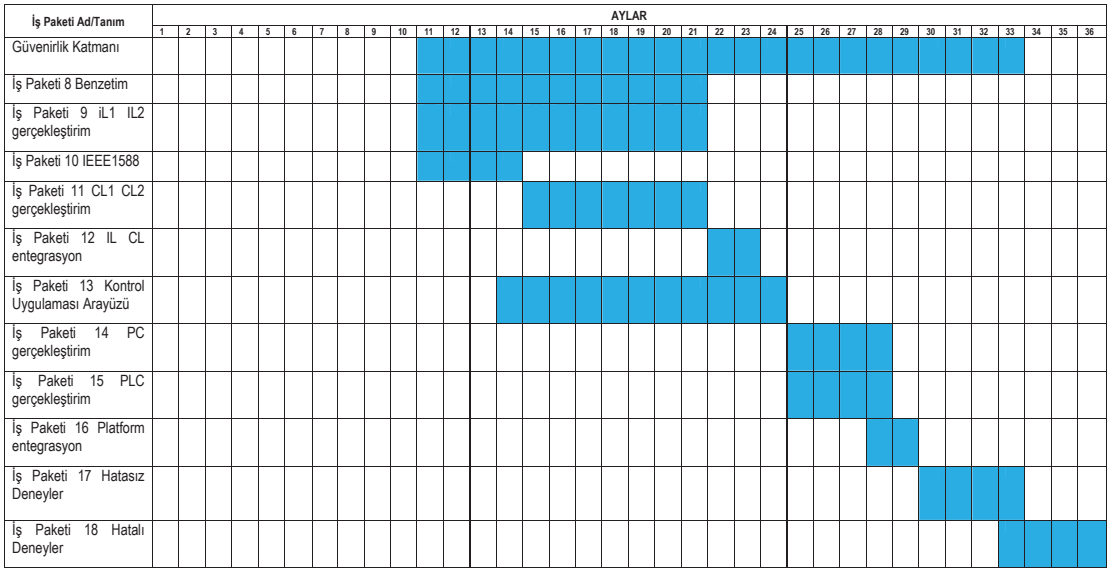


Şekil 7: Kontrol Noktası Oluşturma ve Geri Dönme [RAMANATHAN (1993)]

Şekil 7'te verilen uygulama temelde dağıtık düğümlerin birbiri ile eşzamanlı (senkron) çalıştığı varsayımına dayanmaktadır. Eşzamanlı çalıştığı bilinen düğümler için kabul testleri (acceptance test) yapılacak ortak bir zaman periyodu belirlenmektedir. Kabul testlerini belirlenen süre içerisinde tamamlayamayan düğüm diğerlerine gecikmesini bildirmekte bu sayede birbiri ile senkron kurtarma noktaları tanımlanabilmektedir. Her ne kadar mesajlaşma yükünü çok azaltmış da olsa ağda halen yalnızca Kurtarma Noktası tanımlama amaçlı kullanılan mesajlar bulunmakta ve bu durum ağ verimliliğini olumsuz etkilemektedir. Bizim geliştirdiğimiz iş çerçevesinde dağıtık düğümler arasında eşzaman öngörülmesinin yanı sıra düğümler arasındaki haberleşme Paylaşımlı Ortam (Shared Medium) üzerinden sağlanmaktadır. Geliştirilen güvenilirlik katmanı bu iki özelliğin getirdiği avantajları kullanacak şekilde tasarlanmış ve ek mesaj yükünün tamamen ortadan kaldırılması hedeflenmiştir.

3 Gereç ve Yöntem

Projede önerilen ve daha sonra 2. Dönem raporunda proje işleyişine göre yeniden düzenlenen iş paketlerinin zaman planlaması Şekil 8'de görülmektedir.



Şekil 8: İş Paketleri.

Proje sürecinde geliştirilen iki katmanlı gerçek zamanlı endüstriyel haberleşme mimarisi, D^2RIP (Dynamic Distributed Real-time Industrial communication Protocol) bu mimarinin güvenirlik katmanı ile birlikte son hali, D^3RIP (Dynamic Distributed Dependable Real-time Industrial communication Protocol) olarak adlandırılmaktadır.

Bu iş paketleri sonucunda elde edilen çıktılar aşağıda sıralanmaktadır:

1. Paylaşımlı ortamda gerçek zamanlı haberleşme için önerilen D^2RIP iş çerçevesinin zamanlanmış giriş çıkışlı otomat (Timed I/O Automata-TIOA) aracı ile formal modellemesi

2. Bu modele uygun olarak tanımlanan Koordinasyon ve Arayüz katmanlarının PC, PLC ve gömülü geliştirme ortamında gerçeklenmesi ve 3 düğümlü bir örnek üzerinde testleri.

3. IEEE 1588 eşzamanlı protokolun yazılım yoluyla gerçekleştirilmesi ve testleri.

4. Bu gerçeklemeye uygun olarak ve hem hızlı testlerin yapılabilmesi hem de karşılıklı doğrulama amacıyla benzetim yazılımı geliştirilmesi ve testleri

5. D^2RIP protokolunun UPPAAL yazılım aracıyla doğrulanması. *Bu çıktı proje önerisinde öngörülmemiştir.*

6. TIOA ile modellenen sistemlerin UPPAAL modellerinin otomatik oluşturulması için yeni bir algoritma. *Bu çıktı proje önerisinde öngörülmemiştir.*

7. D^2RIP gerçeklemesinin başarımının artırılması amacıyla optimizasyonunu yapılması ve son haline getirilmesi. Bu son gerçekleme ile kapsamlı bir örnek üzerinde detaylı başarım analizi sonuçları.

8. Güvenirlik katmanı tasarımı ve katmanın eklenmesi ile D^3RIP iş çerçevesinin zamanlanmış giriş çıkışlı otomat (Timed I/O Automata-TIOA) aracı ile formal modellemesi

9. D^3RIP protokolunun UPPAAL yazılım aracıyla doğrulanması. *Bu çıktı proje önerisinde öngörülmemiştir.*

10. D^3RIP modeline uygun olarak güvenir protokolunun gerçeklenmesi ve testleri.

Aşağıdaki bölümlerde önce proje yürütülmesi ile ilgili bilgiler ardından yukarıda sıralanan çıktılar için izlenen yöntemler açıklanmaktadır. Bu bölümde bu çıktıların elde edilmesinde kullanılan yöntemler ve gereçler anlatılmaktadır. Bölüm 3.2 ve 3.3'da anlatılan çıktılar daha önceki rapor dönemlerinde tamamlanmış ve detaylı olarak anlatılmıştır. Bu sebeple bu bölümler özetlenecek, diğer çıktılar detaylı olarak anlatılacaktır.

3.1 Proje Yönetimi, Tezler, Makale ve Bildiriler, Ortak Çalışmalar ve Ziyaretler

Bursiyerler ve Tezler:
Projede toplam 6 öğrenci çalışmıştır. Tamamlanan ve referans verilen yüksek lisans tezleri TÜBİTAK proje sistemine yüklenmiştir. Bu öğrenciler ve yaptıkları işler aşağıda sıralanmıştır:

1. Yusuf Bora Kartal: Doktora öğrencisi yarı zamanlı bursiyer olarak projenin başından itibaren çalışmıştır. Bölüm 3.7, 3.8, 3.9 ve 3.10'de anlatılan sonuçların elde edilmesinde görev almıştır. Proje yürütücüsü danışmanlığında yürüttüğü Doktora tezi tamamen bu projede yaptığı çalışmalardan oluşmaktadır. Beklenen tez sunum tarihi Haziran 2013'tür.

2. Ahmet Korhan Gözcü: Yüksek Lisans öğrencisi olarak projenin başından itibaren çalışmıştır. BİDEB burslusu olduğu için projede gönüllü olarak yer almıştır. Bölüm 3.3'da anlatılan ilk sürüm D^2RIP gerçekleştirim çıktılarının elde edilmesinde görev almıştır. Bu gerçekleme ve çıktıları 3. 4. ve 5. Gelişme Raporlarında detaylı olarak anlatılmıştır. Proje yürütücüsü danışmanlığında yürüttüğü "*Implementation and evaluation of a synchronous time-slotted medium access protocol for networked industrial embedded systems*" başlıklı yüksek lisans tezi tamamen bu projede yaptığı çalışmalardan oluşmaktadır ve Eylül 2011'de başarıyla tamamlanmıştır [GÖZCÜ (2011)].

3. Ulaş Turan: Yüksek Lisans öğrencisi yarı zamanlı bursiyer olarak projenin başından itibaren çalışmıştır. Bölüm 3.3'da anlatılan ilk sürüm D^2RIP gerçekleştirim çıktılarının elde edilmesinde görev almıştır. Gerçekleme ve çıktıları 3. 4. ve 5. Gelişme Raporlarında

detaylı olarak anlatılmıştır. Kendisi proje yürütücüsü ile birlikte proje önerisine belirtildiği şekilde ortak çalışma yapmak üzere 2011'de Erlangen Üniversitesi'ne bir çalışma ziyareti yapmıştır. Bu ziyarette söz konusu kurumda bulunan laboratuvar kontrol sistemi üzerinde protokol yığıtı gerçeklenmiş ve deneyler yapılmıştır. Proje yürütücüsü danışmanlığında yürüttüğü "*Implementing and evaluating the Coordination Layer and time-synchronization of a new protocol for industrial communication networks*" başlıklı yüksek lisans tezi tamamen bu projede yaptığı çalışmalardan oluşmaktadır ve Eylül 2011'de başarıyla tamamlanmıştır [TURAN (2011)].

4. Güray Aybar: Yüksek Lisans öğrencisi yarı zamanlı bursiyer olarak projenin başından itibaren çalışmıştır. Bölüm 3.3'da anlatılan ilk sürüm D^2RIP gerçekleştiriminin benzetimi için bir yazılım geliştirmiş ve donanım gerçeklemesi ile uyumlu sonuçlar elde ederek hem benzetim yazılımını hem de gerçeklemeyi karşılıklı doğrulamıştır. Bu yazılım 4. Gelişme Raporlarında detaylı olarak anlatılmıştır. Proje yürütücüsü danışmanlığında yürüttüğü "*Simulation and performance evaluation of a distributed real-time communication protocol for industrial embedded systems*" başlıklı yüksek lisans tezi tamamen bu projede yaptığı çalışmalardan oluşmaktadır ve Eylül 2011'de başarıyla tamamlanmıştır [AYBAR (2011)].

5. Adem Kaya: Yüksek Lisans öğrencisi yarı zamanlı bursiyer olarak Haziran 2012'den itibaren projede çalışmaktadır. Bölüm 3.4 çıktılarının ve Bölüm 4.3 'de anlatılan sonuçların elde edilmesinde görev almıştır. Proje yürütücüsü danışmanlığında yürüttüğü yüksek lisans tezi tamamen bu projede yaptığı çalışmalardan oluşmaktadır ve beklenen tez sunum tarihi Eylül 2013'tür.

6. Ömer Sezer: Yüksek Lisans öğrencisi olarak Haziran 2012'den itibaren projede çalışmaktadır. TUBITAK Uzay personeli olduğu için projede gönüllü olarak yer almıştır. Bölüm 3.11 çıktılarının ve Bölüm 4.4 'de anlatılan sonuçların elde edilmesinde görev almıştır. Proje yürütücüsü danışmanlığında yürüttüğü yüksek lisans tezi tamamen bu projede yaptığı çalışmalardan oluşmaktadır ve beklenen tez sunum tarihi Eylül 2013'tür.

Makale ve Bildiriler:
Aşağıda sıralan Makale ve Bildiriler TÜBİTAK Proje sistemine Sonuç raporu ile birlikte yüklenmiştir.

- Bölüm 3.2'de anlatılan D^2RIP İş Çerçevesi TIOA Modeli, IEEE Transactions on Parallel and Distributed Systems Dergisi'nde yayınlanmıştır [SCHMIDT (2012)].

- Bölüm 3.3'de anlatılan gerçekleme IEEE 20. Sinyal İsleme ve İletisim Uygulamalari Kurultayı'nda sunulmuştur [GOZCU et al. (2012)].

- Bölüm 3.7'de anlatılan doğrulama IEEE 20. Sinyal Isleme ve Iletisim Uygulamalari Kurultayı'nda sunulmuştur [KARTAL et al. (2012)]. Bildiri SIU 2012 En İyi Uygulama Makalesi Ödülü için finale kalmıştır. Bu doğrulamanın detaylı hali ve TIOA modellerinin UPPAAL modellerine dönüştüren algoritma IEEE Transactions on Software Engineering Dergisi'nde değerlendirme altındandır [KARTAL et al. (2013)].

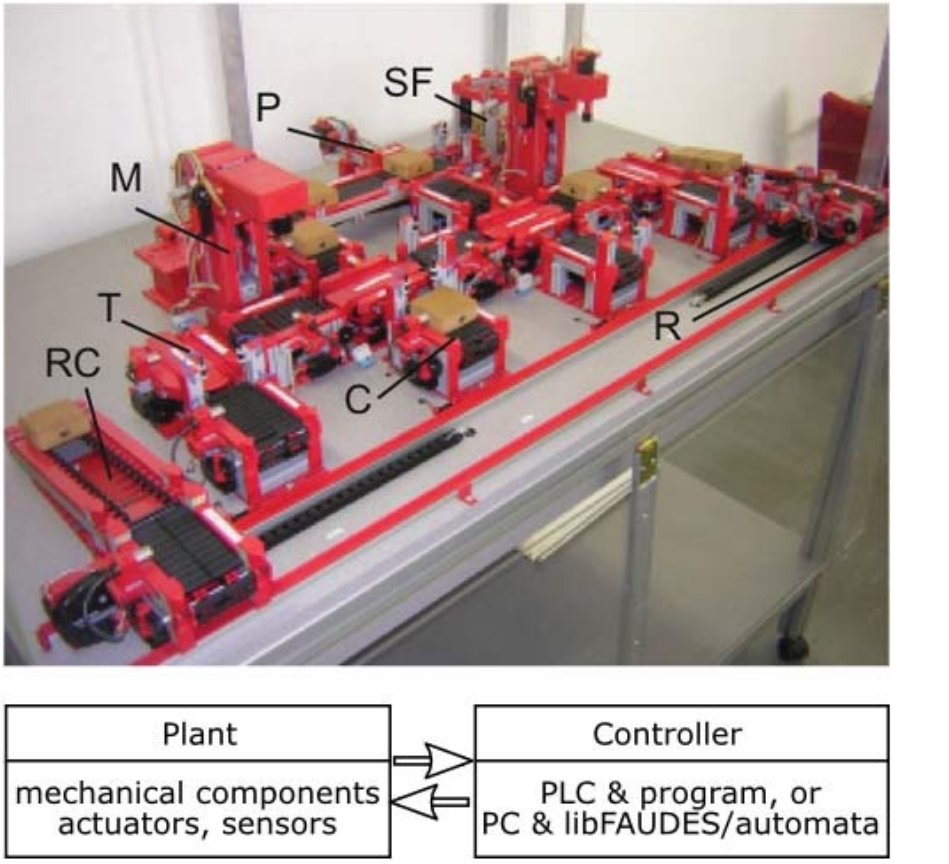
- Bölüm 3.4'de anlatılan yüksek başarımlı gerçekleme makale olarak Elsevier Computer Standards and Interfaces Dergisi'nde değerlendirme altındadır [KAYA et al. (2013)].

- Bölüm 3.8'de anlatılan Güvenirlik Katmanı ile birlikte D^{3}RIP İş Çerçevesi TIOA Modeli makale olarak IEEE Transactions on Parallel and Distributed Systems Dergisi'ne sunulmak üzere hazırdır [KARTAL et al. (2013a)].

Ortak Çalışmalar ve Ziyaretler:

Proje önerisinde Erlangen Üniversitesi'nde Prof. Dr. Thomas Moor'un ekibi ile ortak çalışmalar yapılacağı öngörülmüştür. Bu ekipten Dr. Klaus Werner Schmidt daha sonra Çankaya Üniversitesi'nde çalışmaya başlamıştır. Bu sebeple Erlangen Üniversitesine ek olarak Çankaya Üniversitesi ile de ortak çalışmalar yapılmıştır.

Proje önerisinde planlandığı gibi Erlangen Üniversitesi'nde Prof. Dr. Thomas Moor'un ekibi ile Erlangen Üniversitesinde ortak çalışmalar yapmak ve oradaki laboratuvar olanaklarından yararlanmak üzere proje yöneticisi Ece Schmidt ve yarı zamanlı bursiyer Ulaş Turan 2011 yılında bu kuruma bir çalışma ziyareti gerçekleştirmişlerdir. Bu üniversitede kullandığımız laboratuvar düzeneğinin bir resmi Şekil 9'de görülmektedir. Bu düzenek ile ilgili detaylı bilgi [ERL (2006–2013)]'de görülebilir.



Şekil 9: Erlangen Üniversitesi Üretim Sistemi Modeli.

Protokol Katmanlarının geliştirilmesi sırasında yapılan bu ortak çalışmanın ana amacı Koordinasyon Katmanı (KK) ve Kontrol Uygulaması (KU) arayüzü ve entegrasyonunun yapılması

ve bunu takiben elde edilen protokol yığıtı gerçeklemesinin testlerinin yapılmasıdır. D^2RIP mimarisinde KKnın çalışması KU tarafından yaratılan mesajlara bağlıdır. Bu mesajların içinde KUnın kendi çalışması için ihtiyaç duyduğu bir verinin yanısıra KK çalışmasını belirleyen bir parametre taşınmaktadır. Bu parametre çoğunlukla o anda uygulamanın ihtiyaç duyduğu haberleşme bantgenişliğinin dağıtık olarak hesaplanmasında kullanılmaktadır.

KK ve KU entegrasyonu KUnın KK'na yolladığı mesajlar içinde doğru parametrelerin olması ve KK'nın bu parametreleri işleyebilmesini amaçlamaktadır. Bu sebeple KK ve KU arasında bir arayüz gereklidir.

Bu arayüzün geliştirilmesi sırasında örnek olarak endüstriyel otomasyonda standart bir kontrol uygulaması olan bir üretim sisteminin mantıksal kontrolu seçilmiştir. Bu kontrol uygulamaları büyük fabrika ortamlarında dağıtık olarak gerçekleştirilmektedirler. Çalışmalarımızdaki kontrol uygulamaları Erlangen Üniversitesinde geliştirilmış olan libfaudes [MOOR et al. (2008), LIB (2006–2013)] yazılımı ile dağıtık olarak gerçekleştirilmıştir.

Erlangen üniversitesie yapılan ziyaret sırasında projede ortak çalışma yapılması öngörülen araştırma grubunun yöneticisi Prof. Dr. Thomas Moor tarafından Kontrol Uygulaması ve bu uygulamanın gerçekleştirilmesini sağlayan libfaudes kütüphanesi hakkında destek verilmiştir. KK ve KU arasındaki arayüz ve entegrasyonun tamamlanması ile protokol mimarisinin gerçekleştirilmesi temel olarak tamamlanmış ve mimarinin doğru çalışmasını test eden ilk deneyler yapılabilmiştir.

Arayüzünün ve bütün prokol mimarisinin test edilebilmesi için laboratuvarda bir deney ortamı tasarlanmıştır. Bu deney ortamında bir üretim sistemi ve iki kontrolcu bulunmaktadır. Kontrol uygulaması bu iki kontrolcu üzerinde dağıtık olarak çalışmakta ve uygulama parçaları arasındaki haberleşme D^2RIP protokolu ile paylaşımlı ortam Ethernet üzerinden sağlanmıştır.

Yapılan ziyaretle ilgili olarak Erlangen Üniversitesi'nden Prof. Dr. Moor'un yazdığı rapor Ekler bölümünde sunulmaktadır.

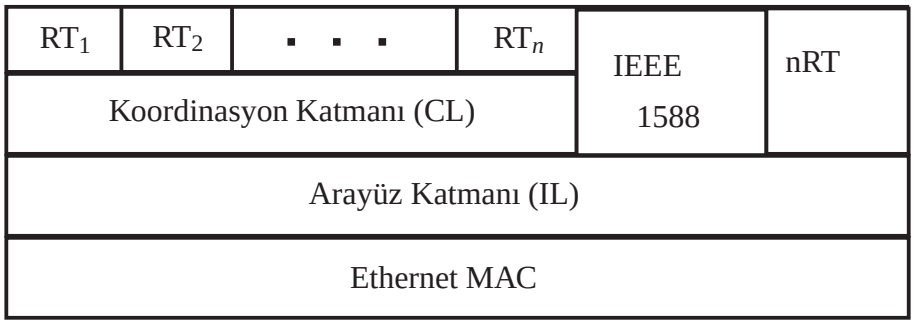
Yapılan deney sonuçlarında D^2RIP protokolunun doğru çalıştığı görülmüştür.

Proje kapsamında yapılan çalışmalar Erlangen üniversitesi'nde Prof. Dr. Thomas Moor'un araştırma grubu ile proje yürütücüsü arasında süre gelen ortak çalışmaların devamı niteliğindedir. Yapılan ziyarette Prof. Moor ile bu genel olarak çalışmalar ile ilgili toplantılar da yapılmış ve görüş alışverişinde bulunulmuştur.

Eğitim Faaliyetleri:
Projenin konusu olan gerçek zamanlı endüstriyel haberleşme ağları ile ilgili konular Proje Yöneticisi tarafından verilen Yüksek hızlı ve Gömülü Bilgisayar Ağları başlıklı yüksek lisans dersi kapsamına eklenmiştir. Bu konular anlatılırken projede elde edilen deneyimler öğrenciler ile paylaşılmaktadır.

3.2 D^{2}RIP İş Çerçevesi Modeli

Proje işleyişinde öncelikle Şekil 10'de görülen mimari için genel ve formal bir model oluşturulmuştur.



Şekil 10: Arayüz ve Koordinasyon Katmanları yazılım mimarisi.

Bu amaçla gerçek zamanlı sistemlerin modellenmesi ve analizi amacıyla [KAYNAR et al. (2003), KAYNAR et al. (2003a)] çalışmalarında önerilen matematiksel iş çerçevesi kullanılmıştır. Bu Zamanlı Giriş-Çıkışlı Otomat (Timed I/O Automata) işçerçevesinde sistem durumu atomik olarak ayrık geçişlerle (*discrete transitions*) ya da zaman içerisinde evrilerek (*trajectories*) değişmektedir. Çok bileşenli bir sistemin zamanlı davranışı her bileşenin TIOA modellerinin birleşimi (*composition*) alınarak elde edilebilmektedir. Geliştirdiğimiz protokol yığıtı mimarisinde zamana göre davranışİarı hem ayrık hem sürekli olarak değişen çok sayıda bileşen olması nedeniyle katmanların çalışması ve özelliklerini formal olarak tanımlamak için bu TIOA işçerçevesi uygun bulunmuştur. TIOA ile ilgili önbilgiler Bölüm 2.2'de bulunmaktadır.

Bu işçerçevesine göre TIOA durumları değişken kümeleriyle kodlanmakta ve bu değişkenlerin zamana göre aldıkları değerlerle sistem durumu tanımlanmaktadır. Buna ek olarak bir TIOA modelinin diğer modellerle arayüzü giriş ve çıkış aksiyonları ile tanımlanmaktadır. Çıkış aksiyonları modelin durumuna göre koşullu olarak gerçekleşirken giriş aksiyonları diğer modellerin çıkış aksiyonları olarak her hangi bir zaman gerçekleşebilirler.

Geliştiridiğimiz iş çerçevesinin ilk sürümünde Şekil 10' da görülen Koordinasyon ve Arayüz katmanlarının jenerik TIOA modelleri geliştirilmıştir. Bu modellerde giriş ve çıkış aksiyonları ile; Koordinasyon ve Arayüz katmanları arayüzü, Koordinasyon ve Gerçek zamanlı uygulamalar arasındaki arayüz, Arayüz ve Gerçek zamanlı olmayan uygulamalar arasındaki arayüz, Arayüz katmanı ve Paylaşımlı ortam arasındaki arayüzler tanımlanmaktadır. Bunun yanısıra Koordinasyon ve Arayüz Katmanlarının kendi iç durum değişiklikleri de modellenmiştir. Arayüz Katmanı modelindeki $\mathtt{vIL}_i^{\mathrm{d}}$ ayrık değişkeni ve Koordinasyon Katmanı modelindeki $\mathtt{vCL}_i^{\mathrm{d}}$ ayrık değişkeni sırasıyla ağ düğümü *i* için bu katmanların iç durumlarını tanımlamaktadırlar. Bu değişkenler ve bu değişkenleri güncelleyen iç durum değişimleri fonksiyonlarını tanımlayarak farklı katman modelleri oluşturmak mümkündür. Buna göre Arayüz Katmanı için Gerçek-Zamanlı Erişim Arayüz Katmanı (RAIL) ve Zaman Dilimli (Time-Slotted) Arayüz Katmanı (TSIL) özel modelleri ve Koordinasyon Katmanı için Gerçek-Zamanlı Dinamik Tahsis Protokolü (DART) ve Aciliyete Dayalı Gerçek-Zamanlı Protokol (URT) özel modelleri tanımlanmıştır.

Detaylı olarak 2. Dönem Raporunda ve [SCHMIDT (2012)] makalesinde anlatılan TIOA tabanlı jenerik AK (IL), KK (CL) ve paylaşımlı ortam (SM) modellerinin ayrı ayrı ve birleştirilmış olarak (composition) önemli özellikleri gösterilmiştir. Buna göre:

- Paylaşımlı ortam modeli *SM* pratik olarak uygulanabilirdir.

  - *SM* ilerleyebilir (progressive)dir. Bu özellik modelin her giriş aksiyon dizisine bir çıkış oluşturabilir.
  - Paylaşımlı ortam modeli *SM*'de üstüste binen zamanlarda gönderilme durumu hariç mesaj çarpışması yoktur.

- Bir düğüm *i* için AK modeli IL_i ve bu modelin *SM*'le birleştirilmesiyle elde edilen $IL\_SM := SM||(||_{i\in I} IL_i)$ ilerleyebilir (progressive)dir.
- *IL_SM*'in bütün erişilebilir durumları için paylaşımlı ortamda mesaj çarpışması yoktur. Bu şekilde AK'nın istenen şekilde çarpışmadan kaçınmayı sağladığı gösterilmiştir.
- AK üzerinde üst katmanlar tarafından gönderilen gerçek zamanlı mesajlar için bir gecikme sınırı bulunabileceği gösterilmiştir.
- Bir düğüm *i* için KK modeli CL_i ve bu modelin *SM* ve IL_i'le birleştirilmesiyle elde edilen $CL\_IL\_SM = SM||(||_{i\in I} IL_i)||(||_{i\in I} CL_i$ ilerleyebilir (progressive)dir
- *CL_IL_SM*'in bütün erişilebilir durumları için paylaşımlı ortamda mesaj çarpışması yoktur. Bu şekilde AK-KK birlikte çalışmasıyla kontrol uygulamasından gelen mesajların şekilde çarpışmadan kaçınmayı sağladığı gösterilmiştir.

Bu yukarıdaki özelliklerin sağlanabilmesi ve AK ve KKdan oluşan protokol yığıtının mesaj çarpışması olmadan çalışabilmesi için TIOA durum değişkenlerinin ve durum değiştirme fonksiyonlarının sağlaması gereken özellikler belirlenmiştir. Bu jenerik modellerin özelleştilmesi ile elde edilen DART ve URT KK modelleri ve TSIL ve RAIL AK modellerinin iç durum değişkenlerinin değerlerini belirleyen fonksiyonlar de bu özellikleri sağlayacak şekilde seçilmiştir.

Geliştirilen jenerik TIOA modeli ve yukarıda bahsedilen şekilde özelleştirilmış katman tanımları detaylı olarak 2. Dönem Raporunda ve proje çıktısı olan makalede [SCHMIDT (2012)] detaylı olarak anlatılmıştır.

Bu modelin proje ilerledikçe elde edilen sonuçlara göre güncellenmiş hali Bölüm 3.9'de ve Güvenirlik Katmanı eklenmiş son sürümü Bölüm 3.8'de sunulmaktadır.

3.3 D^{2}RIP Protokol Yığıtı ve Eşzamanlama Protokolu Gerçeklenmesi

Projede öncelikle D^{2}RIP Protokol Yığıtı TIOA modelinin ilk sürümüne göre (Bölüm 3.2) gerçekleme yapılmıştır. Bu gerçeklemeler kişisel bilgisayar, gömülü geliştirme bilgisayarı ve programlanabili mantık kontrolcusu (PLC) üzerinde çalıştırılmış ve üç düğümlü görece basit bir örnek üzerinde testleri yapılmıştır.

3.3.1 D^{2}RIP Protokol Yığıtı Donanım Gerçeklemesi, Eşzamanlama Protokolu Yazılım Gerçeklemesi

Bu gerçekleme ve başarım testleri 3. ve 4. Dönem Raporlarında ve [GOZCU et al. (2012)] bildirisinde detaylı olarak anlatılmaktadır.

D^2RIP mimarisinin bu sürümü gerçek zamanlı Linux çekirdeği 2.6.33.7 sürümü üzerinde gerçekleştirilmiştir. Gerçek zaman yaması uygulanmış, HPET zamanlayıcısı ile birlikte yüksek çözünürlüklü zamanlayıcılar ve tam önboşaltma (complete preemption) etkinleştirilerek gerçek zamanlı çalışma sağlanmıştır. Linux modüler yapıda olduğu için sürücüler üzerindeki değişiklikler ana çekirdeği etkilemeden yüklenebilir modüller olarak yapılabilmektedir.

Arayüz Katmanı (AK) paketlerin fiziksel kanala gönderilmesi/alınmasından sorumlu olduğu AK gerçekleştirimi sürücü işlevlerininde değişiklikler gerektirmektedir. Bu nedenle AK Linux çekirdek alanında (kernel space), Koordinasyon Katmanı (KK) ise kullanıcı alanında (user space) gerçekleştirilmıştir.

Zamanlama İşlevleri ve IEEE1588 ile Eşzamanlama

Eşzamanlama ve gerçek zamanlı başarım için her düğüm kendi zamanını hassas olarak bilmek zorundadır. AK için çekirdek alanından ulaşılabilecek Linux 2.6 çekirdeğinin POSIX zaman tiplerinden istenen değere ayarlanabilen CLOCK_REALTIME seçilmiştir. Bİr işlevi istenen zamanda gerçekleştirebilmek için *zamanlayıcı kesmesi* (timer interrupt) yöntemi seçilmiştir.

Kullanıcı alanında gerçekleştirilen IEEE 1588 eşzamanlama protokolu için HPET (High Precision Event Timer) saat kaynağı kullanılmıştır. HPET bir osilatör ile 1ns'ye kadar hassasiyette saat sağlayan 10MHz, 64bit bir sayaçtır. Osilatörden kaynaklanan kaymalar olabilmektedir. Kullanıcı alanından değerini okumak 1ns civarında zaman almaktadır. HPET'e Linux 2.6 çekirdeğinde POSIX arayüzü ile erişilebilir.

Çalışmamızda IEEE 1588 protokolunun *Announce, Sync, Follow Up, Delay Request, Delay Response* mesajları ile saat ayarlaması için bir saat servo modülü gerçekleştirilmıştir [CORRELL (2006)]. Hassasiyet için gönderilen eşzamanlama mesajlarının zaman damgalarının olabildiğince donanıma yakın alınması gereklidir. gerçekleştirimimizde zaman damgaları ağ arayüz kartı sürücüsünde alınmaktadır. Eşzamanlama paketleri gerçek zamanlı olmayan uygulamadan gönderildikleri için gerçek zamanlı paketler nedeni ile AK tarafından geciktirilebilirler. Bu gecikme zamanı çekirdek alanındaki AK tarafından ağ arayüz kartı sürücüsüne `ioctl` işlevi ile yollanılarak eşzamanlama paketlerinin zaman damgaları düzeltilmektedir. Eşzamanlama paketleri gerçek zamanlı olmayan diğer paketlere göre yüksek öncelikle gönderilmektedirler. Bu amaçla AK tarafından gerçek zamanlı olmayan paketler için yüksek öncelikli ve normal paketler için iki ayrı kuyruk tutulmaktadır.

Zaman dilimleri arasında eşzamanlamada hassasiyet içinde kalacak şekilde olabilecek farklardan dolayı protokol tarafından kullanılmayan koruyucu periyodlar (guard periods) bulunmaktadır. Bu sebeple eşzamanlama hassasiyeti zaman dilimlerinin boyutlarını doğrudan belirlemektedir.

Yaptığımız deneylerde iki düğüm 10Mbps hub üzerinden eşzamanlama protokolunu 1 saat boyunca çalıştırmışlardır. İki cihaz arasındaki saat farkları -36.69 μs ile 34.31 μs arasında değişmiş, ortalama saat farkı ise 0.25 μs olmuştur. Saat farkının standart sapması 7.15 μs olarak ölçülmüştür.

Koordinasyon Katmanı gerçekleştirimi

KK gerçekleştirimi güvenilir bir uygulama program arayüzü (API) olan POSIX arayüzü ile yapılmıştır. Gerçek zaman kısıtlarından dolayı KK gerçekleştiriminde global değişkenler ve sayfalamadan (paging) kaçınılmış, `memcpy` ve `malloc` kullanımı en azda tutulmuştur. KK işletim sisteminde en yüksek gerçek zamanlı öncelikle koşmaktadır. KK zaman hakkında gerekli bilgileri AK'ndan almaktadır.

Koordinasyon Katmanı-Kontrol Uygulamaları arayüzü:
Kontrol uygulaması verisi ve uygulamanın sağladığı protokol ile ilgili parametreler bir XML dosyası ile KK'na iletilir. Kontrol uygulaması göndereceği gerçek zamanlı mesajı KK'na yazılacağı gönderme arabelleği (transmit buffer) bilgisi ile birlikte bir fonskiyon çağırarak iletir. Yine kontrol uygulaması KK'nını başka düğümlerden mesaj almışsa uygulamaya iletmesi için bir fonksiyonla sürekli kontrol eder. KK mesaj almışsa uygulamaya iletir almamamışsa mesaj arabelleğinin boş olduğunu bildirir. Kontrol uygulaması ve KK arasındakı haberleşme ayrı bir POSIX mesaj kuyruğu üzerinden yapılır.

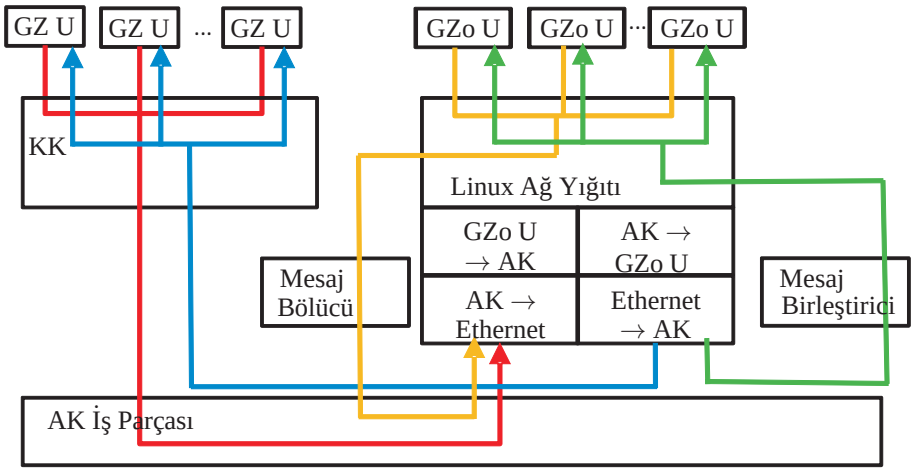
Koordinasyon Katmanı-Arayüz Katmanı arayüzü:
AK, KK'na bir sonraki zaman dilimi hakkında bilgi almak için o andaki zamanı da içeren bir mesaj yollar. Bu mesajı aldığında KK katmanı bir sonraki zaman diliminin gerçek zamanlı olup olmadığı ve bu düğüm tarafından kullanılıp kullanılmayacağı ile ilgili bir hesaplama yapar. Protokolun doğru çalışabilmesi için bu hesabın Şekil 10 (b)'de görülen *KK hesap zamanı* içinde tamamlanması gereklidir. KK, AK'na bu bilgileri ve eğer bir sonraki zaman dilimi bu düğüme aitse yollanacak gerçek zamanlı kontrol uygulaması verisini de içeren bir mesaj yollar. KK başka bir düğümden gelen kontrol uygulaması verisini AK'ndan aldıktan sonra bu veriye uygun olarak kendi durumunu günceller ve yeni durumuna göre bir sonraki zaman dilimi için hesaplama yapar. KK Linux kullanıcı alanında, AK çekirdek alanında gerçekleştirildiği için katmanlar arası haberleşmede POSIX mesaj kuyruğu yerine iki katman tarafından paylaşılan bir dosyaya yazıp okuyan bir karakter cihazı kullanılmıştır.

Arayüz Katmanı gerçekleştirimi
Ethernet sürücüsünün alma ve gönderme fonksiyonları yolu ile AK'nın Linux ağ yığıtına girmeden AK'nın doğrudan Ethernet'e mesajları gönderip alması sağlanmıştır. Buna ek olarak gerçek zamanlı olmayan paketlerin zaman dilimine sığmamaları durumunda bölünmeleri ve tekrar birleştirilmeleri için Ethernet sürücülerinde değişiklik yapılmıştır. Yapılan değişiklikler arayüz kartı sürücü üreticisinden bağımsızdır. AK iş parçası ve genel işlevi Şekil 11'de görülmektedir.

Zaman dilimli yapının eş zamanlı başlatılması:
Zaman dilimli yapının düzgün çalışabilmesi için farklı düğümlerde çalışan Arayüz Katmanlarının aynı anda çalışmaya başlaması gerekmektedir. Bu amaçla bir düğüm geçici olarak ana düğüm seçilir ve özel bir eşzamanlama mesajı (SYNC) ile haberleşme döngüsünü başlatır. Bu eşzamanlama mesajının farklı düğümler tarafından farklı gecikmeler ile alınması nedeni ile düğümler eş zamanlama mesajını aldıktan 10 saniye sonra mesaj göndermektedirler. Eğer sinyal gecikmesinden dolayı SYNC mesajı bazı düğümler tarafından 10 saniyeden fazla gecikme ile alınıyorsa bu düğümler eşzamanlamada 10 saniye geri kalabilirler. Bu sebeple ana düğüm SYNC mesajını 10 saniyelik periyodun ortasında yollar.

Mesaj bölücü ve birleştirici:
Zaman dilimine sığmayan gerçek zamanlı olmayan mesajların taşındığı uygulama paketlerinin bölünüp tekrar birleştirilmesi için iş parçaları (thread) gerçekleştirilmıştir. Bu amaçla D^2RIP için toplam 56 bitlik özel bir çerçeve başlığı yapısı tasarlanmıştır. Toplam 56 bit olan başlık bölümleri: *nodeID*: gönderen düğümün ID numarası (8 bit), *packetID*: uygulama paketi numarası (8 bit), *packetLength*: bölünmemiş paket uzunluğu (16 bit), *frameNum*: paketin kaç çerçeveye bölündüğü (8 bit), *frameSeq*: çok çerçeveye bölünmüş bir paket için bu çerçevenin kaçıncı çerçeve olduğu (8 bit), *frameLength*: çerçeve uzunluğu (8 bit), olarak sıralanabilir.

Ethernet-Arayüz Katmanı Arayüzü:
Ethernet sürücüsünün gönderme ve alma işlevleri ile AK gerçek zamanlı veri taşıyan ya da



Şekil 11: AK çalışması ve iş parçası yapısı.

taşımayan çerçeveleri işletim sistemi ağ yığıtına girmeden doğrudan Ethernet hattına gönderip alabilir. Bu amaçla sürücü kaynak kodları uyarlanmış ve çekirdek ağacının dışında ayrıca derlenmiştir. Sistem başlatıldıktan sonra standart Ethernet sürücü modülü çıkarılıp yerine uyarlanmış modül yerleştirilebilir. Uyarlanmış modül için gerçek zamanlı uygulama, gerçek zamanlı olmayan uygulama ve eşzamanlama paketlerini taşımak üzere IL_RT_PACKET, IL_NRT_PACKET ve IL_SYNC_PACKET olmak üzere üç yeni Ethernet çerçeve tipi tanımlanmıştır.

Ethernet sürücüsünün gönderme işlevinde *mesaj bölücü iş parçası* (fragmenter thread) çalıştırılmadıysa standart fonksiyon kullanılmakta aksi halde uyarlanmış sürüm koşturulmaktadır. Ethernet sürücüsünün alma işlevinde paket *mesaj birleştirici iş parçasından* (reassembler thread) gelmiyorsa doğrudan standart alım işlevi ile yollanmaktadır. Aksi halde yukarıda tanımlanan çerçeve tiplerine göre farklı yollar izlenmektedir.

Arayüz Katmanı - Koordinasyon Katmanı Arayüzü:
AK ve KK arasında gerçek zamanlı paket iletimi *mesajlaşma*, protokoun çalışması için kontrol bilgisi iletimi ise *sinyalleşme* yoluyla yapılmaktadır. Mesajlaşma çekirdek alanından kullanıcı alanındaki bellek alanına hızlı ulaşabilen `memcpy()` işlevi ile gerçekleştirilmiştir. Sinyalleşme için semafor, bekleme kuyruğu (wait queue), yazma, okuma ve bloklu okuma yönetmlerinin gecikmelerinin aynı ölçülerde olduğu görülmüş ve semafor kullanılabileceği kanısına varılmıştır.

3.3.2 D^2RIP Protokol Yığıtı Simülatör Yazılımı

Bu gerçekleştirmelere paralel olarak TIOA modeline uygun bir benzetim yazılımı (simulatör) oluşturulmuş ve gerçekleştirimler ile karşılıklı doğrulanmıştır. Simülatör nesneye yönelim yaklaşımıyla C++ dili kullanılarak yazılmıştır.

Simülatör olay tabanlıdır. Buna göre gerçek sistemde herhangi bir anda aktif olan bir eylem, simulatör tarafından bir olay olarak yaratılmakta, olayın *D^2RIP* Protokol Yığıtında tanımlanan şekilde gerçekleşeceği zaman hesaplanarak simülatör içindeki zamana göre öncelikli olay kuyruğuna (ÖK) yerleştirilmektedir. Sistem olaylar kuyruktan servis edildikçe ilerlemektedir.

ÖK'den çıkartılmış bir olaydan etkilenerek oluşan, o olaydan belirli bir süre sonra ve belirli şartlarda gerçekleşecek olan yeni olay veya olaylar, ÖK'ye itilmektedir. Kimi olaylar esnasında, birden çok olayın yaratılıp, ÖK'ye sokulacağı durumların yanı sıra, hiçbir olay yaratılmayacağı durumlar da vardır. Simülatörde her cihaz için Arayüz ve Koordinasyon katmanlarının özelleştirilmış gerçekleştirimlerinin yanı sıra bütün cihazlar için payaşımlı ortam gerçekleştrimi de bulunmaktadır.

Şurası not edilmelidir ki, D^{2}RIP Protokol Yığını tasarımına göre, her bir zaman aralığında, sistemin durumunu güncellemek amacıyla bir UPDATE olayı yaratılır ve ÖK'ye itilir. Dolayısıyla, ÖK hiçbir zaman boşalmayacaktır.

D^{2}RIP Benzeştiricisi, D^{2}RIP Protokol Yığını ile çalışan gerçek bir sistemi, sistemin anlık durumu ve değişkenlerin değerlerinin gerçek sistem ile aynı olacak şekilde mümkün olduğunca gerçekçi ve birebir bir şekilde benzeştirmeyi amaçlamaktadır. Örneğin, bir paketin gerçek sistemde ve benzeştirim deneyinde aynı anda yaratıldığını, aynı anda servis edildiğini vb. bilgilerine ulaşarak gerçek sistemin analizini çok daha basit ve hızlı bir şekilde yapılabilecektir.

Bu benzetim yazılımının detayları 4. Dönem Raporunda ve simülatörü gerçekleştiren proje bursiyeri Güray Aybar'ın tezinde [AYBAR (2011)] sunulmaktadır.

3.4 D^{2}RIP Başarım İyileştirilmesi

Bölüm 3.9'de belirtildiği gibi Arayüz Katmanı için basitleştirilmiş bir yapıya geçilmiştir. Koordinasyon katmanı için 2. Dönem Raporunda detaylı anlatılan ve işçerçevesine uygun DART gerçekleştirimi tablo bakmadan kaynaklanan hesaplama karmaşıklığından dolayı bu hızlandırılmış gerçekleştirimde kullanılmamıştır.

Bu sebeple Projenin son döneminde yukarıda belirtildiği şekilde basitleştirilmiş IL katmanı ve URT katmanı gerçeklemeleri protokol operasyonunu hızlandırarak iletilen toplam veri hacmini artıracak ve gecikmeleri azaltacak şekilde uyarlanmış ve optimize edilmiştir. Bu çalışmaların sonucunda ortaya proje önerisinde belirtildiği gibi standart donanımla çalışabilen ve taşınabilir bir protokol sürümü çıkmıştır.

Bu bölümde öncelikle Bölüm 3.3'da özetlenen ilk gerçekleştirime göre farklar vurgulanarak katmanlı yapı anlatılacak daha sonra bu iyileştirilmış gerçeklemenin detayları sunulacaktır.

Zaman dilimleri:
İlk TIOA modeline göre [SCHMIDT (2012)] yapılan gerçekleştirimde bir sonraki zaman dilimi için gerekli hesaplar yapılmakta ve zaman dilimi veri transferi ile başlamaktaydı. UPPAAL modelleme ve doğrulama çalışmalarında elde edilen sonuçlara dayanarak, yeni gerçekleştirimdeki zaman dilimi yapısı Şekil 12'de görüldüğü gibi düzenlenmiştir. Her *dSlot* uzunluğundaki zaman dilimi başında önce CL ve IL için gerekli hesaplamalar (CL_{cmp} ve IL_{cmp}) yapılmakta bunu takiben veri iletimi *Frame* zamanda gerçekleşmektedir.

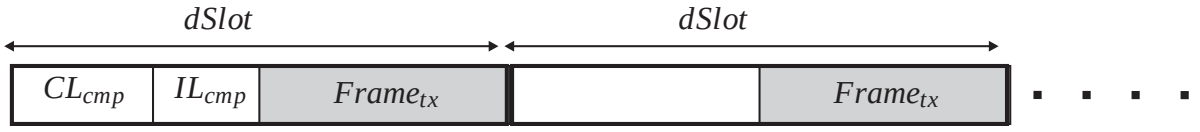
Koordinasyon Katmanı (CL) :
CL_i düğüm i'nin bu zaman dilimi içinde mesaj gönderip göndermeyeceğini belirler. Buna ek olarak güncel zaman diliminin gerçek zamanlı (RT) ya da gerçek zamanlı olmayan (nRT) mesajlar iletimine ayırıldığını belirler.

Kontrol uygulaması modeli tarafından üretilen olaylar ve modele göre o andaki durumdan hesaplanan gelecek zaman dilimlerinde yapılacak olan haberleşmeler için haberleşme istek-

leri (communication request) paylaşımlı ortam üzerinden bütün düğümlere broadcast yoluyla gönderilir. Bunu takiben bu mesajı alan bütün CL_i katmanları tutarlı bir hesap yaparak bir sonraki zaman diliminde gönderilecek mesajın tipini ve gerçek zamanlı mesaj ise gönderecek düğümü belirlerler.

Arayüz Katmanı (IL) :
UPPAAL modelleme ve doğrulama çalışmalarında elde edilen sonuçlara dayanarak, IL katmanının sadece zaman dilimleri yönetimi ile Ethernet'e TDMA ulaşımı sağlaması uygun görülmüştür. Buna göre *koşulsuz olarak* her zaman diliminin başında CL_i, IL_i katmanına zaman dilimi kullanımı ile ilgili bilgi göndermekte ve IL_i da buna uygun olarak sıradaki Ethernet Çerçevesini zaman dilimi içinde yollamaktdır.

Eş zamanlama:
Paylaşımlı ortamda haberleşen düğümler IEEE 1588 eş zamanlama protokolu ile zaman birliğini sağlamaktadırlar. Bu protokolun mesajları öncelikli nRT mesajlar olarak yollanmaktadır. Bu şekilde deterministik kontrol uygulaması modellerinden farklı olan bu protokol model formal işçerçevesinin parçası yapılmamıştır. Dolayısı ile istendiği zaman kolayca başka bir eşzamanlama protokolu ile değiştirilebilir. Ayrıca gerçek zamanlı mesajlardan farklı olarak eşzamanlama mesajlarının belli sınırlar içinde gecikmesi protokol işleyişini bozmamaktadır.



Şekil 12: Zaman dilimli çalışma.

3.4.1 Koordinasyon Katmanı (KK-CL) Veri yapısı

CL kontrol uygulamasından mesajları iletmek üzer alır. Her mesaj *m* kontrol uygulaması verisi *m*.data ve haberleşme istekleri *m*.req içerir. Kontrol uygulaması verisi Bölüm 3.6'de anlatıldığı gibi sorgular, bildirimler ve komutlardan oluşmaktadır. Haberleşme istekleri her düğüm *i* üzerinde koşan bir arayüz modülü tarafından uygulama otomat modelinin durumuna göre bir tablodan bakılarak hesaplanmaktadır. Bir düğüm *i*'den farklı kontrol uygulamalarına ilişkin veriler gönderilebilir. Bu sebeple transport katmanı portlarına benzer şekilde her mesaj gönderici düğüm *i* ve gönderen *kanal-channel* ile tanımlanır.

Bu bölümde zaman birimi olarak *dSlot* kullanılmaktadır ve bu birimden güncel zaman *t* ile gösterilir.

Haberleşme istekleri ve öncelikli kuyruk (priority queue) yapısı:

- Haberleşme isteği (communication request) (b, c, eT, dT) olarak gösterilir. Bu gösterimde; *b* mesaj gönderecek cihazı, *c* mesajın ait olduğu (transport katmanı portu benzeri) kanal numarasını, *eT* (time) mesajın geçerli olacağı zamanı *dT* ıse don gönderilme zamanını (deadline) göstermektedir. Bir haberleşme isteği gönderildiği mutlak *t* zamanından *eT* zaman dilimi sonra geçerli olmaktadır. Bu geçerlilik zamanı kontrol sistemlerinde kullanılan programalanabilir mantık kontrolcularının (PLC) çevirimsel (cyclic) çalşma prensibine göre tanımlanmıştır. Bir PLC hesaplarını çevirim içerisinde

tamamlanmadan haberleşmeyecek ancak eT zaman dilimi sonra haberleşecektir. Buna göre o cihaza bu zamandan önce haberleşme hakkı tanımak anlamlı değildir. Buna göre eT ve dT parametreleri kontrol uygulaması ve cihazlarına göre belirlenecektir.

- $\texttt{PQ}_i$: Haberleşme isteklerini (b,c,eT,dT) şeklinde tutan öncelikli kuyruk.
- $\texttt{PQ}_i$.Top: En küçük son gönderilme zamanına sahip ve geçerli olan ($eT \geq t$) haberleşme isteğinin öncelikli kuyraktaki pozisyonu.

Boolean Değişkenler:
Zaman dilimlerinin nasıl kullanılacağını kodlarlar. $\texttt{PQ}_i$.Top pozisyonundaki haberleşme isteğine göre belirlenir. Bu değişlenler her düğüm i için tutarlı olarak hesaplanır.

- $\texttt{RTCL}_i$: Güncel zaman dilimde göderilecek mesaj gerçek zamanlı ise true, değilse false değerini alır. Bütün cihazlarda bu değişkenin değeri tutarlı olarak aynı olmalıdır.
- $\texttt{myCL}_i$: Eğer güncel zaman diliminde cihaz i gerçek zamanlı mesaj gönderecekse true değilse false değerini alır. Herhangi bir zaman dilimi için bu değişken sadece tek bir cihazda true diğer bütün cihazlarda false olmalıdır. Bu şekilde çarpışmadan kaçınma davranışı sağlanmaktadır.

Mesaj arabellekleri:
Uygulama ve Arayüz katlanlarından alınan mesajlar bu arabelleklerde tutulur.

- $\texttt{Tx}_i$: Kontrol uygulamasından gelen ve paylaşımlı ortamda gönderilmek üzere KK'na iletilen gerçek zamanlı mesajları tutar. Bu mesajların farklı uygulamalardan gelmeleri durumunda kanal numarasıyla birbirlerinden ayırtedilir. Transport katmanı portlarına benzer şekilde her kanal b cihaz numarası ve c kanal numarasıyla tanımlanır. $\texttt{Tx}_i$ bir arabellek vektörüdür ve her arabellek bir kanala ait mesajları tutar.
- $\texttt{Rx}_i$: Arayüz katmanından gelen mesajarı FIFO kuyruk olarak tutar.

Uygulama Katmanı mesaj arayüzü:
Bütün mesaj transferlei uygulama katmanı tarafınan başlatılmaktadır.

- $\texttt{AP2CL}(data,req,ch)_i$: Kontrol uygulamasından gerçek zamanlı mesaj alımı . $\texttt{Tx}_i[ch]$deki mesaj veri ($data$) ve haberleşme isteklerinde oluşur (Şekil 13).
- $\texttt{CL2AP}(\texttt{Rx}_i)_i$: $(\texttt{Rx}_i)_i$'de tutulan mesaj mi çıkarıp m.data bölümünü uygulamaya iletir. Bu işlemin ardından PQ_i aşağıdaki gibi güncellenir:
  - Eğer $\texttt{RTCL}_i = \text{true}$ ve m.data boş değil ise geçerli bir gerçek zamanlı mesaj alınmıştır. $\texttt{PQ}_i$.Top pozisyonundaki haberleşme isteği kuyruktan çıkarılır.
  - Eğer $\texttt{RTCL}_i = \text{true}$ ve m.data boş ise kontrol uygulaması hala bir algılayıcı verisi ya da eyleyici komutu için beklemektedir. Dolayısı ile ilgili kontrol aksiyonu için haberleşmesi gereken mesajı daha yaratmamıştır. Bu durumda $\texttt{PQ}_i$.Top pozisyonundaki (b,c,eT,dT) haberleşme isteği kuyruktam çıkarılır ve cihaz b yeniden haberleşme için çizelgelenir. Bu amaçla haberleşme isteği mutlak zaman t ile $(b,c,eT+t,dT+t)$ olarak güncellenir ve tekrar PQ_iye yerleştirilir.

Şekil 13: Kontrol uygulaması mesajları.

Dağıtık hesaplama ile karar değişkenlerinin güncellenmesi:
Haberleşme istekleri *m*.req uygulama katmanından gelen her mesaja eklenmektedir. Mesaj paylaşımlı ortama gönderildikten sonra ortamın tüme gönderim (broadcast) özelliği sayesinde bütün cihazlar tarafından alınmaktadır. Mesaj alındıktan sonra haberleşme istekleri bütün cihazlar için PQ_i içine geçerlilik durumlarına göre ve son gönderme zamanlarına göre sıralı olarak yerleştirilmektedir. Bu şekilde bütün cihazlarda $\mathtt{PQ}_i$.Top pozisyonundaki haberleşme isteği aynı olmakta ve en acil haberleşmesi gerekn cihazı göstermektedir. Bütün düğümlerde utarlı olarak aynı olan bu bilgiye göre bu istekteki *b* cihazı ortama erişim hakkını almaktadır.

Her zaman dilimi başında her düğümdeki KK (CL) g_{RT} and g_{my} fonksiyonlarını çalıştırarak durumunu günceller. Buna göre:

- $g_{\mathrm{RT}}(\mathtt{PQ}, \mathtt{RTCL}_i)$: Bu zaman diliminin herçek zamanlı trafik için ayrılıp ayrılmadığını gösteren $\mathtt{RTCL}_i$ değişkeninin değerini belirler. Bu fonksiyon bütün cihazlarda aynı deperi verir.
  - $\mathtt{RTCL}_i = \text{true}$ if $\mathtt{PQ}.\text{Top}.eT \leq t$
  - $\mathtt{RTCL}_i = \text{false}$ otherwise
- $g_{\mathrm{my}}(\mathtt{PQ}, \mathtt{RTCL}_i)$: Bir düğüm *i* için bu zaman diliminin sahibini olup olmadığını belirleyen $\mathtt{myCL}_i$ ve eğer öyleyse hangi kanaldaki uygulamanın verisinin gönderileceğini gösteren $\mathtt{ch}_i$ değişkenlerinin değerlerini belirler. Sadece tek bir cihaz için $\mathtt{myCL}_i = \text{true}$ doğrudur.
  - $(\mathtt{myCL}_i, \mathtt{ch}_i) = (\text{true}, a)$ if $\mathtt{PQ}_i.\text{Top}.b = i \wedge \mathtt{RTCL}_i = \text{true} \wedge \mathtt{PQ}_i.\text{Top}.c = a$.
  - $(\mathtt{myCL}_i, \mathtt{ch}_i) = (\text{false}, 0)$ değilse

3.4.2 Arayüz Katmanı (AK-IL) Veri Yapısı

AK paylaşımlı ortam üzerinde zaman dilimi ortama erişim sağlamakla sorumludur. KKya benzer şekilde her düğüm *i* üzerinde çalışan AK bileşeni IL_i olarak gösterilmektedir. Bu IL_i, CL_iden aldıkları $\mathtt{RTCL}_i$ ve $\mathtt{myCL}_i$ değerlerine göre güncel zaman diliminde tutarlı olarak ortama erişimi düzenleyerek çarpılmadan kaçınmayı sağlamaktadırlar. Eğer KK hesaplaması bu zaman diliminin gerçek zamanlı olmayan trafik için kullanılmasına karar vermişsse bütün düğümlerdeki AK birimleri tutarlı ve çevrimsel tekrar eden bir çizelgelemeyle (sıralı gönderme ya da ağrlıklı sıralı gönderme gibi) hangi düğümün gerçek zamanlı olmayan mesaj yollayacağına karar vermektedirler.

Mesaj arabellekleri:
Üst ve alt katmanlardan gelen mesajları depolar.

- $\texttt{TxRT}_i$: Paylaşımlı ortama gönderilecek gerçek zamanlı mesajları tutar.
- $\texttt{TxnRT}_i$: Paylaşımlı ortama gönderilecek gerçek zamanlı olmayan mesajları tutar. $\texttt{TxnRT}_i$.Top ilk gönderilecek mesajın pozisyonudur. IEEE 1588 mesajları diğer gerçek zamanlı olmayan mesajlara göre yüksek öncelikle gönderilmektedir.
- RxRT: Paylaşımlı ortamdan alınacak gerçek zamanlı mesajları tutar.
- $\texttt{RxnRT}_i$: Paylaşımlı ortamdan alınacak gerçek zamanlı olmayan mesajları tutar.

Boolean Değişkenler:
Zaman dilimlerinin sahiplğini gösterir.

- $\texttt{RTIL}_i$: true ise bu zaman diliminde gönderilecek mesaj gerçek zamanlı olacaktır. Böyle değilse false olacaktır.
- $\texttt{myIL}_i$: true ise bu zaman diliminde cihaz i gönderecektir. Değilse false olacaktır.

Protokol durum değişkeni:
$\texttt{vIL}_i$ $\texttt{vIL}_i$ cihaz i için gerçek zamanlı olmayan zaman dilimlerinde gönderecek cihazların çevirimsel olarak belirlenmesini sağlar. Buna göre $\texttt{vIL}_i$ özellikleri:

- $\texttt{vIL}_i$.cnt gerçek zamanlı olmayan zaman dilimlerini çevrimsel (cyclic) olarak sayar.
- Gerçek zamanlı olmayan trafik için düğümlere zaman dilimi ayırımı $\texttt{vIL}_i$.cyc dilimden sonra tekrar eder.
- $\texttt{vIL}_i$.nRTSet düğüm i üzerindeki IL_i tarafından gerçek zamanlı olmayan mesajlar için kullanılabilecek zaman dilimleri kümesini tanımlarç Burada; $\texttt{vIL}_i.\texttt{nRTSet} \cap \texttt{vIL}_j.\texttt{nRTSet}\, j = \emptyset$ for $i, j \in I$, $i \neq j$ olmalıdır. I düğüm kümesini göstermektedir.

Diğer katmanlarla mesaj arayüzü:

- $\text{IL2SM}(m)_i$: Mesaj m'i paylaşımlı ortama gönderir. Bu fonksiyon her zaman dilimi başında eğer $(\texttt{myIL}_i = \text{true})$ doğruysa ve aşağıdakilerden birisi doğruysa çağırılır:
  - $\texttt{RTIL} \wedge \neg(\texttt{TxRT}_i \text{ empty})$: Zaman dilimi gerçek zamanlı trafik için ayırılmış ve hazır bir gerçek zamanlı mesaj var.
  - $\neg\texttt{RTIL}_i \wedge \neg(\texttt{TxnRT}_i.\text{Top empty})$: Zaman dilimi gerçek zamanlı olmayan trafik için ayırılmış ve hazır bir gerçek zamanlı olmayan mesaj var.

  $\text{IL2SM}(m)_i$ çalıştırıldığında $\texttt{TxRT}_i$ ya da $\texttt{TxnRT}_i$'dan uygun olanından gönderilen mesaj çıkartılır.

- SM2IL(m): IL paylaşımlı ortamdan gönderilen mnesaj m'i alır. Eğer zaman dilimi RT ise m RxRT'd edeğilse $\texttt{RxnRT}_i$'da saklanır.

- AP2ILNRT$(m)_i$: Uygulama katmanı AK'na mesaj yollar. Şekil 10'de görüldüğü gibi gerçek zamanlı olmayan uygulamalar mesajlarını AK'na KK'ndan geçmeden doğtrudan yollamaktadırlar.

Karar değişkenlerinin güncellenmesi:
Bütün düğümlerdeki AKler protokol durum değişkeni $\texttt{vIL}_i$ değerine göre gerçek zamanlı olmayan trafik için zaman dilimlerinin ayrılmasını yaparlar. Bu amaçla UPDATE_i fonksiyonu Şekil 12'de görüldüğü gibi her zaman diliminde IL_{cmp} bittikten sonra $\texttt{vIL}_i := f_{\text{upd}}(\texttt{vIL}_i, \texttt{RTIL}_i)$ hesabını yapmak için çağırılır. Buna göre:

$$f_{\text{upd}}(\texttt{vIL}_i, \texttt{RTIL}_i).\texttt{cnt} = \begin{cases} \texttt{vIL}_i.\texttt{cnt} & \text{if } \texttt{RTIL}_i \\ (\texttt{vIL}_i.\texttt{cnt} + 1) & \\ \text{mod } \texttt{vIL}_i.\texttt{cyc} & \text{değilse} \end{cases}$$

Koordinasyon ve Arayüz Katmanları arayüzü
AK ve KK arasında mesaj değişimi iki fonksiyon ile yapılmaktadır:

- IL2CLRT$(m)_i$: AK, RxRT arabelleğinde duran gerçek zamanlı mesaj m'i KKya iletir. Bu fonksiyon AK tarafından, AK hesap zamanı IL_{cmp} geçtikten sonra çağırılır (Şekil 12).

- CL2ILRT$_i(\texttt{RTCL}_i, \texttt{myCL}_i, m)$: KK hesaplama zamanı CL_{cmp} geçtikten sonra KK AK'nına bu zaman diliminin gerçek zamnalı trafik ya da gerçek zamanlı olmayan trafik için ayrıldığını ve gerçek zamanlı trafik ise mesaj gönderecek düğümü bu fonksiyonla iletir. Eğer hazır bir gerçek zamanlı mesaj verisi varsa bu fonksiyonla o da m içinde iletilir.

  KKnından alınan bilgilere göre AK kendi durum değişkenlerini günceller. Bun agöre doğrudan $\texttt{RTIL}_i = \texttt{RTCL}_i$ uygulanır. Gerçek zamanlı trafik için AKnında zaman diliminin sahibi de doğrudan $\texttt{myCL}_i$ ile belirlenir. Gerçek zamanlı olmayan trafik için düğüm i üzerindeki AK çevrimsel olarak sayılan zaman dilimlerinin $\texttt{vIL}_i.\texttt{nRTSet}$ içinde olup olmadığını kontrol eder. Buna göre zaman diliminin sahipliği aşağıdaki gibş belirlenir:

$$\texttt{myIL}_i = f_{\text{my}}(\texttt{vIL}_i, \texttt{RTIL}_i, \texttt{myCL}_i, i) \;=\; \begin{cases} \texttt{myCL}_i & \text{if } \texttt{RTIL}_i = \text{true} \\ \text{true} & \text{if } \neg\texttt{RTIL}_i \wedge \texttt{vIL}_i.\texttt{cnt} \\ & \in \texttt{vIL}_i.\texttt{nRTSet} \\ \text{false} & \text{değilse.} \end{cases}$$

3.4.3 Gerçekleştirim Ortamı

D^{2}RIP hem PCde PLC gibi düşük bellek ve işlemci gücüne sahip gömülü kontrolcu cihazlarında koşmak üzere gerçekleştirilmıştir. Buna uygun olarak Lubuntu işletim sistemi, hızlı ve hafif olması nedeniyle düşük donanım gereksinimi olması nedeni ile seçilmiştir. Lubuntu çekirdeği Linux ve Ubuntu tabanlıdır [LUB (2011)].

D^2RIP gerçek zamanlı bir işletim sistemi üzerinde çalışmalıdır. Gerçek zamanlı bir işletim sistemi çekirdeği standart Linux çekirdeğinden gerçek zamanlı yamalar ile elde edilir (real-time patch). Uygun gerçek zamanlı yamaların geliştirildiği en son kararlı çekirdek sürümü 3.6.2dir. gerçek zamanlı yamayla Linux çekirdeğini tekrar derlemeden önce aşağıdaki çekirdek konfigürasyon dosyasına aşağıdaki değişikliklerin yapılması gerkelidir:

- ”Tickless System (Dynamics Ticks)” etkinleştirilmesi
- ”High Resolution Timer Support” etkinleştirilmesi
- ”Preemption Model” parametresinin ”Fully Preemptible Kernel(RT)” yapılması
- ”Timer frequency” parametresinin ”1000Hz” yapılması
- ”Suspend to RAM and standby” etkisizleştirilmesi
- ”Timestamping in PHY devices” etkinleştirilmesi
- ”PTP Hardware Clock (PHC)” etkinleştirilmesi
- ”PTP clock support” etkinleştirilmesi
- ”Show timing information on printks” etkisizleştirilmesi
- ”I/O scheduler” parametresinin ”Deadline” yapılması

Proje önerisinde amaçlanan standart gerçekleştirime ulaşılmıştır. D^2RIP tamamen taşınabilir olarak tek bir USB hafıza çubuğu üzerinde dağıtılabilecek ve bu çubukla kurulabilecek şekilde gerçekleştirilmıştir. Buna ulaşabilmek için Lubuntu üzerinde bazı ayarlar yapılarak verim artırılması sağlanmıştır. Bu ayarlar:

- Dosya erişiminde dosya metaverisinin yazılmasının etkisizleştirilmesi
- Bütün logların RAMe yazılması
- Sayfaların hafızaya yazılıp kaldırılmasının etkisizleştirilmesi (swappiness)
- Gerekli olmayan servislerin, uygulamaların sürücülerin ve görsel efektlerin kaldırılması
- TCP ve UDP soket giriş çıkış kuyrukları için ayrılan bellek miktarının yükseltilmesi ve düşük gecikme ve zaman damgası ayarların etkinleştirilmesi
- İşletim sistemi tarafından etkisizleştirilemeyen özel maskelenmeyen kesmeler kullanan BIOS ayarlarının değiştirilmesi. Bu kesmeler gerçek zamanlı çalışma sırasında kabul edielemeyecek gecikme ve gecikme değişimlerine yol açmaktadırlar.
  - Serial, Parallel Port: Disable (Interruptları azaltacak)
  - Audio: Disable (Interruptları azaltacak)
  - Event Log: Disable (Diske yazmayı azaltacak)
  - Fan Control and Real Time Monitoring: Disable (Interruptları azaltacak)

  - PCI Latency Timer: 32 (Ethernet kartının interruptlarına hızlı cevap verecek. En düşük değer bu)
  - CPU Idle State: High Performance (CPU Power Save modunu kapatıyor)
  - CPU C State: Disable (CPU Power Save modunu kapatıyor)
  - C2 State: Disable (CPU Power Save modunu kapatıyor)

- Ağ arayüz kartının NAPI (Rx polling mode) özelliğinin etkinleştirilmesi. NAPI özelliği gelen mesajların daha az kesme ile daha verimli olarak işlenmesini sağlamaktadır.
- ”InterruptThrottleRate” parametresinin 1 yapılarak Ethernet sürücüsünün gelen trafiğe göre kesme oluşması hızını dinamik olarak ayarlamasının sağlanması
- ”TxIntDelay” parametresinin 0 yapılması ile gerçek zamanlı paketlerin hemen gönderilmeye başlamasının sağlanması.
- Linux’da yer alan etc/rc.local (Linux boot ettiği zaman bunu çalıştırıyor) scriptine:

  ```
  echo 2048 >/sys/class/rtc/rtc0/max_user_freq
  echo 2048 >/proc/sys/dev/hpet/max_user_freq
  echo 'hpet'
  >/sys/devices/system/clocksource/clocksource0/current_clocksource
  ```

  ayarları eklendi. Böylece clock source HPET seçildi ve max freq ayarı 2048 oldu.

3.4.4 Paketlerin İçeriği ve Veri Yapıları

Şekil 10’da görülen D^{2}RIP katman yapısına uygun olarak verilerin her katmanda paketlenmesi Şekil 14’de görüldüğü şekilde yapılmaktadır. Şekil 12’deki sabit *dSlot* uzunluktaki zaman dilimi yapısı AK tarafından paylaşımlı ortamda gönderilecek maksimum çerçeve uzunluğuna bir sınır koymaktadır. Bu sınır $Frame_{max}$ olarak gösterilmektedir.

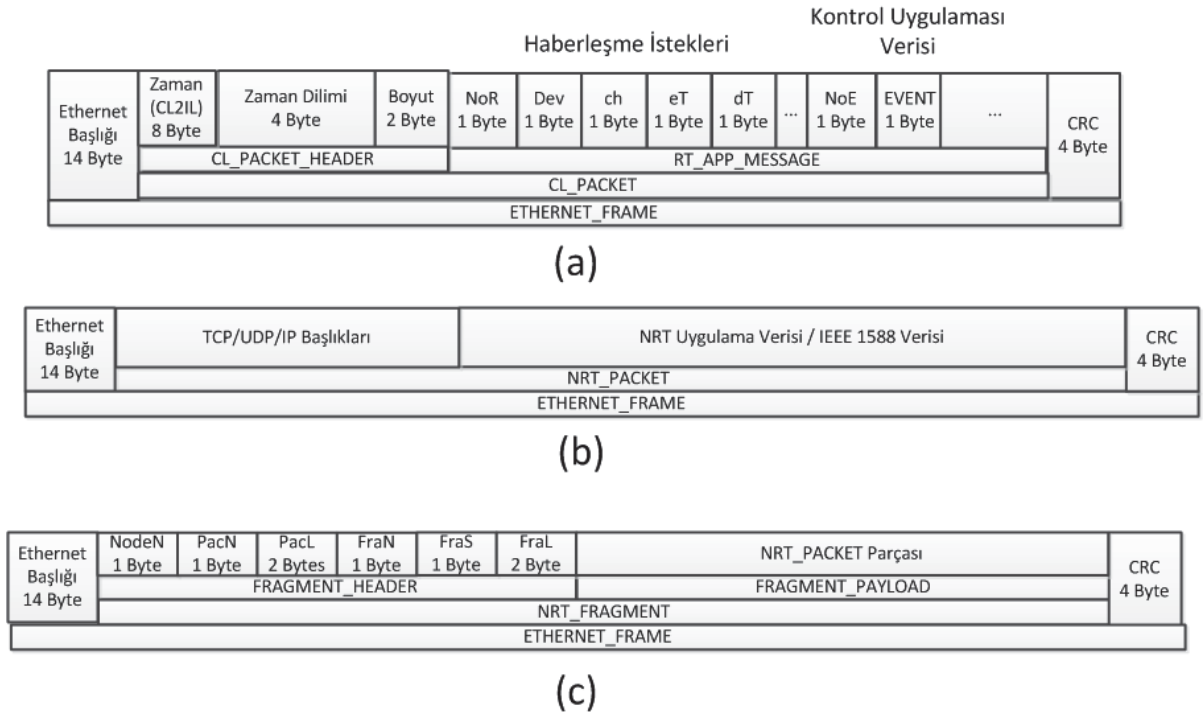
Bölüm 3.5’da tanımlanan (3) ifadesine göre *dSlot* parametresi, D^{2}RIP tarafından garantilenen mesaj son gönerilme zamanlarını belirlemektedir. Buna göre Bölüm 3.4.3’da listelenen işletim sistemi ayarları ve Bölüm 3.4.5’de anlatılan IEEE 1588 eşzamanlaması olabildiğince kısa bir *dSlot* elde edilecek şekilde ayarlanmıştır.

Yapılan deneylerde elde edilebilen $dSlot = 250\mu s$ değeri için $Frame_{max} = 504$ Byte seçilmiştir. Bu değere standart Ethernet çerçevelerinde bulunan 14 Bytes Ethernet Başlığı ve 4 Bytes of CRC dahildir. RT paketlerde Ethernet çerçevesi hep $Frame_{max}$ uzunluğunda gönderilip RT verinin arkası doldurulmaktadır (padding).

Ethernet çerçevelerindeki `EtherType` alanı çerçevede taşınan verinin tipini belirlemek için kullanılmaktadır. Bu alanın değeri RT veriler için `0x1100`, $Frame_{max}$ uzunluğunu geçen nRT paket parçaları için `0x2200` olarak belirlenmiştir. Bu alandaki başka bir değer taşınan verinin kısa, parçalanmamış NRT paketi olduğunu gösterir.

Ethernet çerçevesi taşıdığı veri tipine göre `CL_PACKET` (Şekil 14 (a)), `nRT_PACKET` (kısa, parçalanmamış nRT paket) (Şekil 14 (b)) ya da uzun olan bir nRT paket parçası `nRT_FRAGMENT` olabilir (Şekil 14 (c)).

Şekil 14 (a)'da görünen CL_PACKET, CL_PACKET_HEADER başlığından ve gerçek zamanlı veriden oluşmaktadır. CL_PACKET_HEADER başlığındaki 8 Byte zaman ölümü protokol başlatılmasında (Bölüm 3.4.6) kullanılmaktadır. 4 Byte zaman dlimi sayısı ve CL_PACKET uzunluğunun tamamını gösteren 2 Byte Paket uzunluğu başlığın diğer alanlarıdır.



Şekil 14: Veri paketlenmesi.

CL_PACKET verisi RT uygulama mesajı olan RT_APP_MESSAGE'dır. Bu mesaj eğer güncel zaman diliminde gönderme hakkı olan düğümün mesajı henüz hazır değilse boş olabilir. RT_APP_MESSAGE uzunluğu $Frame_{max} - ((14+8+4)+(4+2))$ Byte ile sınırlıdır.

RT mesaj yapısı Bölüm 3.4.1'da tartışılmış ve Şekil 13'de gösterilmiştir. Buna göre *dSlot* ve $Frame_{max}$ değerlerinin RT mesajların parçalmaya gerek olmayacak şekilde seçilmesi gerekmektedir. Bu varsayım geliştirdiğ—i-miz protokol yapısının genelliğine etki etmemektedir. Bölüm 3.6'deki örnek uygulamada da görüldüğü gibi uygulamanın deterministik olmasından dolayı gönderilebilecek bütün RT mesajların uzunluğu önceden bellidir.

Bölüm 4.1'da detaylı olarak tartışıldığı üzere, deneylerimizde 3.6'deki örnek incelenmektedir ve bu örnek için $Frame_{max} = 504$ Byte seçilmiştir.

3.4.5 IEEE 1588 Eşzamanlaması

Protokol gerçekleştiriminin önceki aşamalarında yazılım ile yapılan IEEE 1588 gerçeklemesi iyileştirmeler yapılırken (Bölüm 3.3) teknolojinin de artık elvermesi ile donanım desteki olarak ve yüksek hassasiyetle yapılmıştır.

Donanım destekli zaman damgalaması için donanım ve yazılım gerekleri aşağıda sıralanmıştır.

- PTP donanım saat destekli bir ağ arayüz kartı (Network Interface Card-NIC) ve Ethernet sürücüsü: Yüksek bulunabilirlikte ve düşük maliyette olması nednei ile Intel Gigabit Ct Desktop Adapter [INT (2012)] seçilmiştir. Bu kart anakartta PCI express slot üzerine takılabilmekte ve başka donanım gerektirmemektedir. Kart Ocak 2013de yayınlanmış-olan 2.2.14 sürümlü `e1000` sürücü modülünü kullanmaktadır. Bu modül derlenirken IEEE 1588 desteğinin etkinleştirilmesi gereklidir.

- Uygun Linux çekirdek sürümü: Donanım zaman damgalaması (PTP) desteği için *ağ paketi zaman damgalaması* ve *saat kontrolu* özellikleri gereklidir. Linux Kernel sürümleri 2.6.30 ve üstünde çekirdek alanında ki bir sürücü modülünü kullanıcı alanındaki IEEE 1588 programına zaman damgası değerlerini yollayabilmek için `SO_TIMESTAMPING` seçeneği bulunmaktadır. Bu şekilde ağ paketi zaman damgalaması özelliği sağlanmaktadır. Saat kontrolu gereği çekirdek sürümleri 3.0 ve üzerinde bulunan ve bir API ile kullanıcı alanından çekirdek alanındaki saatin idaresini sağlayan PTP donanım saat desteği (PTP Hardware Clock (PHC)) ile sağlanmaktadır. Ayrıca Bölüm 3.4.3'de listelenmiş konfigürsayonlar da yapılmış olmalıdır.

IEEE 1588 yazılımı için [PTP (2011)] kaynağından faydalanılmıştır. Bu yazılım IEEE 1588-2008ı ile tam uyumludur ve yüksek hassasiyet için donanım zaman damgalaması ve PTP donanım saat özelliklerini desteklemektedir. Lınux'da standard olan `SO_TIMESTAMPING` soket özelliğini kullanmaktadır. Bu yazılım :

- Eşzamanlamayı sağlamak için düğümler arasında gerekli gecikme ve zaman farklılığı (offset) bilgisi alışverişini sağlayacak IEEE 1588 paketlerinin gönderilmesi

- PTP donanım saat ayarları

- (`REALTIME_CLOCK`) ve PTP Donanım saat eşzamanlaması

Kullanıdğımız IEEE 1588 yazılımı bir PI kontrolcu ile saat frekansını ayarlamakta saatte ayrık atlamalar yapmamaktadır. Bu donanım desteğinin olmadığı durumlarda kullanılan Ethernet sürücüsünün gönderme fonksyonundan önce `skb_tx_timestamp()` fonksiyonu ekleyerek yazılım zaman damgası desteği vermek mümkündür.

3.4.6 D^{2}RIP Çalışmaya Başlatılması

AK D^{2}RIP katman yığıtınınçalışmakta olduğunu Boolean bir değişken olan `D2RIP_Init` değişkeninin `true` değeri ile tutmaktadır. AK Zaman Dilimi Sayısı (Slot Number) 1 olan bir `CL_PACKET` aldığında `D2RIP_Init` değişkeni `true` yapılır ve Ethernet Sürücüsünde yapılan değişiklik etkinleştirilir. Bu değişikliğin detayları 3.4.8 Bölümünde anlatılmaktadır.

Eğer AK KKden 1 Byte `CLP` $=$ `0xFF` verisi alırsa `D2RIP_Init` sıfırlanarak `false` değeri alır ve Ethernet Sürücüsündeki değişiklik etkisizleştirilir.

IEEE 1588 ile eşzamanlama sağlandıktan sonra zaman dilimli yapının bütün düğümlerde eşzamanlı başlatılması aşağıdaki gibi yapılmaktadır:

D^{2}RIP başlamadan `D2RIP_Init` değişkeni `true` yapılırsa AKnın KKnından zaman dilimi sahipliği bilgisini henüz alamaması nedeni ile IEEE 1588 eşzamanlama paketleri bellekte bir

süre takılı kalmaktadır. Bu nedenle protokol D^2RIP_Init değişkeni yolu ile başlatılmadan eşzamanlamanın başlatılması gerekmektedir. Eşzamanlamanın başlatıldığı bilgisi ikinci bir Boolean değişken olan SYNC_Init değişkeninde tutulmaktadır.

Eşzamanlamanın başlatılması için herhangi bir bilgisayardan kullanıcı tanımlı sinyal olan SIGUSR1 yaratılması için killall -s SIGUSR1 CLd komutu çalıştırılmaktadır. Bu komutla KK programına SIGUSR1 sinyali gönderlmektedir. Bu sinyali alan KK güncel zamanı alıp saniyenin 5.katına kadar bekleyip aldığı zamanın saniyenin 10. katını paketin içine koyup yollamaktadır. Örnek: Sinyal 12. Saniyede gelirse KK 15. Saniye ye kadar bekleyip 20. Saniye bilgisini pakete koyup yollayacaktır. Bu paket gerçek zamanlı paket olarak gönderilmektedir. AK tarafından alınan bu paket paylaşımlı ortam Ethernet üzerinden gönderilmekte ve bütün düğümler tarafından alınmaktadır. Paketin alınmasının ardından bütün düğümlerde SYNC_Init değişkeni true olmaktadır. Bu ilk paket içinde zaman dilimi sayısı (Slot Number) 0 olarak gönderilmektedir.

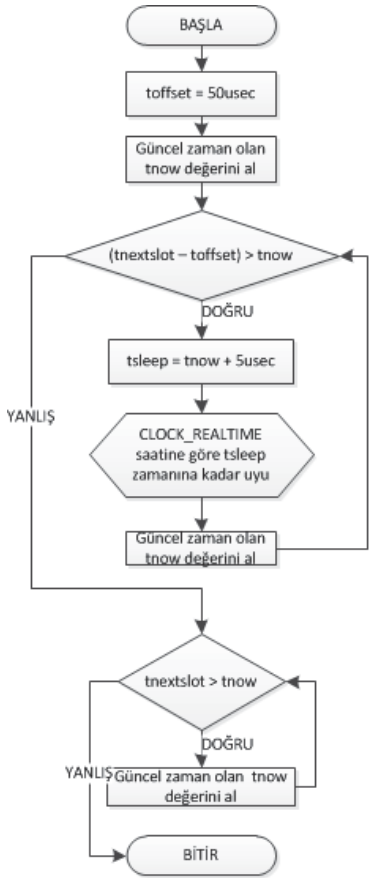
3.4.7 Koordinasyon Katmanı (KK-CL) Gerçeklemesi

Koordinasyon Katmanı, 3.4.1 Bölümünde anlatılan veri yapısına göre kullanıcı alanında bir iş parçacığı (thread) olarak gerçekleştirilmiştir. Bütün düğümlerde koşan CL_i iş parçaçığı periyodik olarak her zaman diliminin başında uyanmasının ardından ilk olarak Bölüm 3.4.1'de tanımlanan AP2CL$(data, req, ch)_i$ fonksiynunu çalıştırır. Eğer uygulamanın gönderecek mesajı varsa bu alınan RT_APP_MESSAGE mesaj $\texttt{Tx}_i$ arabelleğine konur.

Bilindiği gibi iş parçacıkları eğer kullanılmadıkları zaman uyutulmaz, sürekli uyanık tutulurlarsa işlemciyi çok yormaktadırlar. Bu nedenle KK işparçacığı da uyutulmakta ve gerekli olduğu zaman uyandırılmaktadır. Gerçekleştirmenin zaman testlerinde KK işparçacığının uyandırılması sırasında uzun zamanlar geçtiği ve bunun sonucunda sistemin çalışmasının yavaşladığı gözlemlenmiştir. Bu sebeple özelleştirilmiş bir işparçacığı uyandırma mekanizması tasarlanmıştır. Buna göre zaman dilimi boyunca CL işparçacığı periyodik olarak uyandırılarak bu katmanın kulladığı eş zamanlama ve saat değerleri güncellenmektedir. Zaman diliminin son 50 μsnlik bölümünde ise CL işparçacığı uyanık tutularak bir sonraki zaman diliminin başında uyanık olması sağlanmaktadır. Bu basamkalı uyandırmak mekanizmasının iş akış diyagramı Şekil 15'de görülebilir

Bunun ardından CL_i PQ_i haberleşme istekleri öncelikli kuyruğuna göre karar değişkenleri $\texttt{RTCL}_i$ ve $\texttt{myCL}_i$'i günceller. Bir sonraki adımda:

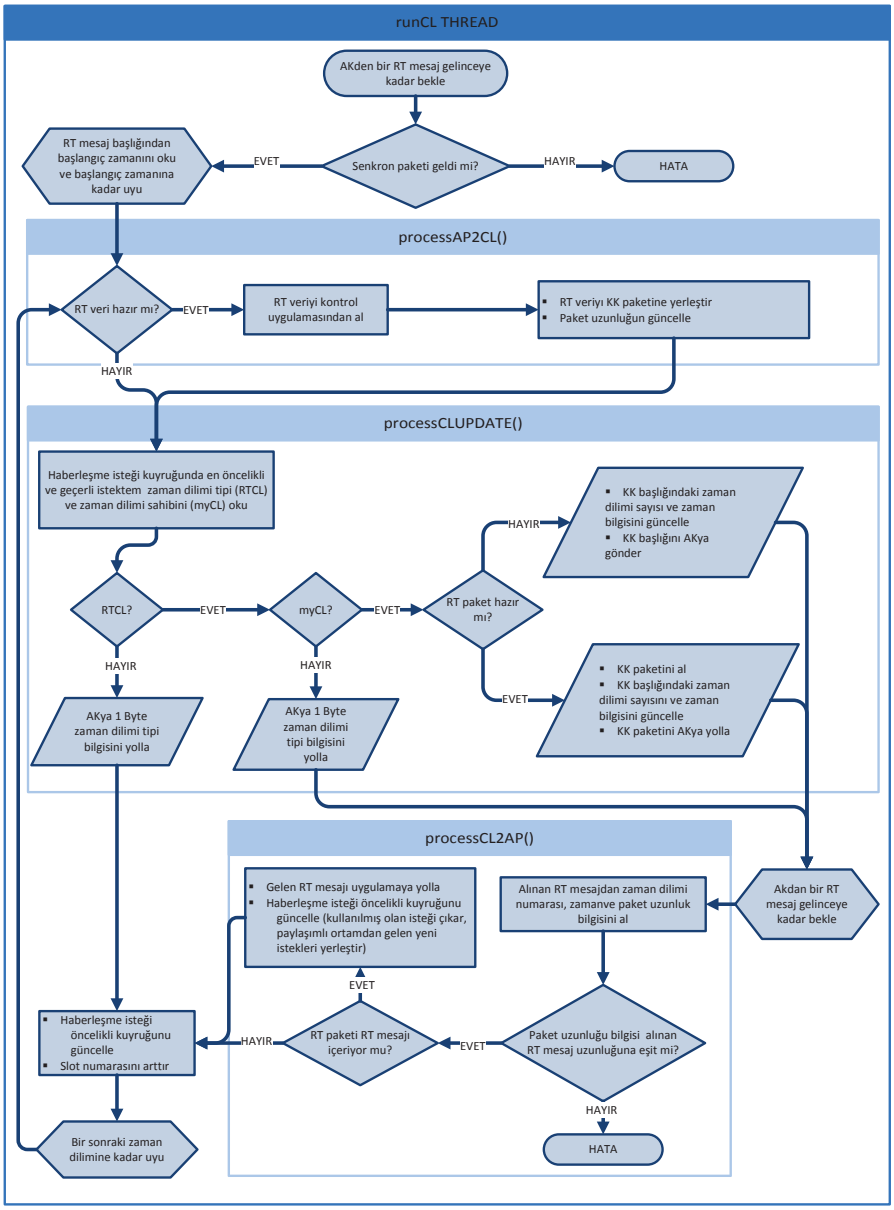
- Eğer $\texttt{myCL}_i$ = false ise RT_APP_MESSAGE boş bir mesajdır. Bu durumda CL_i IL_i programına sadece karar değişkenlerini içeren 1 Bytelık bir bilgi gönderir.
- Eğer $\texttt{RTCL}_i$ = true ve $\texttt{myCL}_i$ = true ise düğüm $i$nin $\texttt{Tx}_i$ belleğinde duran ilk RT mesajı yollaması beklenmektedir. Uygulama haberleşme isteği servis edilmesine ve haberleşme hakkı almasına rağmen henüz gönderecek veri hazır değilse $\texttt{Tx}_i$ boş olabilir. Bu durumda $\texttt{RTCL}_i$ = true ve $\texttt{myCL}_i$ = true değişkenleri AKnına bir mesaj olmadan CL2ILRT$_i(\texttt{RTCL}_i, \texttt{myCL}_i, m)$ fonksiyonu ile iletilir. Eğer $\texttt{Tx}_i$ boş değilse RT_APP_MESSAGE bellekten çıkarılır ve IL_i'ye karakter aracı (character device) ile yollanır. Mesaj daha sonra paylaşımlı ortamdan yollanır.



Şekil 15: KK işparçacığı uyandırılma akış diyagramı.

IL_i paylaşımlı ortamdan gerçek zamanlı trafiğe ayırılmış bir zaman diliminde gerçek zamanlı veri taşıyan bir Ethernet paketi aldığı zaman bu paketi (CL_PACKET) CL_iye yollar. CL_i paketteki haberleşme isteklerini doğru sırada PQ_i kuyruğuna koyar.

KK çalışması akış diyagramı Şekil 16'de görülebilir.



Şekil 16: KK Akış Diyagramı.

3.4.8 Arayüz Katmanı (AK-IL) Gerçeklemesi

AK güncel zaman diliminde KKda gelen karara uygun olarak gerçek zamanlı olan ya da olmayan paketi paylaşımlı ortama yollamakla yükümlüdür. KK ile aynı şekilde düğüm *i* üzerinde koşan AK bileşeni IL_i olarak gösterilecektir.

AK gerçeklemesi 3 çekirdek alanı modülünden oluşmaktadır. Bunlar; `Fragmentation` (Parçalama), `Reassembly` (Birleştirme) ve `IL` modülleridir. `IL` modülü $\text{CL2ILRT}(b_1, b_2, m)_i$ ve$\text{IL2SM}(m)_i$ fonksiyonlarını gerçekler.

Gerçek zamanlı olan ve olmayan paketlerin kontrollu olarak gönderilmesi için `e1000e` Ethernet sürücüsünün uyarlanması gerekmiştir. `Fragmentation` ve `Reassembly` çalıştırılmadıkları zaman uyuyan çekirdek alanı işparçacıkları (thread) olarak gerçekleştirilmişlerdir.

Bu modüller ile birlikte uyarlanmş `e1000e` sürücüsü sadece sabit boyda AK Ethernet çerçevelerinin paylaşımlı ortama gönderilmesini sağlamaktadır.

AKnındaki $\texttt{TxnRT}_i$ belleği iki FIFO kuyruk olarak gerçekleştirilmıştir. Bunun sebebi gerçek zamnalı olmayan trafik olarak gönderilen IEEE 1588 paketlerinin diğer gerçek zamanlı olmayan paketlere göre yüksek öncelikle gönderilmesini sağlamaktır. Buna göre $\texttt{TxnRTH}_i$ öncelikli IEEE 1588 eşzamanlama paketlerini ve $\texttt{TxnRT}_i$ kalan gerçek zamnalı olmayan mesajları depolamaktadır. Düğüm *i* gerçek zamanlı olmayan bir paket göndereceği zaman önce $\texttt{TxnRTH}_i$ belleğini kontrol eder. Burada göndereceği paket varsa gönderir yoksa $\texttt{TxnRT}_i$ belleğini kontrol eder ve buradan bir paket gönderir.

Paylaşımlı Ortam Haberleşmesi ve IL Modülü

$Frame_{max}$ herhangi bir gerçek zamanlı paketi sığdırabilecek şekilde seçilecektir. Bir düğüm üzerinde Bölüm 4.3'de açıklandığı gibi birden fazla Ethernet arayüzü olabilir. Buna göre D^{2}RIP çalıştıran arayüzü `eth0` olarak tanımlamaktayız. Bu arayüz için `e1000e` Ethernet sürücü modülü kullanılmaktadır. Bu modüle `eth0` arayüzüne gelen bütün paketlerin yolunu kesen, büyüklüklerini kontrol eden ve uzun geçek zamanlı olmayan paketleri `Fragmentation` işparçacığına yönlendiren bir uyarlanmış gönderme fonksiyonu olan `e1000_xmit_frame_modified()` eklenmiştir. Parçalanmış ve gönderilmeye hazır gerçek zamanalı olmayan paketler yukarıda tanımlanan yüksek öncelikli ve normal öncelikli kuyruklara $\text{AP2ILNRT}(m)_i$ fonksiyonu ile konulmaktadır.

IL_i, RT ve nRT paketleri CL_iden gelen bilgiye göre gönderir. CL_iden alınan `CL_PACKET`leri ve $\texttt{TxnRT}_i$ ve $\texttt{TxnRTH}_i$den alınan `nRT_PACKET`leri uygun şekilde paketlendikten sonra standard gönderme fonksiyonu `e1000_xmit_frame()`na gönderilerek veri yapısındaki $\text{IL2SM}(m)_i$ fonksiyonu gerçekleştirilmış olur. Güvenirlik katmanı (Dependability Layer) gönderen düğümün aynı zamanda kendi paketini de almasını gerektirmektedir. Hub ile bağlı Ethernet ağlarında bu olmadığı için `ECHO_SENDER` opsiyonu ile gönderenin tüme gönderilen paketi alamaması durumunda gönderilen `CL_PACKET`lerinin gönderen düğüme arayüzden geri döngü ile getirilmesi yapılmaktadır.

Paylaşımlı ortamdan paket alınmasını sağlayan $\text{SM2IL}(m)$ fonksiyonu aşağıdaki gibi gerçeklenmiştir:

`eth0` arayüzünde bir çerçeve alındığında eğer çerçevedeki `EtherType` parametresi bunun bir `nRT_PACKET` parçası olduğunu gösteriyorsa paket `Reassembly` (birleştirme) işparçacığına yollanır. Birleştirme tamamlandığında paket soket arabelleğini üst katmanlara teslim eden

`netif_rx()` fonksiyonu ile kullanıcı alanına yollanır. Eğer parçalanmamış bir kısa `nRT_PACKET` ise doğrudan standard alma fonksiyonu `e1000_receive_frame()`e gönderilir. Eğer çerçevenin `EtherType` alanı bu paketin bir `CL_PACKET` olduğunu gösteriyorsa $\text{IL2CLRT}(m)_i$ yönlendirilir. Bunun ardından paket kullanıcı alanındakı KK gerçeklemesine geçer.

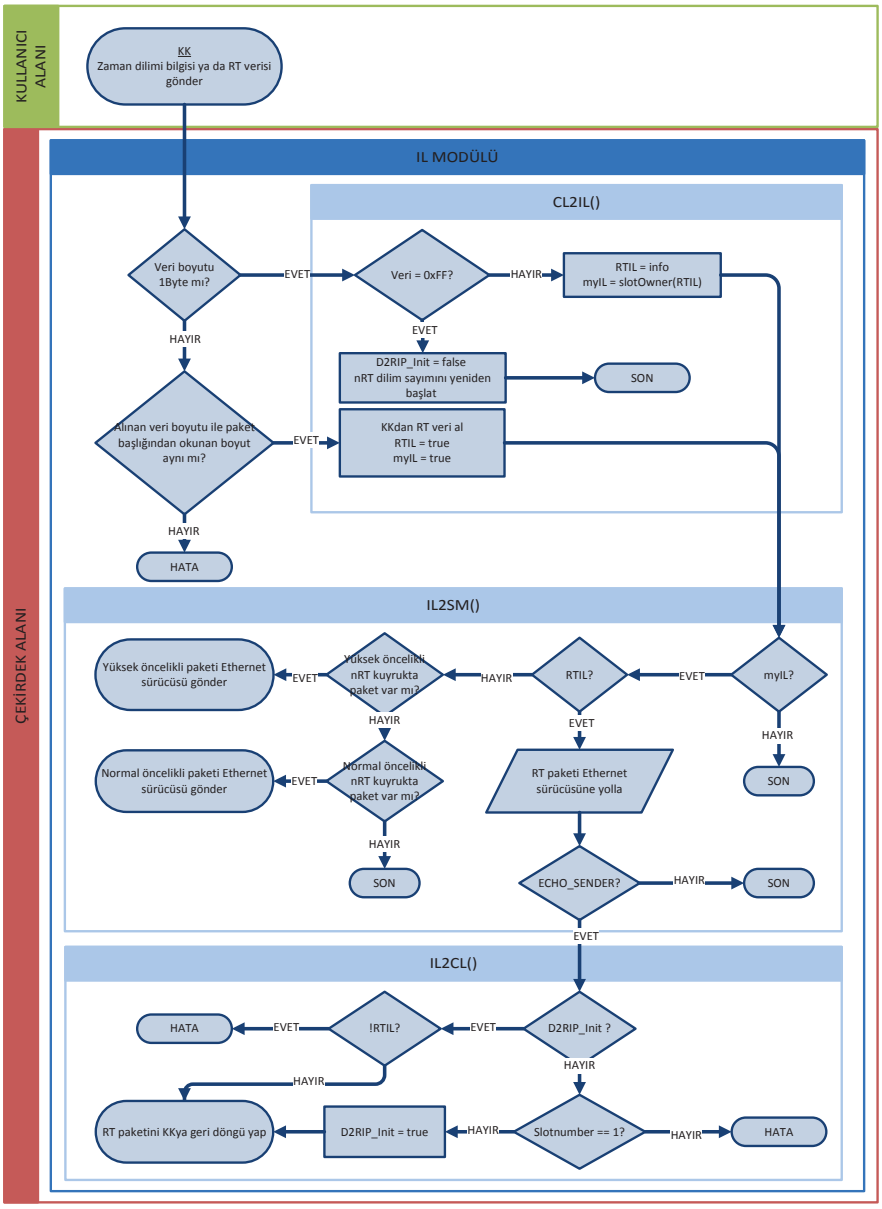
$\text{CL2ILRT}_i(\texttt{RTCL}_i, \texttt{myCL}_i, m)$ karar parametreleri olan $\texttt{RTCL}_i$,$\texttt{myCL}_i$nin CL_iden IL_i'e olası bir uygulama mesajı `RT_APP_MESSAGE` ile geçmesini sağlar.

Her zaman dilimi başında her cihaz $i$de bulunan CL_i karakter aracı yoluyla IL_i'ye bir `CL_PACKET` yollar. Eğer `RT_APP_MESSAGE` boş değilse düğüm $i$nin güncel zaman diliminde gönderecek RT verisi vardır. Bu durumda $\texttt{RTIL}_i = \text{true}$ ve $\texttt{myIL}_i = \text{true}$ olur.

Eğer `RT_APP_MESSAGE` boşsa sadece 1 Bytelık $\texttt{RTCL}_i$,$\texttt{myCL}_i$ bilgisi AK'ya Şekil 14'de görüldüğü gibi paketlenerek geçer.

Eğer $\texttt{RTIL}_i = \text{true}$ ise bunun sonucu olarak $\texttt{myIL}_i = \text{false}$ olur. Eğer $\texttt{RTIL}_i = \text{false}$ ise gerçek zamanlı olmayan trafiği gönderecek düğüm $\texttt{myIL}_i = \texttt{nRTslotOwner()}$ olarak çevrimsel olarak yapılan gerçek zamanlı olmayan zaman dilimi eşleştirmesini döndüren `nRTslotOwner()` fonksiyonu ile yapılır.

IL modülü çalışması akış diyagramı Şekil 17'de görülebilir.



Şekil 17: IL modülü Akış Diyagramı.

Gerçek Zamanlı Olmayan Paketlerin Parçalanması ve Birleştirilmesi

Parçalama (`Fragmentation`) ve Birleştirme (`Reassembly`) çekirdek alanında gerçekleştirilmış ve bir iş kesmesi gelinceye kadar uyuyan iş parçacıkları olarak gerçekleştirilmişierdir. Bu işlevler için bir çerçeve başlığı `sHdr` veri yapısı oluşturulmuştur. Bu yapının alanları Tablo 2'da tanımlanmıştır. Bu yapının büyüklüğü 7 Bytetır.

Alan	**Özellikleri**
nodeID	Gönderen düğümün numarası (ID). Unsigned char (8 bit)
packetID	Paket numarası. Unsigned char (8 bit) ile 256 arka arkaya paket tanımlanır.
packetLength	Paketin parçalanmadan önceki tüm uzunluğu (Byte). Unsigned char (16 bit) ile maksimum uzunlukta Ethernet çerçevesini destekler.
$frameNum$	Paketin parçalanacağı toplam çerçeve sayısı. Unsigned char (8 bit).
$frameSeq$	Parçalandıktan sonra elde edilen bir çerçevenin çerçeve dizisi içinde sayısı. Unsigned char (8 bit) Maximum değeri $frameNum$.
$frameLength$	Çerçevenin uzunluğu (Byte). Unsigned char (8 bit).

Tablo 2: Çerçeve Başlığı veri yapısı.

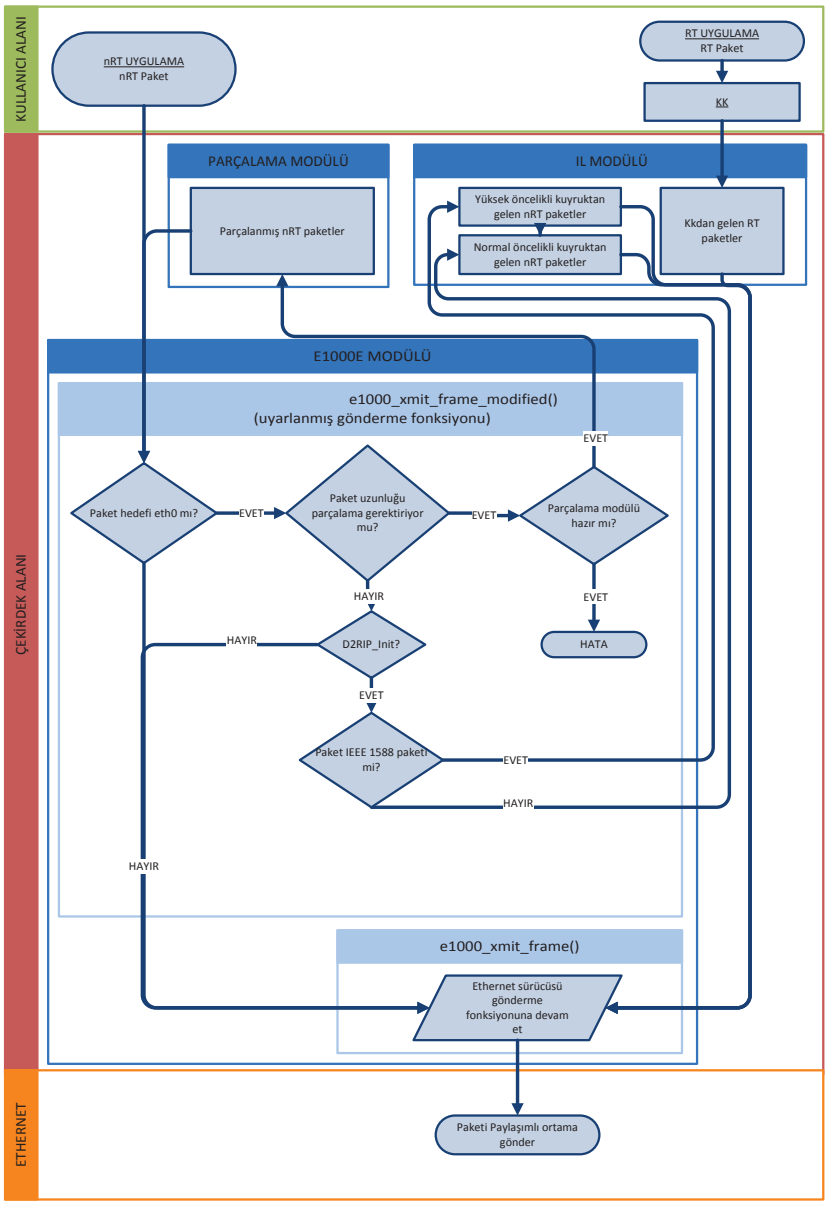
`e1000_xmit_frame_modified()` fonksiyonu uygulama katmanından gelen uzun bir `nRT_PACKET` ile karşılaştığında `Fragmentation` işparçacığını uyandırır. İş parçacığı önce Şekil 2'de görülen bir parça başlığı (fragment header) veri yapısı şablonu hazırlar. Daha sonra parçalanacak yeni paket için *packetID* numarasını artırır ve paket uzunuluğu (*packetTotalLength*) oluşacak toplam parça sayısını ($frameNum$) hesaplar. $frameSeq$ değeri 1 yapılır. Bunun ardından Şekil 14'de görüldüğü gibi her parçada $frameSeq$ artırarak $frameNum$ parça yaratılır.

nRT parça verisi son parça hariç $Frame_{max} - (18 + 7)$ Byte uzunluğundadır. Ayrıca her parça için Ethernet çerçevesinin başlığı da ayrıca hazırlanır ve parçalara eklenir. Her yaratılan parça onları önceliklerine göre $\texttt{TxnRT}_i$ ya da $\texttt{TxnRTH}_i$da depolayan `e1000_xmit_frame_modified()` fonksiyonuna gönderilir.

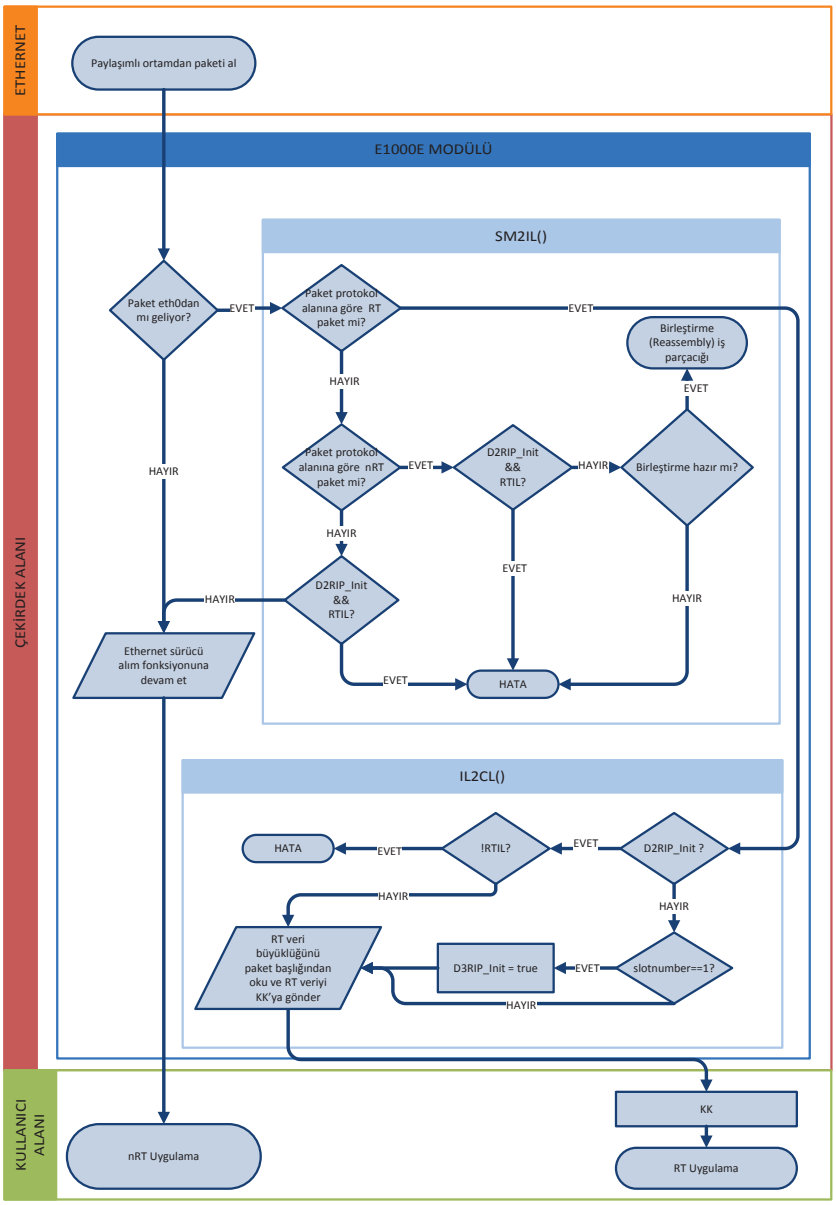
`Reassembly` iş parçacığı `reassembly_pkt_array[sHdr.nodeID].skb` ile gösterilen bir dizi birleştirme belleği tutarak her düğümden alınan parçaları birleştirmek için ayrı bellekler üzerinde çalışır. Paylaşımlı ortamdan bir nRT paket geldiğinde bu iş parçacığı uyanır. Eğer $frameSeq < frameNum$ ve $frameNum > 1$ ise daha alınacak parçalar var demektir. Buna göre alınan parça `reassembly_pkt_array[sHdr.nodeID].skb`'ya kopyalanır. $frameSeq = frameNum$ olduğunda paket `netif_rx()` fonksiyonu ile kullanıcı alanına yollanır.

`Reassembly` iş parçacığı birleştirilen paketin uzunluğunu başlıktaki *packetLength* alanı ile karşılaştırarak olası hataları kontrol eder, uyumsuz ise paketi atar.

Şekil 18 ve 19'de paket gönderimi ve alımı akış diyagramları ile birlikte parçalama ve birleştirme işlemleri de gösterilmektedir.



Şekil 18: Paket gönderimi akış diyagramı.



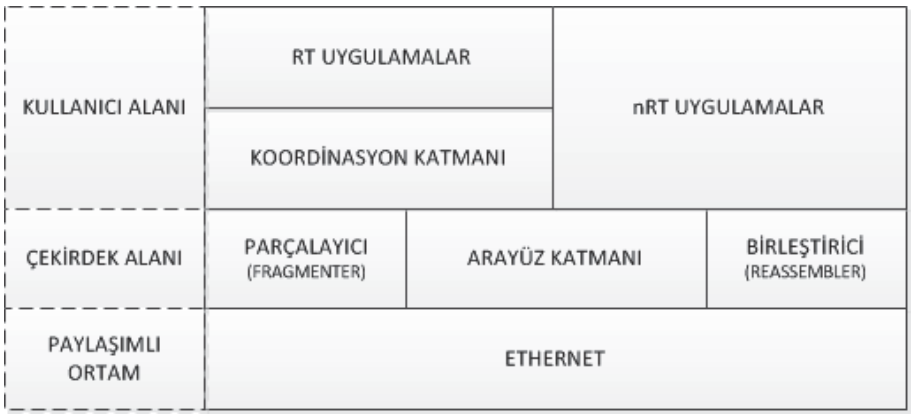
Şekil 19: Paket alımı akış diyagramı.

D^2RIP İlk Versiyon Gerçeklemeye Göre Yapılan İyileştirmeler Ekler Bölümünde tablolanmıştır.

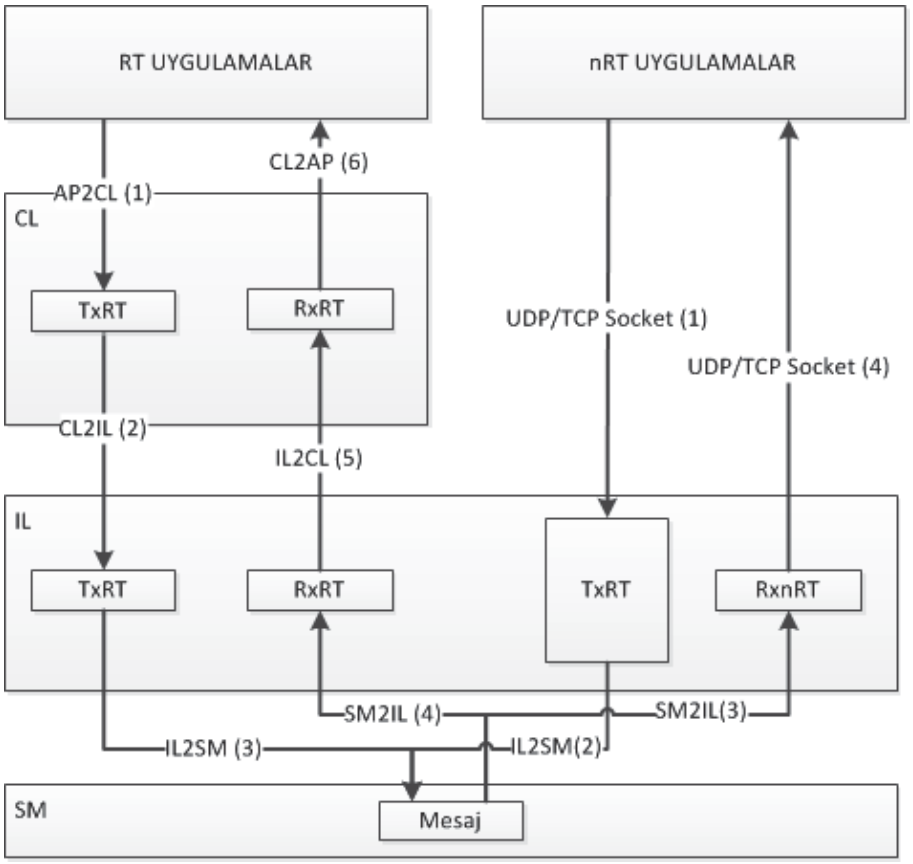
3.4.9 D^2RIP Gerçeklemesi Özet ve Genel Bakış

Yukarıda anlatılan gerçekleştirim bu bölümde iki Şekil ile özetlenmektedir. Şekil 20'de yukarıda detaylı olarak anlatılan Koordinasyon Katmanı, Arayüz Katmanı, uzun gerçek zamanlı olmayan paketleri parçalayan ve birleştiren modüller Lubuntu işletim sisteminde gerçekleştirildiği alanlar görülmektedir.

Bir zaman dilimin başından sonuna kadar hangi fonksiyonların hangi katmanlarda ve hangi sırayla çağırıldıkları ise Şekil de özetlenmektedir. Bu fonksiyonların zamanlaması Bölüm 4.2'de incelenmektedir.

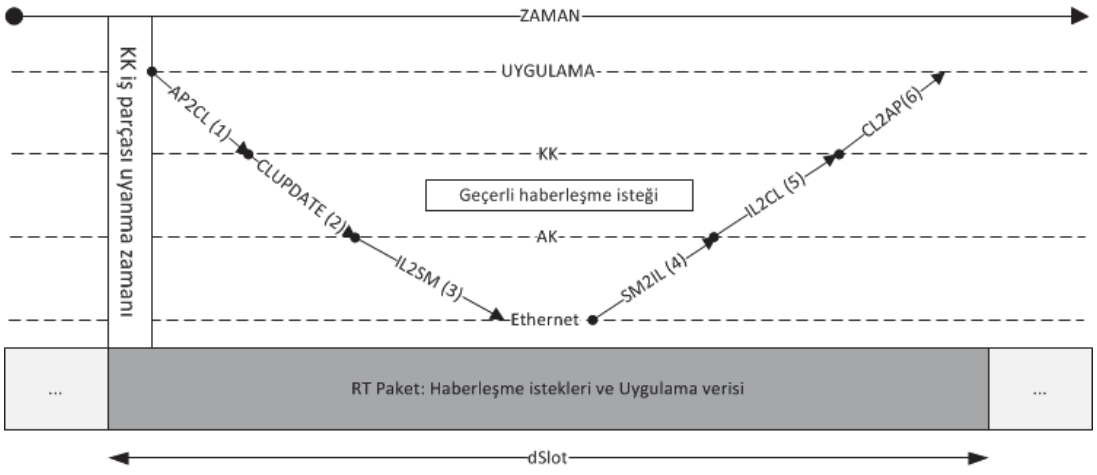


Şekil 20: D^2RIP yazılım modülleri.

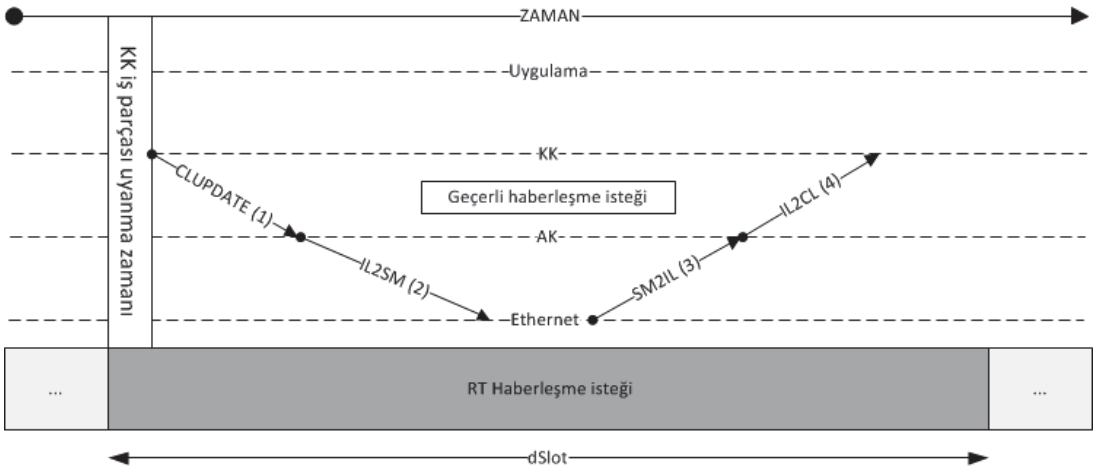


Şekil 21: D^{2}RIP fonksiyonları ve yapılan işler.

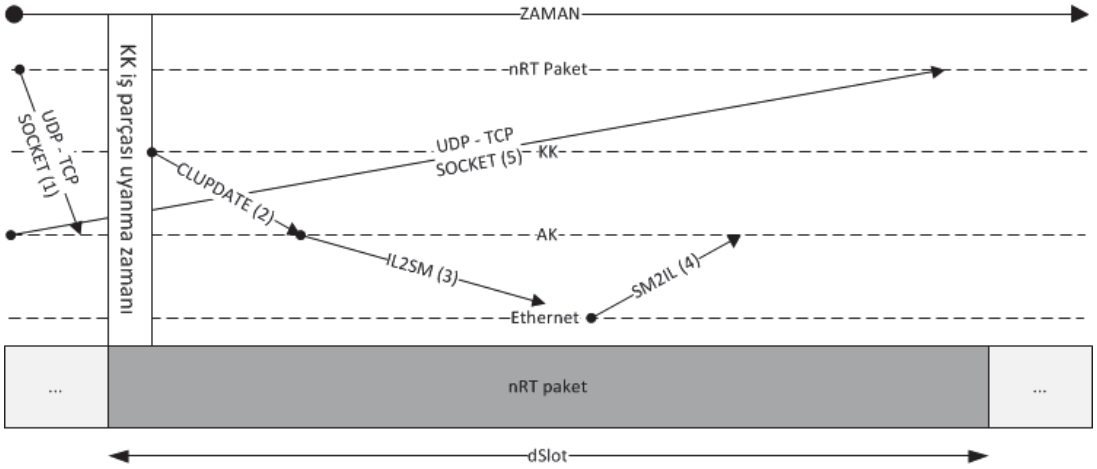
Bu bölümde anlatılan olabilecek üç farklı gönderme olayının zamanlamaları, aşağıdaki şekillerle özetlenmiştir. Şekil 22'de öncelikli haberleşme isteği kuyruğunda ilk mesaj



Şekil 22: Haberleşme istekleri ve RT veri gönderimi zamanlaması.



Şekil 23: RT mesaj olmadan haberleşme istekleri gönderimi zamanlaması.



Şekil 24: nRT mesaj gönderimi zamanlaması.

Yukarıdaki şekillerin haricinde bir durum da hiç bir paketin gönderilmediği ve zaman diliminin boş geçtiği durumdur. Buna bir örnek olarak nRT olarak belirlenen zaman diliminde sıra kendinde olan düğümün gönderecek verisi olmadığı durum gösterilebilir.

3.5 D^{2}RIP Başarım Parametreleri

Bölüm 2.1.1'de sıralanan gereksinimlere uygun olarak bu bölümde D^{2}RIP Başarım Parametreleri tanımlanmaktadır. Bu kapsamda ağa özel,protokola özel ve uygulamaya özel parametreler Tablo 3'da görülebilir.

B_{max} ve B_{min} D^2RIP'in dinamik kapasite ayırımı yeteneği için özel tanımlı parametrelerdir. Sistem çalışması sırasında D^2RIP RT kapasite ayırımı dinamik olarak B_{min} ve B_{max} arasında değişmektedir. Haberleşme istekleri için geçerlilik zamanları ve son gönderilme zamanları

Tablo 3: D^{2}RIP Başarım Parametreleri.

Ağa özel parametreler	
B	Paylaşımlı ortam kapsitesi
Protokola özel parametreler	
$dSlot$	Zaman dilimi uzunluğu
L	RT mesaj uzunluğu (bit)
Uygulamaya özel parametreler	
B_{sum}	Bütün düğümler için en kötü durum RT kapasite ayırımlarının toplamı
B_{max}	Anlık en kötü durum RT kapasite ayırımları
B_{min}	Anlık en iyi durum RT kapasite ayırımları
$eT_{\texttt{req}}$	Haberleşme isteği `req` için geçerlilik zamanı (Eligibility time)
$dT_{\texttt{req}}$	Haberleşme isteği `req` için son gönderilme zamanı (deadline)
$Q_{\texttt{req}}$	`req` ile birlikte öncelikli istek kuyruğundaki maksimum istek sayısı

önceden bilindiği için bu maksimim ve minimum değerler önceden hesaplanabilir. Yine analiz ile istek `req` önünde sıralı diğer isteklerin sayısı $Q_{\texttt{req}}$ da önceden hesaplanabilir. Bu hesaplar örnek uygulama 3.6 bölümünde gösterilmektedir.

Bu parametreler ile Tablo 1'de incelenen ölçütleri *D*^{2}*RIP* için inceleyebiliriz.

Etken RT kapasite (effective RT throughput):

$$B_{\text{eff}} = \frac{L}{dSlot}. \tag{1}$$

Önceki çalışmamız [SCHMIDT et al. (2008)] uygulama parametreleri ve zaman dilimi boyutu arasında bir ilişki bulmaktadır. Buna göre:

$$dSlot \leq \frac{dT_{\texttt{req}} - eT_{\texttt{req}}}{Q_{\texttt{req}} + 1}. \tag{2}$$

Sonuç olarak herhangi bir haberleşme isteği `req` için minimum son gönderme zamanı :

$$dT_{\texttt{req}} \geq dSlot\,(Q_{\texttt{req}} + 1) + eT_{\texttt{req}}. \tag{3}$$

olarak hesaplanır.

(1) ve (3) ifadelerine bakıldığında *dSlot* parametresinin olabildiğince küçültülmesi gerektiği görülmektedir.

D^2*RIP* statik kapasite ayırımlı (TDMA gibi yöntemler) protokollar ile karşılaştırıldığında RT kapasite kazanımı G_{RT} aşağıdaki gibidir:

$$G_{\text{RT}} = \frac{B_{\text{sum}}}{B_{\text{max}}}. \tag{4}$$

Buna göre kullanılabilecek nRT kapasite B_{nRT} sınırları:

$$B - B_{\text{max}} \leq B_{\text{nRT}} \leq B - B_{\text{min}}. \tag{5}$$

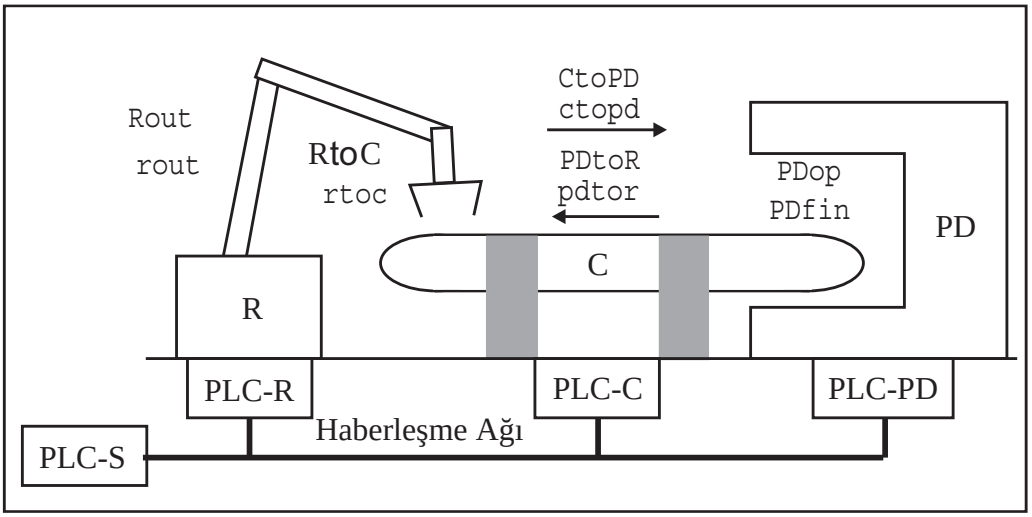
olarak belirlenir.

3.6 Örnek Dağıtılmış Kontrol Uygulaması

D^2*RIP* çalışmasını proje kapsamında yaptığımız [SCHMIDT (2012)] yayınındaki uygulama örneği üzerinde inceleyeceğiz.

Bu örnek seçilirken geliştirilen işçerçevesinin incelenen; aktif olan ve olmayan düğümlerin zaman içinde değişmesi, haberleşme isteğine göre mesaj gönderme sırası kendisine gelen düğümün mesajı hazır olmadığı için haberleşme isteğini güncelleyerek yeniden çizelgelemesi, sağıtık eş zamanlama ve kontrolcu düğümleri koordinasyonu gibi gerekli bütün özelliklerini içermesi dikkate alınmıştır. Buna ek olarak [SCHMIDT (2012)] makalemizde bu örnek için analitik olarak mesaj gecikmesi, olabilecek geçerlilik ve son gönderilme zamanları sınırları hesaplanmıştır.

Şekil. 25'de 3 bileşen bulunmaktadır. Bu bileşenler, bir robot kol (R), iki yönde işlenen ürünleri taşıyan bir taşıma bandı (C) ve ürünleri boyayan bir boyama aracı (PD) dır. Her bileşen ayrı bir PLC aracıyla kotrol edilmektedir (PLC-R, PLC-C, PLC-PD). Bu PLC'leri koordine eden üst düzey kontrolcu PLC-S ile gerçekleştirilmıştır.



Şekil 25: İş hücresi: Robot, taşıma bandı ve boyama aracı.

İstenen çalışma şeklinde ürünler R'den PD'ye C ile taşınmakta, boyamadan sonra ters yönde taşınarak R tarafından iş hücresini terk etmektedir. PLClerin koordinasyonu PLC-S tarafından ağ üzerinden yollanan komutlar ile yapılmaktadır.

Sistemdeki olaylar ve ilgili PLCler Tablo 4'da sıralanmaktadır. Örnek olarak, `RtoC`/`rtoC` ürünün R'den C'ye taşınmasının başlamasını/bitmesini göstermektedir. `RtoC`/`rtoC` olayının olması PLC-S ve PLC-R arasında haberleşme gerektirmektedir. Her işlem bir "start (başla)" ve "completion (bitir)" olayı ile tanımlanmaktadır.

Daha önceki çalışmalarımıza [SCHMIDT et al. (2007), SCHMIDT et al. (2008), SCHMIDT (2012)] uygun olarak bu sistemin çalışması için gerekli üç basamaklı haberleşme modeli aşağıdaki gibidir:

1. Koordinatör PLC-S ilgili bileşenin PLC cihazına olayı gerçekleştirip gerçekleş-

Tablo 4: Örnekteki olaylar listesi.

Olay	Tanım	İlgili PLCler
`RtoC`	Ürünü Rden Cye taşımaya başla	PLC-R, PLC-S
`rtoc`	Ürünü Rden Cye taşımayı bitir	PLC-R, PLC-S
`CtoPD`	Ürünü Cden PDye taşımaya başla	PLC-C, PLC-S
`ctopd`	Ürünü Cden PDye taşımayı bitir	PLC-C, PLC-S
`PDop`	PD çalışmaya başla	PLC-PD, PLC-S
`PDfin`	PD çalışmayı bitir	PLC-PD, PLC-S
`PDtoR`	Ürünü PDden Rye taşımaya başla	PLC-C, PLC-S
`pdtor`	Ürünü PDden Rye taşımayı bitir	PLC-C, PLC-S
`Rout`	Rden ürünü çıkarmaya başla	PLC-R, PLC-S
`rout`	Rden ürünü çıkarmayı bitir	PLC-R, PLC-S

tiremeyeceğini sorar (*sorgu-query*).

2. Soru sorulan PLC olayı gerçekleştirebileceği zaman PLC-S'e haber verir *haber-notification*.

3. PLC-S olayla ilgili bütün PLC cihazlarından haber aldığı zaman hepsine olayı gerçekleştirmeleri için tek bir komut (*komut-command*) yollar ve olay ilgili PLCler tarafından eşzamanlı olarak gerçekleştirilir.

Yukarıda tanımlanan haberleşmenin zamanlaması Şekil 26'de görülebilir.

Şekil 26: Örnek için PLC haberleşmesi zamanlaması.

Örnekte görüldüğü gibi haberleşme ağı üzerinde olan düğümlerin hepsi aynı anda mesaj yollamamakta ve hangisinin ne zaman ve hangi sıra ile mesaj yollayacağı önceden bilinmektedir. Örnek olarak önce, PLC-S, PLC-R'ye ürün yollamak için sorarken kalan PLClerin haberleşmesi gerekli değildir. Bunun ardından sadece PLC-Rnin haberleşmesi beklenmektedir. Buna uygun olarak dinamik şekilde ağ kapasitesi ayırılması mümkündür. Bu şekilde anlık ihtiyaca göre ağ kapasitesinin ayrılması sadece usta yamak protokollarda ek haberleşme ve usta düğümün yüksek işlem yükü ve tek hata noktası olması ile sağlanabilir.

Bu öreneğe uygun olarak D^2RIP çalışması Tablo 5'da özetlenmiştir. Her mesaj için son gönderilme zamanı ve geçerlilik zamanı 5 ms ve 4 ms ($dT = 5$ ms ve $eT = 4$ ms,) olarak belirlenmiştir.

Bu örneğe göre Bölüm 3.5'de açıklanan başarım parametreleri aşağıdaki gibi hesaplanabilir:

Tablo 5: Örneğe göre haberleşme istekleri.

Soru	Haberleşme İsteği	Haber verme	Haberleşme İsteği	Komut	Haberleşme İsteği
?RtoC	(PLC-S,1,4,5)	!RtoC	(PLC-R,1,4,5)	RtoC	(PLS-S,1,4,5)
?rtoc	(PLC-S,1,4,5)	!rtoc	(PLC-R,1,4,5)	rtoc	(PLC-S,1,4,5)
?CtoPD	(PLC-S,1,4,5)	!CtoPD	(PLC-C,1,4,5)	CtoPD	(PLC-S,1,4,5)
?ctopd	(PLC-S,1,4,5)	!ctopd	(PLC-C,1,4,5)	ctopd	(PLC-S,1,4,5)
?PDop	(PLC-S,1,4,5)	!PDop	(PLC-PD,1,4,5)	PDop	(PLC-S,1,4,5)
?PDfin	(PLC-S,1,4,5)	!PDfin	(PLC-PD,1,4,5)	PDfin	(PLC-S,1,4,5)
?PDtoR	(PLC-S,1,4,5)	!PDtoR	(PLC-C,1,4,5)	PDtoR	(PLC-S,1,4,5)
?pdtor	(PLC-S,1,4,5)	!pdtor	(PLC-C,1,4,5)	pdtor	(PLC-S,1,4,5)
?Rout	(PLC-S,1,4,5)	!Rout	(PLC-R,1,4,5)	Rout	(PLC-S,1,4,5)
?rout	(PLC-S,1,4,5)	!rout	(PLC-R,1,4,5)	rout	(PLC-S,1,4,5)

Her zaman tek PLC haberleşeceğiçin haberleşme isteği kuyruğunda en fazla bir istek olacaktır. Bu durumda, $Q_{\max} = 1$dir. Her mesaj için $dT = 5\,\text{ms}$ ve $eT = 4\,\text{ms}$ ise, olabilecek en uzun zaman dilimi: (2):

$$dSlot \leq \frac{5\,\text{ms} - 4\,\text{ms}}{2} = 0.5\,\text{ms}.$$

olacaktır.

Bu örneğe uygun olarak:

$$B_{\text{sum}} = 4\,\frac{dSlot}{5\,ms}B.$$

$$B_{\max} = \frac{dSlot}{4\,ms}B.$$

$$G_{\text{RT}} = \frac{16\,\text{ms}}{5\,\text{ms}} = 3.2.$$

hesaplanabilir.

3.7 TIOA Modellerinin Algoritmik olarak UPPAAL Modellerine Dönüştürülmesi

D^3RIP geliştirme faaliyetleri sırasında, geliştirilen işçerçevesinin simule edilmesi ve formal doğrulamalarının yapılması ihtiyacı ortaya çıkmıştır. Bu amaçla projede UPPAAL ortamı kullanılmıştır. D^3RIP katmanlarının UPPAAL ile simule edilebilmesi ve doğrulamalarının yapılabilmesi için TIOA sentaksı kullanılarak geliştirilen katman modellerinin UPPAAL tarafında kullanılabilecek davranışsal modellere çevrilme ihtiyaçları doğmuştur. Bu ihtiyacı karşılamaya yönelik çalışmalar yürütülürken literatürde TIOA modellerinin davranışsal UPPAAL modellerine çevrilmesinde kullanılacak kabul görümüş bir yöntemin olmadığı farkedilmiştir. Hem proje geliştirme faaliyetlerinde kullanılmak hem de literatürdeki bu açığı kapatmak amacıyla TIOA modellerini UPPAAL modellerine çevirmede kullanılacak olan

TU_CONVERT algoritması geliştirilmıştir. Bu bölümde, geliştirilen algoritma ve algoritma işleyişi anlatılmaktadır.

3.7.1 TIOA Model Kısıtlamaları

TU_CONVERT algoritması [GARLAND et al. (2005)] ve [KAYNAR et al. (2003)]'te verilen TIOA modelini kullanmaktadır. Bu modele göre bir TIOA modeli $A = (X, Q, Q_0, S, D, T)$ şeklinde tanımlanmaktadır. Bu tanımdaki değişkenler:

- X: *değişkenler* kümesini,
- Q: *durum* kümesini,
- $Q_0 \subseteq Q$: *başlangıç durumları* kümesini,
- S: *aksiyonlar* kümesini,
- D: *geçişler* kümesini,
- T: *sürekli zaman geçişleri* kümesini,

belirtmektedir.

TU_CONVERT algoritmasının TIOA modelleri üzerinde kullanılabilmesi için modellerin uyması gereken bazı kısıtlamalar bulunmaktadır. Bu nedenle geliştirilen algoritma TIOA modelleri üzerinde belirli kabuller yapmaktadır. Bu kabuller:

Assumption 1 Her bir TIOA modelinde yalnızca tek bir başlangıç durumu vardır. Bir başka deyişle ile başlangıç durumları kümesi tek elemalıdır. ($Q_0 = \{q_0\}$) □

Assumption 2 TIOA modeli içersinde kullanılan bütün data tipleri UPPAAL'da da kullanılabilir. □

Assumption 3 Çevrime sokulacak TIOA modellerinde kullanılan saat değişkenlerinin değerleri eğimi bir olan sabit bir eğri şeklinde artar ($d(\texttt{clock}) = 1$) ve clock değişkenleri üzerinde sıfırlama ($\texttt{clock} := 0$) dışında herhangi bir aritmetik işlem yapılamaz. □

Assumption 4 TIOA modellerindeki geçişlerin etkileri model sürekli zaman geçişlerinde tanımlı bir durma (stop) durumunu tetiklemez. Bir başka değişle TIOA modellerinde oluşan herhangi bir geçiş diğer geçişler tarafından kullanılan durma durumlarına yol açacak ani değişiklikler barındırmaz □

Assumption 5 Modeldeki sürekli zaman geçişleri birden fazla zaman dilimi içermezler. □

Yukarıda tanımlı varsayımları kullanarak hazırlanmış *TU_CONVERT* algoritması Bölüm 3.7.2'de verilmektedir.

3.7.2 *TU_CONVERT* Algoritması

TU_CONVERT girdi olarak Bölüm 3.7.1'de belirtilen TIOA modelini kullanıp çıktı olarak ise davranışsal UPPAAL modeli $U(F)$ üreten bir algoritmadır. Algoritma UPPAAL'ın 4.0.13 ve 4.1.3 versiyonları ile uyumludur. Bu bölümde algoritma modeli verilmekte ve işleyişi model üzerinden anlatılmaktadır. Algoritma detayları ve örnek uygulamalar IEEE Software Engineering dergisine gönderilen makalede [KARTAL et al. (2013)] verilmektedir. *TU_CONVERT* algoritması Algoritma 1'de verilmektedir.

TIOA Parametreleri

TIOA tanımında bulunan parametreler (P) belirli bir değer taşıyan sabit parametreler olabildiği gibi belirli bir veri tipini belirtmede kullanılan parametreler de olabilmektedir. UPPAAL yalnızca sabit parametreler ile tanımlanmış modellere izin vermektedir. O nedenle TIOA parametre setinde bulunan sabit parametreler direk olarak UPPAAL parametreleri olarak kullanılabilmektedir. Veri tipi gösteren parametreler ise UPPAAL ortamına global değişkenler olarak aktarılmaktadır. Parametre tanımlamaları Algoritma 1'in 1 ve 4 satırları arasında verilmektedir.

TIOA Değişkenleri

X kümesi içerisinde tanımlanan TIOA değişkenleri, tek bir otomata ait değişkenler olduğundan bu değişkenler UPPAAL'a *lokal değişkenler* olarak taşınmaktadır. Değişken tanımlamaları Algoritma 1'in 5 ve 12 satırları arasında verilmektedir.

TIOA Aksiyonları

TIOA tanımındaki aksiyonlar otomat içi (internal) H ve otomatlar arası (external) $I \cup O$ olmak üzere iki kümede toplanmaktadır. TIOA modelinde tanımlı aksiyonlar otomatlar arası parametre geçirmede kullanılmaktadır. Bu geçişler otomat işleyişinde ani durum değişiklikleri tetiklemektedir. UPPAAL ortamında ani durum değişikliklerinin tetiklenebilmesi ve aksiyonlara birden fazla otomat tarafından erişilebilmesi amacıyla otomatlar arası aksiyonlar *urgent broadcast channel* olarak tanımlanmaktadır. Bu aksiyon tanımlamaları Algoritma 1'in 13 ve 16 satırları arasında gösterilmektedir. Otomatlar arası aksiyonların taşıdığı parametreler ise birden fazla otomat tarafından erişilebilmesi amacıyla UPPAAL'a global parametreler olarak aktarılmıştır. Global parametre tanımlamaları Algoritma 1'in 17 ve 22 satırları arasında gösterilmektedir. TIOA modeli iç aksiyonları için herhangi bir kanal tanımlaması yapılmayıp yalnızca durum geçiş özellikleri kullanılmıştır.

TIOA Durum Geçişleri

TIOA modelinde tanımlı durum geçişleri (t_a $a \in S$), geçiş sırasında herhangi bir zaman kaybı yaşatmayan anlık geçişlerdir. UPPAAL üzerinde bu durum geçişleri bir lokasyon (location) ve iki köşe (edge) ile tanımlanmaktadır. UPPAAL'a aktarılmış basit bir durum geçişi Şekil 27'da gösterilmektedir.

UPPAAL ortamındaki otomat modeli herhangi bir aksiyon tetiği gelene kadar bekleme durumunda ($l_{idle} \in L$) kalır. Bir aksiyon tarafından tetiklendiği anda ise o aksiyon ile ilgili tanımlanmış lokasyona ($l_a \in L$) ilgili köşeyi ($e_a \in E$) kullanarak geçiş yapar. Bu geçiş sonrasında vakit kaybetmeden bekleme lokasyonu l_{idle}'a $e_{a,lb}$ köşesi ile dönüş yapar. l_a lokasyonunda vakit kaybedilmemesi önemli olduğundan bu lokasyon UPPAAL'da *acil (urgent)* olarak işaretlenir.

l_{idle} lokasyonu Algoritma 1'in 25 ve 29 satırları arasında tanımlanmaktadır. Köşe geçişleri ise

input : $A(P)$
output: $U(F)$

Global Structures
for *all* $p \in P_V$ **do**
  $typedef\ struct\ type(p)$;
end
Local Variables (V)
for *all* $x \in X$ **do**
  if $type(x)\ != analog$ **then**
    $type(x)\ v_x$;
  else
    $clock\ v_x$;
  end
end
Global Channels
for *all* $a \in (I \cup O)$ **do**
  $urgent\ broadcast\ chan\ c_a$;
end
Global Variables (G)
for *all* $a \in (I \cup O)$ **do**
  for *all parameters p passed by* t_a **do**
    $type(p)\ g$;
  end
end
Locations (L)
$committed\ initial\ location\ l_{init}$;
$location\ l_{idle}$;
$l_{idle}.invariant = \neg stop(X)$ (*)
for *all* $a \in S$ **do**
  $urgent\ location\ l_a$;
end
Edges (E)
$edge\ e_{init}$; $with\ l_{init} \overset{e_{init}}{\rightarrow} l_{idle}$
$e_0.update = InitAutomaton(X_0)$; (**)
for *all* $a \in S$ **do**
  $edge\ e_a\ with\ l_{idle} \overset{e_a}{\rightarrow} l_a$
  $e_a.guard = t_a.pre$; % $t_a \in D$
  if $a \in I$ **then**
    $e_a.synch = c_a?$; % *input channel*
  end
  if $a \in O$ **then**
    $e_a.update = t_a.eff$
    $update\ t_a.glob$;
  else
    $e_a.update = t_a.eff$;
  end
  $edge\ e_{a,lb}\ with\ l_a \overset{e_{a,lb}}{\rightarrow} l_{idle}$: % *loopback edge*
  if $a \in O$ **then**
    $e_{a,lb}.synch = c_a!$; % *output channel*
  end
end
(*) Negation ($\neg$) operation ($\leq$ for analog variables).
(**) *InitAutomaton* assigns the initial values in X_0 to the local variables in L.

Algorithm 1: *TU_CONVERT* Algoritması

34 ve 45 satırlarında tanımlanmaktadır.

UPPAAL ortamında tanımlanan köşeler *guard*, *synch ve update* alanları ile tanımlanmaktadır.

Guard: e_a geçişinin guard alanı, bu geçişin olabilmesi için modele ait lokal değişkenlerin sağlaması gereken özellikleri belirlemektedir. Bu özellikler direk olarak TIOA modelindeki geçişlerin (t_a) ön koşullarına (preconditions) tekabül etmektedir. Otomatlar arası içe dönük ($a \in I$) geçişlerde herhangi bir ön koşul bulunmamaktadır. O nedenle bahsi geçen durum yalnızca otomat içi geçişler ve otomatlar arası dışa dönük geçişler ($a \in H \cup O$) için geçerlidir. $e_{a,lb}$ geçişi hemen gerçekleşmesi gereken bir geçiş olduğundan bu geçişler için ön koşul tanımlı değildir. UPPAAL'da köşeler için tanımlı guard koşulları Algoritma 1'in 35 satırında belirtilmektedir.

Sync: e_a köşesinin senkronizasyon alanı bu köşe ile hangi dış aksiyonun ($a \in O$) veya iç aksiyonun ($a \in I$) tetikleneceği bilgisini taşımaktadır. Eğer e_a köşesinin tanımlanmasında kullanılan t_a geçişi bir dışa gönük aksiyonu tetikliyorsa, UPPAAL'da bu alan c_a! şeklinde ifade edilmektedir. (Bknz. Algoritma 1'in 46 ve 48 satırları arası.)

Diğer taraftan, eğer köşe içe dönük bir aksiyon tarafından tetikleniyorsa bu alan c_a? şeklinde ifade edilmektedir. (Bknz. Algoritma 1'in 36 ve 38 satırları arası.)

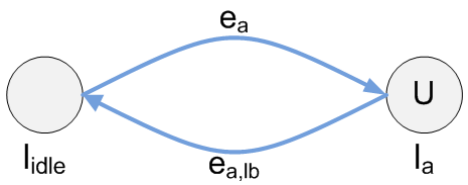
Update: TIOA modelinde gerçekleşen her bir geçiş t_a TIOA değişkenlerinin değişmesine yol açmaktadır. Bu değişiklik t_a geçişinin *eff* alanında belirtilmektedir. Eğer geçişi tetikleyen aksiyon a bir iç aksiyon ise ($a \in H$) değişkenlerin yeni değeri otomatın iç dinamikleri ile hesaplanmaktadır. Öte yandan a aksiyonu otomatlar arası bir aksiyonsa ($a \in I \cup O$) otomat değişkenlerin yeni değerleri otomatın iç dinamikleri ve a aksiyonu ile geçirilen parametrelere bağlı olarak değişmektedir.

UPPAAL'da tanımlı köşelerin update alanlarının nasıl doldurulduğu Algoritma 1'in 39 ve 44 satırları arasında gösterilmektedir. Bu işlem sırasında duruma göre otomatın iç değişkenleri ile birlikte otomatlar arası geçişler nedeniyle tanımlanan global parametreler de kullanılmaktadır.

TIOA Sürekli Zaman Geçişleri

TIOA'da tanımlı sürekli zaman geçişleri (trajectories) otomat işleyişindeki analog değişkenlerin zamana bağlı değişimlerini göstermektedir. TIOA modelinde tanımlı bütün analog değişkenlerin herbirine karşılık olarak UPPAAL'da değişim hızı 1 olan ($d(v_a) = 1$) clock değişkenleri (v_a) tanımlanmaktadır.

TIOA modeli sürekli zaman geçişleri içerisinde tanımlanan *stop* durumları zaman geçişinin durup aksiyon geçişi yapılması gerektiği anları belirtmekte kullanılmaktadır. Bu nedenle bu kısmdaki ifadenin tersi ($\neg stop(X)$) UPPAAL modeli bekleme durumunda (l_{idle}) *invariant* olarak tanımlanmaktadır. *stop* durumlarının UPPAAL modeline aktarımı Algoritma 1'in 26 satırında gösterilmektedir.



Şekil 27: UPPAAL'da geçişler

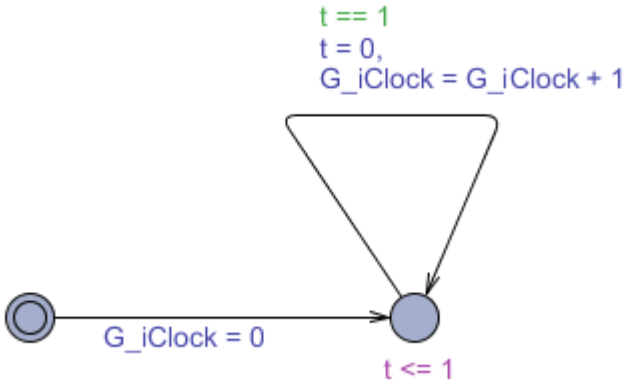
Otomat İlklendirmesi

Otomat modelinin UPPAAL ortamında ilklendirilebilmesi için bir lokasyon l_{init} ve bir köşe e_{init} tanımlanmaktadır. l_{init} lokasyonundan e_{init} köşesi ile l_{idle} lokasyonuna yapılan geçiş otomat modelinin ilk geçişi olmaya zorlanmaktadır. Bu amaçla l_{idle} lokasyonu UPPAAL'da *committed initial* lokasyon olarak işaretlenmektedir. Değişken ilklendirmesi için de e_{init} geçişinin *eff* alanı kullanılmaktadır. TIOA modeli içersindeki ilk durum değerleri bu geçişte tanımlı $InitAutomaton(q_0)$ fonksiyonu ile UPPAAL modeline aktarılmaktadır. l_{init} lokasyon tanımı Algoritma 1'in 24 satırında verilmektedir. e_{init} köşe tanımlaması ve ilklendirilmesi ise sırasıyla 31 ve 32 satırlarında verilmektedir.

3.7.3 UPPAAL Tamsayı saat gerçeklemesi

Geliştirilecek olan işçerçevesinde katmanların davranışlarını belirleyen etkenlerden bir tanesi, belki de en önemlisi zaman değişimidir. Geliştirilecek olan işçerçevesinin gerçek zamanlı haberleşme protokollerinin geliştirilmesine yönelik olması sebebi ile aksiyonlardan birçoğu zaman ile tetiklenmektedir. Bir başka değişle işçerçevesinin temeli zaman tetikli (event-triggered) olaylara bağımlıdır. İşçerçevesi TIOA modeline bakıldığında her katmanda zaman değişimi *now* analog değişkeni ile takip edilmektedir. Bu değişken 1'er adımlar ile *d(now)=1* artan bir değişim hızına sahiptir. UPPALL ile yapılan çalışmalarda karşılaşılan ilk problem, katmanlar arası senkronizasyonun sağlanmasında kullanılacak olan 1'er adım ile değişen saat (clock) değişkeninin gerçeklenmesinde yaşanmıştır.

UPPAAL kendi içersinde gömülü bir saat (clock) yapısına sahiptir. Buradaki ilk sorun bu saat değişkeninin değişim sıklığının ayarlanamıyor olması idi. Karşılaşılan ikinci sorun ise saat değişkeninin değerinin başka tipte bir değişkene aktarılamıyor olması idi. Bu sorun modelde zamana bayalı kıstasların (constraint) yazılamıyor olmasına neden olmaka idi. Yapılan literatür araştırmasında, ilgili sorunların UPPAAL kullanan birçok kişi tarafından uğraşılan problemler olduğu görülmüştür. Verilen çözüm önerileri değerlendirilmiş ve [XIAOWAN HUANG 1997)]'te verilen integer saat çözümü işçerçevesi gerçeklemesine en uygun çözüm olarak belirlenmiştir. Burada verilen çözüm yolu işçerçevesi ihtiyaçlarına uygun olarak güncellenmiş ve şekil 28'de verilen çözüm geliştirilmıştır.



Şekil 28: UPPAAL ile Tamsayı Saat Gerçeklemesi

Verilen çözümde UPPAAL'ın kullandığı analog saat değişkenini değeri monitör edilerek her 1

adımda integer olarak tutulan saat değişkeni 1 ilerletilmektedir. Bir node içinde gerçeklenecek olan Koordinasyon, Arayüz ve Güvenirlilik Katmanları aynı donanım üzerinde gerçekleneceğinden aynı saat değişkenine senkron olmaları modelin gerçeğe uygunluğu açısından önem arz etmektedir. Verilen çözüm ile bu gerek sağlanmış olmaktadır. İlerleyen bölümlerde anlatılmakta olan Arayüz Katmanı (IL) ve Paylaşımlı Ortam (SM) modelleri verilen integer saat değişkenine senkron olarak çalışmaktadır.

3.8 D^{3}RIP Güvenirlik Katmanı ve İş Çerçevesi Modeli

D^3*RIP* Güvenilirlik Katman tasarımı Timed Input/Output Automata (TIOA) sentaksı kullanılarak yapılmıştır. Bu sayede tasarım aşamasında oluşabilecek bileşen seviyesi hataların önlenmesi (fault prevention) planlanmıştır. Bu katman ile çalışma esnasında oluşabilecek, geçici (transient) yazılım hatalarının [AVIZIENIS et al. (2004)] giderilmesi hedeflenmiştir.

D^3*RIP* çalışması zaman eşlemesine, önerilen kaynak paylaşımı ise dağıtık düğümlerde aynı kaynak paylaşım bilgisinin bulunmasına dayanmaktadır. Örneğin gerçek zamanlı mesajların gönderileceği zaman dilimlerinin aidiyet bilgisi Koordinasyon Katmanı tarafından belirlenirken yapılacak bir hata birden fazla düğümün Paylaşımlı Ortama aynı anda mesaj göndermesine yol açabilecektir. Bu durum paylaşımlı ortamda mesaj çarpışması (collision) oluşmasına yol açacak ve sistem işleyişi sekteye uğrayacaktır.

D^3*RIP* işleyişini bozabilecek diğer bir muhtemel hata ise sistem durumunun belirlenmesinde kullanılan değişken içeriklerinin bozulmasıdır. *D*^{3}*RIP* ile geliştirilen haberleşme protokollerinin gerçeklemesi yapılırken oluşabilecek herhangi bir bellek hatası (bellek kaçağı, sınır aşımı vb.) sistem değişken içeriklerinin bozulmasına, sistem değişken içeriklerinin bozulması da beklenmeyen durum geçişlerine (unintended state alterations) yol açabilecektir. Örneğin Arayüz Katmanında, o anki zaman diliminin düğüme ait olup / olmadığı bilgisinin tutulduğu *myIL* değişken içeriği bellek hatası sebebi ile değişirse düğüm mesaj göndermemesi gerekiren beklenmedik bir şekilde mesaj göndermeye çalışacaktır. Bu durum Paylaşımlı Ortamda mesaj çarpışmasına yol açabileceği gibi, daha sonraki zaman dilimleri için verilecek kaynak paylaşımı kararlarını da bozabilecektir.

Yukarıda örnekleri sıralanan hatalar zamanlama hataları ve içerik hataları [AVIZIENIS et al. (2004)] olarak gruplanabilir. Güvenilirlik Katman tasarımı ile birlikte tamamlanmış *D*^{3}*RIP* mimarisi Şekil 1'de görüldüğü gibidir.

Güvenilirlik Katman işleyişi Kontrol Noktası Oluşturma ve Geri Dönme prensibine dayanmaktadır. Sistemde oluşabilecek hatalar düğüm seviyesinde periyodik olarak gerçekleştirilen Kabul Testleri ile yakalanıp, hata durumunda Kurtarma Noktalarına geri dönülerek giderilmektedir. Bu amaçla Güvenilirlik Kaman'ının Şekil 1'te de görüldüğü gibi diğer katmanların tamamı ile arayüzü bulunmaktadır. IL arayüzü hem Kurtarma Bloğu oluşturmada kullanılacak durum bilgilerinin DL'e taşınmasında hem de DL çıktılarının IL'e iletilmesinde kullanılan çift yönlü bir arayüzdür. CL arayüzü de benzer amaçla kullanılan çift yönlü bir arayüzdür. DL ile SM arasındaki arayüzün tek amacı, SM üzerinden gelen mesajların kontrol amacıyla DL'e aktarılmasıdır. Son olarak uygulama katmanı ile olan arayüz ise DL çıktılarının uygulamaya iletilmesinde kullanılan tek yönlü bir arayüzdür.

3.8.1 Güvenilirlik Katmanı TIOA Modeli

Önceki bölümde de anlatıldığı gibi katman işleyişi Kurtarma Bloğu Oluşturma ve Geri Dönme mekanizmasına dayanmaktadır. Bilindiği gibi bu mekanizmanın yarattığı temel sorun, dağıtık sistemlere getirdiği ek mesaj yükü sebebi ile ağ verimliliğine görülen kayıptır. Bu durumun ortadan kaldırılabilmesi için [RAMANATHAN (1993)]'de anlatılan ve eş zamanlılık prensibine dayanan uygulama değiştirilerek D^3RIP mimarisine adapte edilmiştir. Katman modeli ve işleyişi bu bölümde detaylı olarak anlatılmaktadır.

Güvenilirlik Katmanı TIOA modeli Şekil 29'te verilmektedir. DL, döngü sayısı *cyc*, durum geçişlerinde kullanılan zaman değişkenleri t_0 t_1, IL CL ve DL bilgilerini saklayan veri tipleri A_{IL}, A_{CL}, A_{DL} ve mesaj sırası (queue) veri tipi Q değişkenleri ile karakterize edilmektedir. Geliştirilen DL modeli işçerçevesi seviyesinde bir soyutlama (abstraction) olup A_{DL} veri tipi ile farklı DL protokollerinin yaratılmasına olanak sağlamaktadır. Protokol geliştirilirken yapılacak her bir A_{DL} içerik seçimi yeni bir DL protokolü oluşturacaktır. DL'i oluşturan veri yapıları aşağıdaki gibidir:

Durum Bilgileri (states)
Katman içi bilgiler bu değişkenler ile saklanmaktadır. DL içersindeki zaman geçişi analog *now* değişkeni, kabul testlerinin sonuç bilgisi *atRes* değişkeni, herhangi bir geri dönme ihtiyacı olup olmadığı bilgisi *rbReq* değişkeni, o anki zaman slotunun tipi (gerçek zamanlı / değil) *rtSlot* değişkeni, durum geçişini kontrol eden durum numarası *stNo* değişkeni, mesaj gönderen düğüm numarası *nodeID* değişkeni, geri dönülecek durum numarası *rbSt* değişkeni, kurtarma noktası oluşturma işleminde hangi noktada olunduğu bilgisi *cnt* değişkeni, IL katmanı için tutulan geçmiş bilgileri *ILHist* değişkeni, CL için tutulan geçmiş bilgileri *CLHist* değişkeni, DL protokol karakteristiğini belirleyecek bilgiler *vDL* değişkeni, mesaj gönderen düğümün *vCL* içeriği *rVCL* değişkeni ile saklanmaktadır.

Aksiyonlar ve Geçişler (Signatures and Transitions)

DL TIOA modelinin diğer katman ve uygulama modelleri ile arayüzü geçiş aksiyonları ile kurulmaktadır. Şekil 5'te gösterilen $UPDVIL$ aksiyonu ile IL'den gelen güncel *vIL* bilgisi DL'e taşınmaktadır. DL bu bilgiyi IL'in geri dönebileceği kurtarma noktaları ($ILHist$) oluşturmada kullanmaktadır. $UPDVCL$ aksiyonu ile CL'den gelen güncel bilgiler DL'e taşınmaktadır. Bu bilgiler de CL için kurtarma noktası oluşturmada ve bulunulan zaman diliminin tipinin (gerçek zamanlı / değil) anlaşılmasında kullanılmaktadır. $SM2ILDL$ aksiyonu, SM üzerinden gelen mesajların DL'e aktarılmasında kullanılmaktadır. DL bu aksiyon ile kabul testlerinde kullanacağı bilgileri almakta ve bu bilgiler uyarınca iç işleyişini yönetmektedir.

$ATEST$ aksiyonu DL tarafından tetiklenen bir iç aksiyondur. Bu aksiyon SM'den gelen mesaj içeriğine bağlı olarak $SM2ILDL$ aksiyonunun alınmasından hemen sonra tetiklenmektedir. DL $ATEST$ geçişi ile bulunduğu düğümün kabul testini gerçekleştirmekte ve hata bulunması durumunda geri dönülecek durum numarasını hesaplamaktadır. $SENDRES$ aksiyonu, $ATEST$ aksiyonundan hemen sonra tetiklenmektedir. Bu aksiyon ile DL IL'e yaptığı kabul testi sonucunu iletmektedir.

Son olarak, *RBACK* aksiyonu, $SM2ILDL$ ile alınan mesaj içeriğine bağlı olarak tetiklenen bir aksiyondur. Bu aksiyon ile DL, IL, CL ve uygulama katmanlarına geri dönüş (rollback) yapılması isteğini iletmektedir. Bir zaman dilimi içersindeki D^3RIP işleyişi ve bu işleyişte yukarıda anlatılan aksiyonların yeri Şekil 30'da anlatılmaktadır.

TIOA $DL_i(cyc: int, t_0: int, t_1: int, A_{IL}: Type, A_{CL}: Type, A_{DL}: Type, InitDL: A_{DL}, Q: Type)$

states

$\texttt{now}_i^a: R := 0$
$\texttt{atRes}_i^d: bool := \text{true}$
$\texttt{rbReq}_i^d: bool := \text{false}$
$\texttt{rtSlot}_i^d: bool := \text{false}$
$\texttt{stNo}_i^d: int := -1$
$\texttt{nodeID}_i^d: int := -1$
$\texttt{rbSt}_i^d: int := -1$
$\texttt{cnt}_i^d: int := 0$
$\texttt{ILHist}_i^d[cyc]: H_{IL}$
$\texttt{vCLHist}_i^d[cyc]: A_{CL}$
$\texttt{vDL}_i^d: A_{DL} := InitDL$
$\texttt{rVCL}_i^d: A_{CL}$

signatures

input UPDVIL$(vIL: A_{IL}, TXnRT: Q, RXnRT: Q)_i$
input UPDVCL$(vCL: A_{CL}, Tx: V, Rx: Q, \texttt{RTCL}_i^d: bool)_i$
input SM2ILDL$(m: M)$
internal ATEST$()_i$
output RBACK$(ILHT: H_{IL}, clHT: H_{CL}, rbSt: int)_i$
output SENDRES$(atRes: bool, rbSt: int)_i$

trajectories

stop when

$(\texttt{stNo}_i^d = 1) \wedge (\texttt{now}_i^a = t_0)$
$(\texttt{stNo}_i^d = 2) \wedge (\texttt{now}_i^a = t_1)$
$(\texttt{stNo}_i^d = 3) \wedge (\texttt{now}_i^a = t_0)$

transitions

input UPDVIL$(vIL, TXnRT, RXnRT)_i$

eff:
$\texttt{ILHist}_i^d[cnt].vIL := vIL$
$\texttt{ILHist}_i^d[cnt].TXnRT := TXnRT$
$\texttt{ILHist}_i^d[cnt].RXnRT := RXnRT$

input UPDVCL$(vCL, Tx, Rx, RTCL)_i$

eff:
$\texttt{rtSlot}_i^d = RTCL$
$\texttt{CLHist}_i^d[cnt].vCL := vCL$
$\texttt{CLHist}_i^d[cnt].Tx := Tx$
$\texttt{CLHist}_i^d[cnt].Rx := Rx$

input SM2ILDL(m)

eff:
$\texttt{nodeID}_i^d = m.nodeID$
$\texttt{rVCL}_i^d = m.vCL$
$\texttt{rbSt}_i^d = m.rbSt$
$\texttt{now}_i^a := 0$
if $m.atRes == false$
$\texttt{stNo}_i^d = 3$
else
$\texttt{stNo}_i^d = 1$

output SENDRES$(\texttt{atRes}_i^d, \texttt{rbSt}_i^d)_i$

pre:
$\texttt{stNo}_i^d = 2$
$\texttt{now}_i^a = t_1$
eff:
$\texttt{stNo}_i^d = -1$

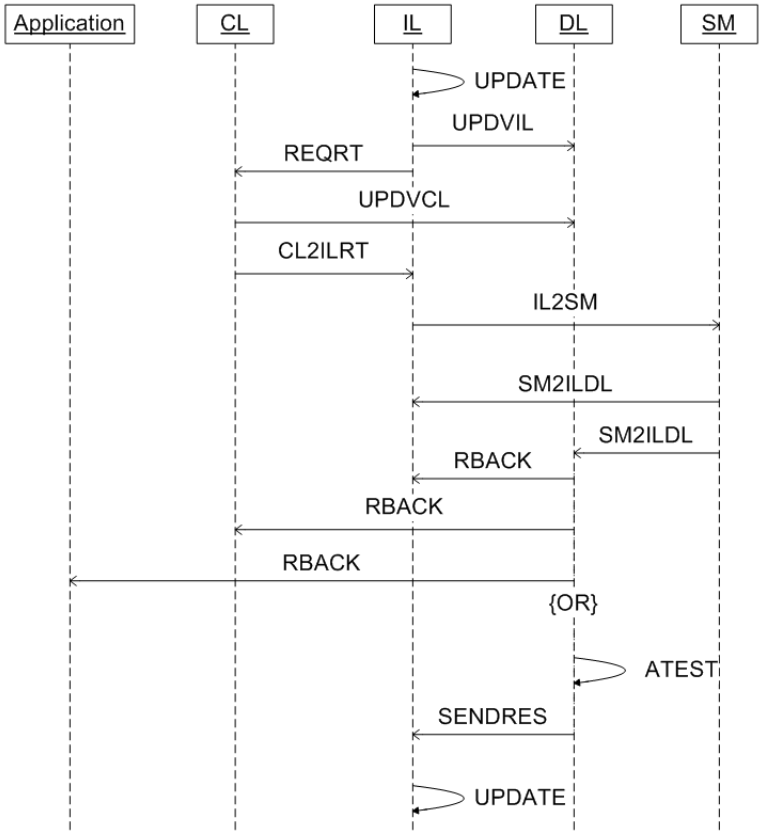
internal ATEST$()_i$

pre:
$\texttt{stNo}_i^d = 1$
$\texttt{now}_i^a = t_0$
eff:
$\texttt{cnt}_i^d += 1$
if $\texttt{rbReq}_i^d == false$
$\texttt{atRes}_i^d = fAT(CLHist[cnt - 1].vCL, vDL, rVCL, nodeID, rtSlot)$
if $\texttt{atRes}_i^d == true$
$\texttt{rbSt}_i^d = \texttt{cnt}_i^d - 1$
else
$\texttt{rbReq}_i^d = true$
$\texttt{rbSt}_i^d = \texttt{cnt}_i^d - 2$
if $\texttt{rbSt}_i^d == -1$
$\texttt{rbSt}_i^d = cyc - 1$
if $\texttt{cnt}_i^d == cyc$
$\texttt{cnt}_i^d = 0$
$\texttt{stNo}_i^d = 2$

output RBACK$(\texttt{ilHT}_i^d, \texttt{clHT}_i^d, \texttt{rbSt}_i^d)_i$

pre:
$\texttt{stNo}_i^d = 3$
$\texttt{now}_i^a = t_0$
eff:
$\texttt{cnt}_i^d = \texttt{rbSt}_i^d$
if $\texttt{cnt}_i^d == cyc$
$\texttt{cnt}_i^d = 0$
$\texttt{ilHT}_i^d = \texttt{ILHist}_i^d[rbSt]$
$\texttt{clHT}_i^d = \texttt{CLHist}_i^d[rbSt]$
$\texttt{stNo}_i^d = -1$
$\texttt{rbReq}_i^d = \text{false}$

Şekil 29: Güvenilirlik Katmanı TIOA Modeli



Şekil 30: Güvenilirlik Katmanı İşleyişi

Şekil 30'da görüldüğü gibi DL işleyişi zaman diliminin hemen başında IL'den gelen *UPDVIL* aksiyonu ile başlamaktadır. Bu aksiyon ile DL IL'den gelen *vIL* bilgisini kurtarma noktası oluşturmak amacıyla saklamaktadır.

UPDVIL aksiyonundan bir süre sonra CL'den *UPDVCL* aksiyonu gelmektedir. DL bu aksiyon ile CL'den gelen bilgileri kurtarma noktası oluşturmak amacıyla saklamaktadır. CL'den gelen bilgiler aynı zamanda kabul testi yapılırken de diğer düğümlerle olan kaynak paylaşımının tutarlılığının kontrolünde kullanılmaktadır.

Zaman diliminin sonlarına doğru SM'den *SM2ILDL* aksiyonu alınmaktadır. Bu aksiyon ile SM üzerinden gelen mesaj alınmakta ve güvenilirlikle ilgili kısımları ayıklanmaktadır. D^3RIP mimarisine DL'in eklenmesi ile birlikte mesaj başlıklarına kabul test sonucu (*m.atRes*), mesaj gönderen düğümün *vCL* bilgisi (*m.vCL*) ve ID bilgisi (*m.nodeID*) ile herhangi bir hata olması durumunda geri dönülecek durum bilgisi (*m.rbSt*) alanları eklenmiştir. DL işleyişi SM'den gelen mesaj içerisindeki *m.atRes* bilgisine göre bu noktadan itibaren farklılık göstermektedir.

Gelen mesajda herhangi bir hata durumu bilgisi varsa işleyiş *RBACK* aksiyonu ile devam etmektedir. DL bu aksiyon ile CL, IL ve uygulama katmanına eski bir duruma geri dönülmesi gerektiğini bildirmektedir.

Mesaj içeriğinde hata olmaması durumunda ise işleyiş *ATEST* aksiyonu ile devam etmektedir. DL bu aksiyon ile o zaman dilimine ayit kabul testini gerçekleştirmektedir. Kabul test içeriği protokol uygulamasına göre değişiklik gösterebilecektir. Bu nedenle test içeriği iş çerçevesinde

$fAT()$ fonksiyonu ile soyutlanmıştır. Örnek bir test fonksiyonu Şekil 31'de verilmektedir.

fAT($\texttt{vCLHT}_i^{\text{d}}, \texttt{vDL}_i^{\text{d}}, \texttt{rVCL}_i^{\text{d}}, \texttt{nodeID}_i^{\text{d}}, \texttt{rtSlot}_i^{\text{d}}$)

if ($\texttt{rtSlot}_i^{\text{d}} == true$) $\wedge$($\texttt{vCLHT.PQ}_i$.Top.$b == nodeID$)

if $\texttt{vCLHT}_i^{\text{d}} == \texttt{rVCL}_i^{\text{d}}$

return(true)

else if ($\texttt{rtSlot}_i^{\text{d}} == false$) $\wedge$($\texttt{vDL}[cnt-1] == nodeID$)

return(true)

else

return(false)

Şekil 31: Örnek Kabul Testi

$ATEST$ aksiyonunun hemen sonrasında test sonucu DL'den tetiklenen $SENDRES$ aksiyonu ile IL'e iletilmektedir. Buradaki temel amaç test sonucunun, düğümün göndereceği bir sonraki mesaj başlığına eklenerek ağdaki diğer düğümlere aktarılmasını sağlamaktır.

İşleyişten de anlaşılacağı gibi DL katmanı sisteme düğümler arası ek bir mesaj yükü getirmemektedir. Paylaşımlı ortam üzerinde çalışılması ve gönderilen her bir mesajın bütün düğümler tarafından alınıyor olmasının avantajları kullanılarak test sonuçları ve hata bilgileri ağa iletilmektedir. Buradaki en önemli husus ne kadar süre ile geri dönme noktası geçmişinin saklanacağı sorusudur. Bu sorunun cevabı için endüstriyel kontrol uygulamalarının periyodik çalışma prensibinden faydalanılmıştır.

D^3RIP ile geliştirilen protokol ailesi çalıştırılmadan önce ağdaki bütün düğümlerin en az 1 defa ağa eriştikleri minimum zaman dilimi sayısı belirlenmeli ve *cyc* değişkeni ile DL'e iletilmelidir. Bu değer DL'e ne kadar süre ile geçmiş bilgileri saklaması gerektiğini göstermektedir. Bu amaçla DL içerisinde tutulan yığınlar (buffers) dairesel tanımlanmıştır.

3.8.2 Güvenilirlik Katmanı Tasarım Varsayımları

Güvenilirlik Katmanının Bölüm 3.8.1'de anlatıldığı şekilde çalışabilmesi için sağlanması gereken gereksinimler bulunmaktadır.

Katman kullanılarak geliştirilen protokollerde işleyiş sırasının düzgün olabilmesi için DL konfigüre edilirken kullanılan zaman değişkenlerinden t_1, t_0 değerinden büyük seçilmelidir. İki zaman değişkeni arasındaki fark ($t_1 - t_0$) DL'in Kabul Testi yapmada harcadığı işlem süresinden (*cmpDL*) daha büyük seçilmelidir. (Kabul 6)

Assumption 6 $t_1 - t_0 > cmpDL$ □

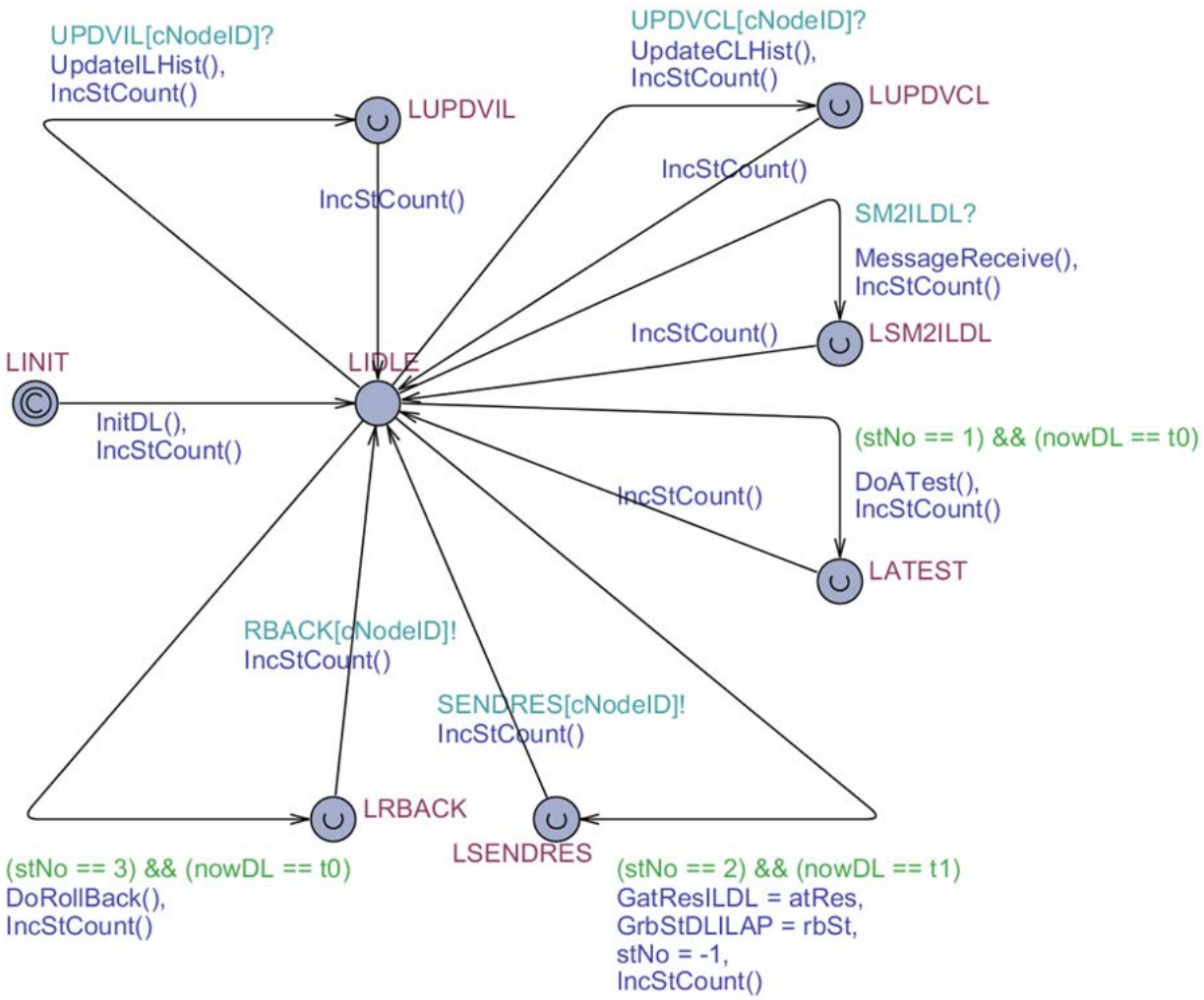
DL tasarımı aynı anda ağda bulunan düğümlerden yalnızca 1 tanesinde hata oluşacağı varsayımına dayanmaktadır. Bir başka değişle, kabul test fonksiyonu $fAT()$ aynı anda en fazla bir düğümde hata dönmelidir. (Kabul 7)

Assumption 7 If $fAT()_i = false$ than $fAT()_j = true \; \forall \; j \in 1,2...N \setminus i$ □

Katman işleyişinin bozulmaması için DL'e paylaşımlı ortamdan, bir zaman dilimi içersinde en fazla 1 mesaj gelmelidir. Bu durum CL ve IL tarafından garanti edilmektedir. Bu katmanlarda herhangi bir sorun olsa dahi bir zaman diliminde 2 mesajın birden Paylaşımlı Ortamda (SM) bulunması mesaj çarpışmasına yol açacağından DL'e aynı zaman diliminde birden fazla mesajın gelmesi imkanlı değildir.

3.8.3 Katman Doğrulaması

Şekil 29'te verilen katman modelinin gereksinimleri sağladığının garanti edilmesi için doğrulama çalışmaları yürütülmüştür. Bu amaçla TIOA modeli [KARTAL et al. (2013)]'de belirtilen *TU_CONVERT* algoritması kullanılarak UPPAAL'a aktarılmıştır. DL'in UPPAAL'da oluşturulmuş davranışsal modeli Şekil 32'de verilmektedir.



Şekil 32: Güvenilirlik Katmanı Davranışsal Modeli

Şekil 32'de verilen model üzerinde UPPAAL *verifyta* motoru işletilerek yapılan doğrulama faaliyetleri ile gereksinimlerin sağlandığı doğrulanmıştır. Doğrulaması yapılan gereksinimler ve bu gereksinimlerin doğrulanmasında kullanılan TCTL sorgu cümleleri aşağıda verilmektedir.

Iddia 1 Katmanın kabul testi yapabilmesi (ATEST durumuna geçebilmesi) ancak zamanın çalışma başında belirlenen t0 anına gelmesi ve stNo değişkeninin 1 olması ile mümkündür. □

Ispat 1 Bu iddianın ispatı için UPPAAL'da Sorgu 1'te belirtilen cümle kullanılmıştır. Verifyta motoru ile bütün durum geçişleri UPPAAL tarafından kontrol edilmiş ve sorgu cümlesindeki iddia ile çelişen geçişler bulunmadığı gösterilerek iddia doğrulanmıştır. □

Sorgu 1 $A[]\ DL.LATEST\ imply\ (DL.nowDL\ ==\ t_0)$ and $(DL.stNo\ ==\ 1)$ □

Iddia 2 Katmanın geri dönme (roll-back) işlevini tetikleyebilmesi (LRBACK durumuna geçebilmesi) ancak zamanın çalışma başında belirlene t_0 anına gelmesi ve stNo değişkeninin 3 olması ile mümkündür. □

Ispat 2 Bu iddianın ispatı için UPPAAL'da Sorgu 2'te belirtilen cümle kullanılmıştır. Verifyta motoru ile bütün durum geçişleri UPPAAL tarafından kontrol edilmiş ve sorgu cümlesindeki iddia ile çelişen geçişler bulunmadığı gösterilerek iddia doğrulanmıştır. □

Sorgu 2 $A[]\ DL.LRBACK\ imply\ (DL.nowDL\ ==\ t0)$ and $(DL.stNo\ ==\ 3)$ □

Iddia 3 Katmanın kabul testi sonuçlarını IL'e gönderebilmesi (LSENDRES durumuna geçebilmesi) ancak zamanın çalışma başında belirlenen t1 anına gelmesi ve stNo değişkeninin 2 olması ile mümkündür. □

Ispat 3 Bu iddianın ispatı için UPPAAL'da Sorgu 3'te belirtilen cümle kullanılmıştır. Verifyta motoru ile bütün durum geçişleri UPPAAL tarafından kontrol edilmiş ve sorgu cümlesindeki iddia ile çelişen geçişler bulunmadığı gösterilerek iddia doğrulanmıştır. □

Sorgu 3 $A[]\ DL.LSENDRES\ imply\ (DL.nowDL\ ==\ t1)$ and $(DL.stNo\ ==\ 2)$ □

Iddia 4 Katman çalışmasında herhangi bir durum geçişi nedeniyle tetiklenecek bir deadlock durumu yoktur. □

Ispat 4 Bu iddianın ispatı için UPPAAL'da iki düğümlü bir simülasyon ortamı oluşturulmuştur. Simülasyon ortamındaki mesajlaşma her iki düğümdeki bütün katmanların bütün durum geçişlerini tetikleyecek şekilde ayarlanmıştır. (Bu durum simülasyon sonucunda UPPAAL tarafından üretilen kapsama raporu ile doğrulanmıştır.) Hazırlanan simülasyon ortamı üzerinde Sorgu 4'te verilen sorgu cümlesi işletilerek katmanda, hatta D^3RIP tasarımının hiç bir yerinde deadlock yaratacak bir hatanın olmadığı doğrulanmıştır. □

Sorgu 4 $A[]$ not deadlock □

Iddia 5 Katman'da kısıtlı zaman içerisinde sonsuz sayıda durum geçişi yaratarak işleyişin ilerlemesini engelleyecek bir durum yoktur. (DL katmanı Non-Zeno [Gómez (2007)] özelliğe sahiptir). □

Ispat 5 İddia'nın ispatı için DL katmanı UPPAAL modeli içerisine durum geçiş sayacı eklenmiştir. Bu sayaç her bir durum geçişinde artırılmaktadır. Bahsi geçen sayaç kullanılarak, İddia 4 için oluşturulan simülasyon ortamında Sorgu 5'te verilen cümle işletilmiştir. Simülasyon ortamında bütün durum geçişleri kapsanarak DL'in kısıtlı zamanda sonsuz durum geçişi yapmadığı (non-zeno davranışa sahip olduğu) doğrulanmıştır. □

Sorgu 5 $A[]\ DL.LRBACK\ imply\ (DL.stCount\ <\ 16)$ □

3.9 Güncel Arayüz ve Koordinasyon Katman Modelleri

D^3RIP tasarimina DL'in eklenmesi ve UPPAAL ortaminda yapilan dogrulama çalismalarinda görülen sonuçlar nedeni ile IL ve CL tasarimlarinda degisiklikler olusmustur. Bu degisikliklerin temel sebepleri asagidaki gibidir:

- Güvenilirlik Katmani (DL) ile arayüz kurmada kullanilan aksiyonlarin Arayüz (IL) ve Koordinasyon (CL) katmanlarina eklenmesi.
- TIOA modellerinin UPPAAL ortamina tasinirken uymalari gereken ve Bölüm 3.7.1'da verilen kisitlar.
- IL ve CL tasariminda sistem güvenilirligini artirabilmek için yapilan degisiklikler. (Bu amaçla Arayüz Katman tasarimi bütün kaynak paylasimi kararlarini CL'e birakacak şekilde güncellenmistir.)

Bahsi geçen sebepler ile yapilan degisiklikler sonrasi güncellenen IL ve CL TIOA modelleri sirasi ile Bölüm 3.9.1 ve 3.9.2'da verilmektedir.

3.9.1 Güncel Arayüz Katman (IL) Tasarımı

Bu bölümde Arayüz Katman tasariminda yapilan temel degisiklikler anlatilmaktadir. Degisiklikler sonrasi güncellenen katman modeli Sekil 33'da verilmistir.

Arayüz Katmani is akisinda yapilan temel degisiklik gerçek-zamanli zaman yariklari (time slot) için verilen aidiyet karar mekanizmasinda olmustur. Sekil 33'da verilen modelde görüldügü gibi IL zaman yarigi tipini (rt/nrt) kontrol etmeden her zaman yariginda CL'e istek (*REQRT* aksiyonu ile) göndermektedir. Bu aksiyona cevap olarak CL tarafindan gönderilen *CL2ILRT* aksiyonu içersindeki $b1$ degiskeni ile zaman yarigi tipine karar verilmekte, ayni aksiyon içindeki $b2$ degiskeni ise zaman yariginin aidiyet bilgisinin hesaplanmasinda kullanilmaktadir. Bu degisiklik ile kaynak paylasimi kararlarinin tamami CL'de toplanmis ve hata olasiligi azaltilmistir.

Arayüz katmanina eklenen *UPDVIL, SENDRES ve RBACK* aksiyonlari ile katmanin DL ile arayüzü kurulmustur. Bu aksiyonlardan *UPDVIL* IL tarafinda tutulan güncel vIL bilgisinin DL katmanina iletilmesinde, *SENDRES* DL katmaninda gerçeklestirilen kabul testi sonucunun alinmasinda, *RBACK* aksiyonu ise DL tarafindan karari verilen geri dönme islevine iliskin bilgilerin alinmasinda kullanilmaktadir.

Arayüz Katman tasarimini UPPAAL'a tasiyabilmek için analog degiskenler üzerinde yapilan aritmetik islemlerin ortadan kaldirilmasi gerekmistir. Bu islemler ortadana kaldirilarak IL islevselliginin korunabilmesi için saat ilklendirme (clock reset) islemi uygulanmistir.

TIOA $IL_i(dSlot\colon int, t_0\colon int, t_1\colon int, t_2\colon int, t_3\colon int, M : Type, Q : Type, A_{IL} : Type, H_{IL} : Type)$

states

$\texttt{now}_i^{\texttt{a}}\colon R := dSlot$
$\texttt{TxRT}_i^{\texttt{d}}\colon M :=$ empty
$\texttt{TxnRT}_i^{\texttt{d}}\colon Q :=$ empty
$\texttt{RxRT}_i^{\texttt{d}}\colon M :=$ empty
$\texttt{RxnRT}_i^{\texttt{d}}\colon Q :=$ empty
$\texttt{RTIL}_i^{\texttt{d}}\colon bool :=$ false
$\texttt{myIL}_i^{\texttt{d}}\colon bool :=$ false
$\texttt{reqIL}_i^{\texttt{d}}\colon bool :=$ false
$\texttt{sendVIL}_i^{\texttt{d}}\colon bool :=$ false
$\texttt{at}_i^{\texttt{d}}\colon bool :=$ true
$\texttt{checkPT}_i^{\texttt{d}}\colon int :=$ -1
$\texttt{vIL}_i^{\texttt{d}}\colon A_{IL} :=$ InitV

transitions

internal UPDATE()$_i$

pre:

$\texttt{now}_i^{\texttt{a}} = dSlot$

eff:

$\texttt{vIL}_i^{\texttt{d}} = f_{\text{upd}}(\texttt{vIL}_i^{\texttt{d}}, \texttt{RTIL}_i^{\texttt{d}})$
$\texttt{now}_i^{\texttt{a}} := 0$
$\texttt{reqIL}_i^{\texttt{d}} :=$ true
$\texttt{sendVIL}_i^{\texttt{d}} :=$ true

output UPDVIL$(vIL, TXnRT, RXnRT)_i$

pre:

$\texttt{sendVIL}_i^{\texttt{d}} =$ true
$\texttt{now}_i^{\texttt{a}} = t_0$

eff:

$\texttt{sendVIL}_i^{\texttt{d}} :=$ false

input SENDRES$(atRes, rbSt)_i$

eff:

$\texttt{at}_i^{\texttt{d}} :=$ atRes
$\texttt{checkPT}_i^{\texttt{d}} :=$ rbSt

input RBACK$(ilHT, clHT, rbSt)_i$

eff:

$\texttt{vIL}_i^{\texttt{d}} :=$ ilHT.$\texttt{vIL}_i^{\texttt{d}}$
$\texttt{TXnRT}_i^{\texttt{d}} :=$ ilHT.$\texttt{TXnRT}_i^{\texttt{d}}$
$\texttt{RXnRT}_i^{\texttt{d}} :=$ ilHT.$\texttt{RXnRT}_i^{\texttt{d}}$
$\texttt{now}_i^{\texttt{a}} := 0$
$\texttt{reqIL}_i^{\texttt{d}} :=$ true
$\texttt{sendVIL}_i^{\texttt{d}} :=$ true
$\texttt{at}_i^{\texttt{d}} :=$ true

output REQRT()$_i$

pre:

$\texttt{reqIL}_i^{\texttt{d}} =$ true
$\texttt{now}_i^{\texttt{a}} = t_1$

eff:

$\texttt{reqIL}_i^{\texttt{d}} =$ false

signature

output IL2SM$(m\colon M)_i$
output UPDVIL$(vIL\colon A_{IL}, TXnRT\colon Q, RXnRT\colon Q)_i$
input SENDRES$(atRes\colon bool, rbSt\colon int)$
input RBACK$(ilHT\colon H_{\text{IL}}, clHT\colon H_{\text{CL}}, rbSt\colon int)$
input SM2ILDL$(m\colon M)$
input CL2ILRT$(b_{my}\colon bool, b_{RT}\colon bool, m\colon M)_i$
input AP2ILNRT$(m\colon M)_i$
input IL2APNRT$(q\colon Q)_i$
output IL2CLRT$(m\colon M)_i$
internal UPDATE()$_i$
output REQRT()$_i$

input IL2APNRT$(\texttt{RxnRT}_i^{\texttt{d}})_i$

eff:

set $\texttt{RxnRT}_i$ empty

input AP2ILNRT$(m)_i$

eff:

$\texttt{TxnRT}_i^{\texttt{d}}$.Push$(m)$

output IL2SM$(m)_i$

pre:

$(\texttt{now}_i^{\texttt{a}} = t_2) \wedge \texttt{myIL}_i^{\texttt{d}} \wedge$
$(\neg(\texttt{TxRT}_i^{\texttt{d}}$ empty$) \wedge \texttt{RTIL}_i^{\texttt{d}}) \vee (\neg\texttt{RTIL}_i^{\texttt{d}} \wedge$
$\neg(\texttt{TxnRT}_i^{\texttt{d}}$.Top empty$))$

eff:

if $\texttt{RTIL}_i^{\texttt{d}}$
  set $m = \texttt{TxRT}_i^{\texttt{d}}$
  set $\texttt{TxRT}_i^{\texttt{d}}$ empty
else
  set $m = \texttt{TxnRT}_i^{\texttt{d}}$.Top
  $\texttt{TxnRT}_i^{\texttt{d}}$.Pop
set $m.rbSt = \texttt{checkPT}_i^{\texttt{d}}$
set $m.atRes = \texttt{at}_i^{\texttt{d}}$
$\texttt{myIL}_i^{\texttt{d}} =$ **false**

input SM2ILDL(m)

eff:

if $\texttt{RTIL}_i^{\texttt{d}}$
  $\texttt{RxRT}_i^{\texttt{d}} = m$
else
  $\texttt{RxnRT}_i^{\texttt{d}}$.Push$(m)$

output IL2CLRT$(m)_i$

pre:

$\texttt{now}_i^{\texttt{a}} = t_3$
$\neg(\texttt{RxRT}_i^{\texttt{d}}$ empty$)$

eff:

set $m = \texttt{RxRT}_i^{\texttt{d}}$
set $\texttt{RxRT}_i^{\texttt{d}}$ empty

input $\text{CL2ILRT}(b_1, b_2, m)_i$

eff:

$\texttt{RTIL}_i^{\text{d}} = b_1$

$\texttt{myIL}_i^{\text{d}} = f_{\text{my}}(\texttt{vIL}_i^{\text{d}}, \texttt{RTIL}_i^{\text{d}}, b_2, i)$

$\texttt{TxRT}_i^{\text{d}} = m$

Trajectories T

stop when

$\texttt{now}_i^{\text{a}} = dSlot$

$(\texttt{sendVIL}_i^{\text{d}} = \text{true}) \wedge (\texttt{now}_i^{\text{a}} = t_0)$

$(\texttt{now}_i^{\text{a}} = t_1) \wedge (\texttt{reqIL}_i^{\text{d}} = \text{true})$

$((\texttt{now}_i^{\text{a}} = t_2) \wedge (\texttt{myIL}_i^{\text{d}} = \text{true}) \wedge (\neg(\texttt{TxRT}_i^{\text{d}} \text{ empty}) \wedge \texttt{RTIL}_i^{\text{d}}) \vee (\neg\texttt{RTIL}_i^{\text{d}} \wedge \neg(\texttt{TxnRT}_i^{\text{d}}.\text{Top empty})))$

$\texttt{now}_i^{\text{a}} = t_3 \wedge \neg(\texttt{RxRT}_i^{\text{d}} \text{ empty})$

evolve

$d(\texttt{now}_i^{\text{a}}) = 1$

Şekil 33: Güncel Arayüz Katmanı TIOA Modeli.

3.9.2 Güncel Koordinasyon Katman (CL) Tasarımı

Bu bölümde Koordinasyon Katmani'nda yapilan degisiklikler anlatilmistir. Güncellenen katman modeli Sekil 34'de verilmektedir.

Koordinasyon Katmani islevselligini etkileyen en önemli degisiklik zaman yarigi tipine ve zaman yarigi aidiyet bilgisine bu katman tarafindan karar veriliyor olmasidir. Bu amaçla *REQRT* ve *CL2ILRT* aksiyon tanimlamalarinda güncellemeler yapilmistir.

Katmanin Güvenilirlik Katmani ile arayüzünün kurulabilmesi amaciyla *UPDVCL* ve *RBACK* aksiyonlari eklenmistir. Bu aksiyonlardan *UPDVCL* CL tarafinda tutulan güncel vCL bilgisinin DL katmanina iletilmesinde, *RBACK* aksiyonu ise DL tarafindan karari verilen geri dönme islevine iliskin bilgilerin alinmasinda kullanilmaktadir.

Koordinasyon Katman tasarimini UPPAAL'a tasiyabilmek için analog degiskenler üzerinde yapilan aritmetik islemlerin ortadan kaldirilmasi gerekmistir. Bu islemler ortadana kaldirilarak CL islevselliginin korunabilmesi için saat ilklendirme (clock reset) islemi uygulanmistir.

TIOA $CL_i(del_i\colon int, t_0\colon int, M\colon Type, Q\colon Type, V\colon Type, A_{\mathrm{CL}}\colon Type, H_{\mathrm{CL}}\colon Type, InitCL\colon A_{\mathrm{CL}})$

states

$\mathtt{send}_i^{\mathrm{a}}\colon real := del_i$
$\mathtt{Tx}_i^{\mathrm{d}}\colon V :=$ empty
$\mathtt{Rx}_i^{\mathrm{d}}\colon Q :=$ empty
$\mathtt{RTCL}_i^{\mathrm{d}}\colon bool :=$ false
$\mathtt{myCL}_i^{\mathrm{d}}\colon bool :=$ false
$\mathtt{ch}_i^{\mathrm{d}}\colon int := 0$
$\mathtt{reqCL}_i^{\mathrm{d}}\colon bool :=$ false
$\mathtt{sendvCL}_i^{\mathrm{d}}\colon bool :=$ false
$\mathtt{vCL}_i^{\mathrm{d}}\colon A_{\mathrm{CL}} := InitCL$

signatures

input $\textsc{ap2cl}(m\colon M, ch\colon int)_i$
input $\textsc{il2clrt}(m\colon M)_i$
input $\textsc{reqrt}()_i$
output $\textsc{updvcl}(vCL\colon A_{CL}, Tx\colon V, Rx\colon Q, \mathtt{RTCL}_i^{\mathrm{d}}\colon bool)_i$
input $\textsc{rback}(ilHT\colon H_{\mathrm{IL}}, clHT\colon H_{\mathrm{CL}}, rbSt\colon int)$
output $\textsc{cl2ilrt}(\mathtt{RTCL}_i^{\mathrm{d}}\colon bool, \mathtt{myCL}_i^{\mathrm{d}}\colon bool, m\colon M)_i$
input $\textsc{cl2ap}(q\colon Q)_i$

transitions

input $\textsc{ap2cl}(m, ch)_i$
eff:
$\mathtt{Tx}_i^{\mathrm{d}}[ch].\mathtt{data} := m.dat$
$\mathtt{Tx}_i^{\mathrm{d}}[ch].\mathtt{par} := m.p$

input $\textsc{il2clrt}(m)_i$
eff:
$\mathtt{Rx}_i^{\mathrm{d}}.\mathrm{Push}(m)$
$\mathtt{vCL}_i^{\mathrm{d}} := g_{\mathrm{upd}}(\mathtt{vCL}_i^{\mathrm{d}}, m.\mathtt{par})$

input $\textsc{reqrt}()_i$
eff:
$\mathtt{RTCL}_i^{\mathrm{d}} = g_{RT}(\mathtt{vCL}_i^{\mathrm{d}}, \mathtt{RTCL}_i^{\mathrm{d}})$
$(\mathtt{myCL}_i^{\mathrm{d}}, \mathtt{ch}_i^{\mathrm{d}}) := g_{my}(\mathtt{vCL}_i^{\mathrm{d}}, i)$
$\mathtt{send}_i^{\mathrm{a}} := 0$
$\mathtt{reqCL}_i^{\mathrm{d}} :=$ true
$\mathtt{sendVCL}_i^{\mathrm{d}} :=$ true

output $\textsc{updvcl}$(vCL, Tx, Rx, RTCL)$_i$
pre:
$\mathtt{sendVCL}_i^{\mathrm{d}} =$ true
$\mathtt{send}_i^{\mathrm{a}} = t_0$
eff:
$\mathtt{sendVCL}_i^{\mathrm{d}} :=$ false

input $\textsc{rback}(ilHT, clHT, rbSt)_i$
eff:
$\mathtt{vCL}_i^{\mathrm{d}} := \mathtt{clHT}_i^{\mathrm{d}}.vCL$
$\mathtt{Tx}_i^{\mathrm{d}} := \mathtt{clHT}_i^{\mathrm{d}}.Tx$
$\mathtt{Rx}_i^{\mathrm{d}} := \mathtt{clHT}_i^{\mathrm{d}}.Rx$

output $\textsc{cl2ilrt}(\mathtt{RTCL}_i^{\mathrm{d}}, \mathtt{myCL}_i^{\mathrm{d}}, m)_i$
pre:
$\mathtt{reqCL}_i^{\mathrm{d}} \wedge (\mathtt{send}_i^{\mathrm{a}} = del_i)$
eff:
if $\mathtt{myCL}_i^{\mathrm{d}}$
$m := \mathtt{Tx}_i^{\mathrm{d}}[\mathtt{ch}_i^{\mathrm{d}}]$
$m.vCL := vCL$
set $\mathtt{Tx}_i^{\mathrm{d}}[\mathtt{ch}_i^{\mathrm{d}}]$ empty
else
set m empty
$\mathtt{reqCL}_i^{\mathrm{d}} :=$ false

input $\textsc{cl2ap}(\mathtt{Rx}_i^{\mathrm{d}})_i$
eff:
set $\mathtt{Rx}_i^{\mathrm{d}}$ empty

trajectories

stop when
$(\mathtt{sendVCL}_i^{\mathrm{d}} =$ true$) \wedge (\mathtt{send}_i^{\mathrm{a}} = t_0))$
$\mathtt{reqCL}_i^{\mathrm{d}} \wedge (\mathtt{send}_i^{\mathrm{a}} = del_i)$

evolve
$d(\mathtt{send}_i^{\mathrm{a}}) := 1$

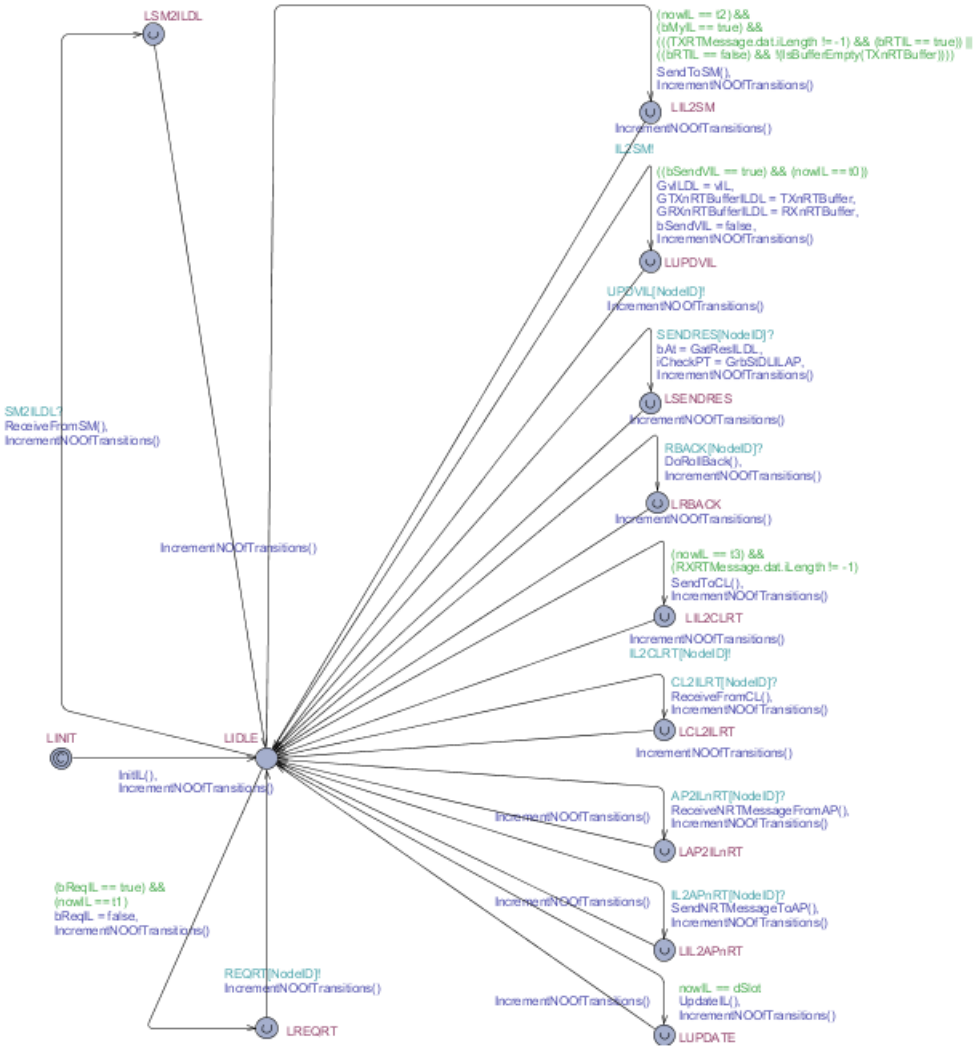
Şekil 34: Güncel Koordinasyon Katmanı TIOA Modeli.

3.10 D^3RIP Doğrulaması

Bölüm 3.9'de anlatılan değişiklikler ve yeni eklenen Güvenilirlik Katmanı sebebi ile D^3RIP iş çerçevesinin var olan gereksinimlerinin doğrulanması ihyiyacı doğmuştur. Bu amaçla UPPAAL aracının *verifyta* motoru kullanılmıştır. Bu bölümde yapılan doğrulama çalışmaları ve doğrulanan gerereksinimler anlatılmaktadır.

3.10.1 Arayüz Katmanı Doğrulaması

Şekil 33'da verilen katman modelinin gereksinimleri sağladığının garanti edilmesi için doğrulama çalışmaları yürütülmüştür. Bu amaçla TIOA modeli *TU_CONVERT* algoritması kullanılarak UPPAAL'a aktarılmıştır. IL'in UPPAAL'da oluşturulmuş davranışsal modeli Şekil 35'de verilmektedir.



Şekil 35: Arayüz Katmanı Davranışsal Modeli

Şekil 35'de verilen model üzerinde UPPAAL *verifyta* motoru işletilerek yapılan doğrulama faaliyetleri ile gereksinimlerin sağlandığı doğrulanmıştır. Doğrulaması yapılan gereksinim örnekleri ve bu gereksinimlerin doğrulanmasında kullanılan TCTL sorgu cümleleri aşağıda verilmektedir.

Iddia 6 Katmanın paylaşımlı ortama (Shared Medium) mesaj gönderebilmesi için (*IL2SM* geçişi ile) zaman slotunun o düğüme ait olması ve zamanın IL konfigürasyonu aşamasında belirlenen t_2 değerine eşit olması gerekmektedir □

Ispat 6 Bu iddianın ispatı için UPPAAL'da Sorgu 6'te belirtilen cümle kullanılmıştır. Verifyta motoru ile bütün durum geçişleri UPPAAL tarafından kontrol edilmiş ve sorgu cümlesindeki iddia ile çelişen geçişler bulunmadığı gösterilerek iddia doğrulanmıştır. □

Sorgu 6 $A[]\ IL.LIL2SM\ imply\ (IL.nowIL == IL.t2)$ and $(IL.myIL == true)$ □

Iddia 7 Katmanın DL'e güncel *vIL* bilgisini gönderebilmesi için (*UPDVIL* geçişi ile), *bSendvIL* değişkeninin ayarlanmış olması ve zamanın IL konfigürasyonu aşamasında belirlenen t_0 değerine eşit olması gerekmektedir □

Ispat 7 Bu iddianın ispatı için UPPAAL'da Sorgu 7'te belirtilen cümle kullanılmıştır. Verifyta motoru ile bütün durum geçişleri UPPAAL tarafından kontrol edilmiş ve sorgu cümlesindeki iddia ile çelişen geçişler bulunmadığı gösterilerek iddia doğrulanmıştır. □

Sorgu 7 $A[]\ IL.LUPDVIL\ imply\ (IL.nowIL == IL.t0)$ and $(IL.bSendvIL == true)$ □

Iddia 8 Katman çalışmasında herhangi bir durum geçişi nedeniyle tetiklenecek bir deadlock durumu yoktur. □

Ispat 8 Bu iddianın ispatı için UPPAAL'da iki düğümlü bir simülasyon ortamı oluşturulmuştur. Simülasyon ortamındaki mesajlaşma her iki düğümdeki bütün katmanların bütün durum geçişlerini tetikleyecek şekilde ayarlanmıştır. (Bu durum simülasyon sonucunda UPPAAL tarafından üretilen kapsama raporu ile doğrulanmıştır.) Hazırlanan simülasyon ortamı üzerinde Sorgu 8'te verilen sorgu cümlesi işletilerek katmanda, hatta D^3RIP tasarımının hiç bir yerinde deadlock yaratacak bir hatanın olmadığı doğrulanmıştır. □

Sorgu 8 $A[]$ not deadlock □

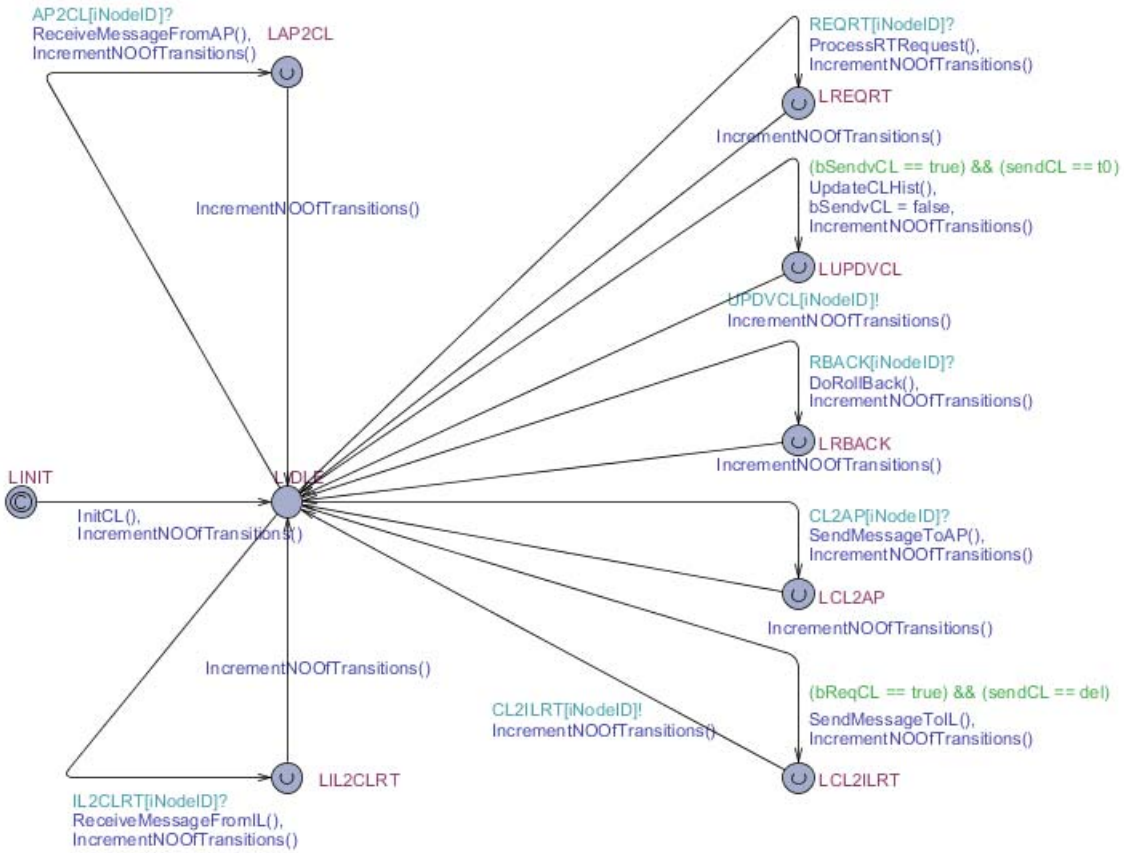
Iddia 9 Katman'da kısıtlı zaman içerisinde sonsuz sayıda durum geçişi yaratarak işleyişin ilerlemesini engelleyecek bir durum yoktur. (IL katmanı Non-Zeno [Gómez (2007)] özelliğe sahiptir). □

Ispat 9 İddia'nın ispatı için IL katmanı UPPAAL modeli içerisine durum geçiş sayacı eklenmiştir. Bu sayaç her bir durum geçişinde artırılmaktadır. Bahsi geçen sayaç kullanılarak, İddia 8 için oluşturulan simülasyon ortamında Sorgu 9'te verilen cümle işletilmiştir. Simülasyon ortamında bütün durum geçişleri kapsanarak IL'in kısıtlı zamanda sonsuz durum geçişi yapmadığı (non-zeno davranışa sahip olduğu) doğrulanmıştır. □

Sorgu 9 $A[]\ IL.LIL2APnRT\ imply\ IL.iNoOfStateTransitions < 26$ □

3.10.2 Koordinasyon Katmanı Doğrulaması

Şekil 34'te verilen katman modelinin gereksinimleri sağladığının garanti edilmesi için doğrulama çalışmaları yürütülmüştür. Bu amaçla TIOA modeli *TU_CONVERT* algoritması kullanılarak UPPAAL'a aktarılmıştır. CL'in UPPAAL'da oluşturulmuş davranışsal modeli Şekil 36'de verilmektedir.



Şekil 36: Koordinasyon Katmanı Davranışsal Modeli

Şekil 36'de verilen model üzerinde UPPAAL *verifyta* motoru işletilerek yapılan doğrulama faaliyetleri ile gereksinimlerin sağlandığı doğrulanmıştır. Doğrulaması yapılan gereksinim örnekleri ve bu gereksinimlerin doğrulanmasında kullanılan TCTL sorgu cümleleri aşağıda verilmektedir.

Iddia 10 Katmanın IL'e mesaj gönderebilmesi (*CL2ILRT* aksiyonu ile) için IL'den istek gelmiş olması ve zamanın CL konfigürasyonu sırasında belirlenen *del* değerine eşit olması gerekmektedir. □

Ispat 10 Bu iddianın ispatı için UPPAAL'da Sorgu 10'te belirtilen cümle kullanılmıştır. Verifyta motoru ile bütün durum geçişleri UPPAAL tarafından kontrol edilmiş ve sorgu cümlesindeki iddia ile çelişen geçişler bulunmadığı gösterilerek iddia doğrulanmıştır. □

Sorgu 10 $A[]\ CL.LCL2ILRT\ imply\ (CL.sendCL == CL.del)$ and $(CL.bReqCL == true)$ □

Iddia 11 Katmanın DL'e güncel *vCL* bilgisini gönderebilmesi için (*UPDVCL* geçişi ile), *bSendvCL* değişkeninin ayarlanmış olması ve zamanın IL konfigürasyonu aşamasında belirlenen t_0 değerine eşit olması gerekmektedir □

Ispat 11 Bu iddianın ispatı için UPPAAL'da Sorgu 11'te belirtilen cümle kullanılmıştır. Verifyta motoru ile bütün durum geçişleri UPPAAL tarafından kontrol edilmiş ve sorgu cümlesindeki iddia ile çelişen geçişler bulunmadığı gösterilerek iddia doğrulanmıştır. □

Sorgu 11 $A[]\, CL.LUPDVCL\ imply\ (CL.sendCL == CL.t0)$ and $(CL.bSendvCL == true)$ □

Iddia 12 Katman çalışmasında herhangi bir durum geçişi nedeniyle tetiklenecek bir deadlock durumu yoktur. □

Ispat 12 Bu iddianın ispatı için UPPAAL'da iki düğümlü bir simülasyon ortamı oluşturulmuştur. Simülasyon ortamındaki mesajlaşma her iki düğümdeki bütün katmanların bütün durum geçişlerini tetikleyecek şekilde ayarlanmıştır. (Bu durum simülasyon sonucunda UPPAAL tarafından üretilen kapsama raporu ile doğrulanmıştır.) Hazırlanan simülasyon ortamı üzerinde Sorgu 12'te verilen sorgu cümlesi işletilerek katmanda, hatta D^3RIP tasarımının hiç bir yerinde deadlock yaratacak bir hatanın olmadığı doğrulanmıştır. □

Sorgu 12 $A[]$ not deadlock □

Iddia 13 Katman'da kısıtlı zaman içerisinde sonsuz sayıda durum geçişi yaratarak işleyişin ilerlemesini engelleyecek bir durum yoktur. (CL katmanı Non-Zeno [Gómez (2007)] özelliğe sahiptir). □

Ispat 13 İddia'nın ispatı için IL katmanı UPPAAL modeli içerisine durum geçiş sayacı eklenmiştir. Bu sayaç her bir durum geçişinde artırılmaktadır. Bahsi geçen sayaç kullanılarak, İddia 12 için oluşturulan simülasyon ortamında Sorgu 13'te verilen cümle işletilmiştir. Simülasyon ortamında bütün durum geçişleri kapsanarak CL'in kısıtlı zamanda sonsuz durum geçişi yapmadığı (non-zeno davranışa sahip olduğu) doğrulanmıştır. □

Sorgu 13 $A[]\, CL.LCL2AP\ imply\ CL.iNoOfStateTransitions < 20$ □

3.11 D^{3}RIP Gerçeklemesi

Bölüm 3.8'da anlatılan işçerçevesi modeline göre geliştirilen Güvenirlik Katmanı (Dependability Layer-DL), Bölüm 3.4'de anlatılan D^{2}RIP başarımı iyileştirilmış versiyon gerçekleştirimine entegre edilmiştir. Bu şekilde elde edilen D^{3}RIP gerçekleştiriminin çalışması aşağıda anlatılmaktadır. DL gerçekleştirimi işletim sisteminde kullanıcı alanında yapılmıştır.

Güvenirlik Katmanı Gerçeklemesi kapsamında protokol yığıtına eklenen fonksiyonlar aşağıda listelenmiştir:

GK için KK'ya Eklenen Fonksiyonlar:
`ListenDependableLayer` işparçacığı: GK'dan gelen mesajları dinler. GK'dan `RBACK` mesajı aldığında `processDL2CLRBACK` fonksiyonunu çağırır.

`processDL2CLRBACK` Fonksiyonu: RBACK (msg,msgLength) fonksiyonunu çağırır.

RBACK Fonksiyonu: Gelen mesajdan vCL parametrelerini çekerek PriorityQueue'ya ekler.

UPDVCL Fonksiyonu: KK'da processCLUPDATE fonksiyonunda ilgili yerlerde çağırılır. O anki vCL parametrelerini GK'ya Message Queue kullanarak gönderir.

GK Tasarımı İçin AK'ya Eklenen Fonksiyonlar:
UPDVIL Fonksiyonu: O anki vIL parametrelerini GK'ya charDev_IL2DL dosyasını kullanarak gönderir.

RBACK Fonksiyonu: charDev_DL2IL dosyası kullanılarak gelen mesajdan vIL parametrelerini çekerek kendi vIL parametrelerine atar.

GK Fonksiyonları:
ListenCoordinationLayer işparçacığı: KK'dan gelen mesajları dinler. KK'dan UPDVCL mesajı aldığında processCL2DLUPDVCL fonksiyonunu çağırır.

processCL2DLUPDVCL Fonksiyonu: UPDVCL (msg,msgLength) fonksiyonunu çağırır.

UPDVCL Fonksiyonu: Gelen mesajdan vCL parametrelerini çekerek vCLQueue'ya ekler.

ListenInterfaceLayer Threadi: AK'dan gelen mesajları dinler. AK'dan UPDVIL mesajı aldığında processIL2DLUPDVIL fonksiyonunu, SM2ILDL mesajı aldığında processSM2IL fonksiyonunu çağırır.

processIL2DLUPDVIL Fonksiyonunu: UPDVIL (msg,msgLength) fonksiyonunu çağırır.

UPDVIL Fonksiyonu: Gelen mesajdan vIL parametrelerini çekerek vILQueue'ya ekler.

processSM2IL Fonksiyonu: Gelen mesajdan atRes,rbSt(checkPt),vCL parametrelerini çekerek ilgili değişkenlere atar. atRes sonucuna doğruysa fAT fonksiyonunu çağırarak ATEST yapıp, sonuçları dosyaya yazarak AK'ya gönderir. Sonuçları göndererek SENDRES yapmış olur. Eğer atRes sonucu yanlışsa, RBACKCL ve RBACKIL fonksiyonlarını çağırır.

fAT: vCL Queue'dan çektiği vCL parametreleri ile AK'dan gelen vCL parametrelerini kıyaslar. Eşitse doğru, eşit değilse yanlış döner. Rapor içerisinde Figür7'de testi görülür.

RBACKCL Fonksiyonu: Hata durumunda, vCLQueue'dan eski vCL parametreleri çekilerek KK'ya gönderilir.

RBACKIL Fonksiyonu: Hata durumunda, vILQueue'dan eski vIL parametreleri çekilerek AK'ya gönderilir.

Koordinasyon Katmanı
KK, Uygulama Katmanından (AP) bir RT mesaj gelip gelmediğini öğrenmek için runCL işparçası içerisinde mMQueueToReceive mesaj kuyruğunu sürekli sorgular, bir RT mesaj gelince, gelen RT mesajı KK'da bulunan TxQueue kuyruğuna kopyalar. Bunun ardından zaman diliminin tipini (RTCL) ve sahibini (myCL) belirlemek için processCLUpdate fonksiyonunda bulunan gRealTime ve gMySlot fonksiyonlarını çağırır. Zaman dilimi tipi ve sahibi belirlendikten sonra;

- Zaman dilimi tipi RT ve zaman dilimi sahibi bu düğüm ise gönderilen mesaj kuyruğu TxQueue'dan mesaj çıkarılır. CL_PACKET başlığındaki zaman dilimi sayısı ve zaman bilgisi güncellenir. UPDVCL fonksiyonu çağrılarak o anki zaman diliminin KK protokol durum değişkeni olan vCL parametreleri Güvenirlilik Katmanına (GK) bir işletim sistemi mesaj kuyruğu yapısı kullanılarak gönderilir. KK, paketi AK'nına gönderir. Eğer

TxQueue'dan mesaj pop edilerek çekilemez ise yani gönderilecek RT paket hazır değilse, KK başlığındaki zaman dilimi sayısı ve zaman bilgisi güncellenir ve CL_PACKET başlığı AK'ya gönderilir.

- Zaman dilimi tipiRT ve sahibi başka bir düğüm ise, UPDVCL fonksiyonu çağrılarak o zaman dilimine ait vCL parametreleri Güvenirlilik Katmanına (GK) mesaj kuyruğu veri yapısı üzerinden gönderilir. AK'ya da 1 Byte zaman dilimi tipi bilgisi yollanır.

- Slot tipi nRT ise UPDVCL fonksiyonu çağrılarak o anki slotun vCL parametreleri Güvenirlilik Katmanına (GK) message queue kullanılarak gönderilir. AK'ya da 1 Byte zaman dilimi tipi bilgisi yollanır.

Arayüz Katmanı
KK'dan AK'ya gelen mesajın uzunluğu:

- 1 Byte ise, gönderilen mesajın içeriğine göre RTIL ve myIL değerleri belirlenir.

- 1 Byte değilse, gelen paketin boyu,mesajın içerisindeki mesaj boyutu ile aynıysa, mesaj RTTxData Bufferına kopyalanır. RTIL ve myIL değerleri true olarak belirlenir. Sonrasında, UPDVIL fonksiyonuyla o anki vIL değerleri GK'ya gönderilir.

UPDVIL fonksiyonundan sonra IL2SM fonksiyonu çağrılır. Eğer slot o düğüme aitse;

- Zaman dilimi RT ise, GK'dan SENDRES ile gelen atRes ve checkPt parametreleri RTTxData'da bulunan mesajın içerisine kopyalanır. Sonrasında RTTxData'da RT paketi göndermek için Ethernet sürücüsünde bulunan transmit fonksiyonu çağırılır. Mesaj gönderildikten sonra loopback durumunda, GK'dan gelen atRes ve checkPt parametreleri ile KK'dan gelen vCL parametreleri, GK'ya gönderilir. RTTxData'da bulunan RT paketi ise RTRxData'ya kopyalanarak KK'ya gönderilir.

- Zaman dilimi nRT ise yüksek öncelikli nRT paketi sırada varsa o gönderilir, yoksa düşük öncelikli nRT paketinin olup olmadığı kontrol edilerek, sırada olması durumunda o paket ethernet sürücüsündeki transmit fonksiyonu çağırılarak SM'a gönderilir.

Güvenirlik Katmanı
RT mesajı gönderilmesi sırasında, KK'dan GK'ya UPDVCL mesajıyla, vCL parametreleri gönderilir. GK sürekli olarak KK'yı bir işparçacığı ile dinleyerek, UPDVCL mesajının gelmesini bekler. Gelen vCL parametreleri daha sonra GK'da vCLQueue'da tutulur. Aynı şekilde AK'dan da GK'ya UPDVIL mesajıyla, vIL parametreleri gönderilir. GK sürekli olarak AK'yı bir iş parçacığı ile dinleyerek, UPDVIL mesajının gelmesini bekler. Gelen vIL parametreleri daha sonra GK'da vILQueue'da tutulur.

RT mesaj AK'dan SM'e gönderildikten sonra, loopback ile mesaj tekrar KK'ya gönderilir. Gönderilme sırasında AK'dan GK'ya SM2ILDL mesajıyla, atRes, checkPt ve vCL parametreleri gönderilir. GK sürekli olarak AK'yı bir işparçacığı ile dinleyerek, SM2ILDL mesajının gelmesini bekler. Mesaj geldikten sonra mesajda bulunan atRes parametresine göre RBACK veya ATEST yapılır.

- Eğer mesajda bulunan atRes parametresi 1 (yani hata yoksa) ise ATEST yapılır. Eğer o anki slot RT ise SM2ILDL ile gelen parametreler ile vCLQueue'dan çekilen vCL parametreleri fAT fonksiyonunda karşılaştırılır. Eğer parametreler eşit ise fAT = true olarak döner, atRes= 1 olur. Eğer parametreler eşit değil ise fAT false olarak döner, atRes=0 olur. Eğer o anki slot nRT ise o anki slotun kime ait olduğunu tutan vDL değişkeniyle SM2ILDL ile gelen r.vCL parametresindeki o anki slot kime ait olduğunu tutan nodeId değişkeni karşılaştırılır. Eğer parametreler eşit ise fAT= true olarak döner, atRes=1 olur. Eğer parametreler eşit değil ise fAT=false olarak döner, atRes=0 olur. Test sonucu belirlenen atRes ve rbSt=checkPt SENDRES mesajıyla AK'ya gönderilir.

- Eğer mesajda bulunan atRes=0 ise (yani hata varsa) RBACK yapılır. Rollback durumunda, sürekli olarak tutulan vCL ve vILQueue'lerinden hata öncesi çalışmış bir önceki vCL ve vIL parametreleri çekilerek AK ve KK'ya gönderilir.

Arayüz Katmanı

RT Mesaj SM'e gönderildikten sonra, karşı düğüm SM'den gelen mesajı AK'da bulunan receive fonksiyonuyla alır. Gelen mesaj paketi RTRxData'ya kopyalanır. Daha sonra mesaj içerisinde bulunan atRes,checkPt ve vCL parametreleri RTRxBufferDL'e atandıktan sonra SM2ILDL mesajıyla GK'ya gönderilir. Mesaj GK'ya gönderildikten sonra RTRxData bufferında bulunan mesaj paketi KK'ya gönderilir.

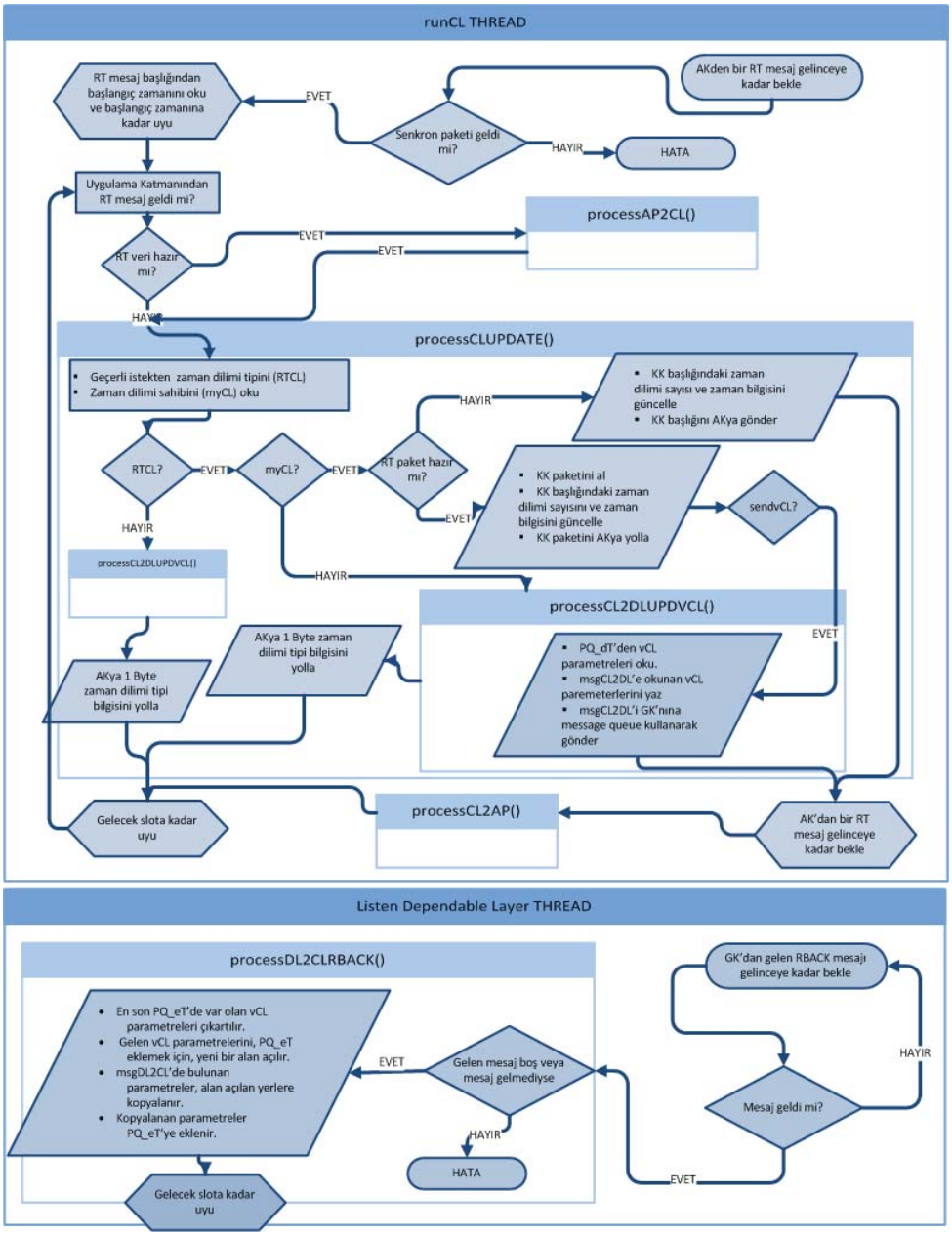
Koordinasyon Katmanı

KK, AK'dan gelen RT mesajı alarak, paket başlığını ayırır. Eğer gelen mesajın boyutu ile mesaj içerisindeki mesaj boyutu aynı ise gUpdate fonksiyonu çağrılır. Mesaj içerisindeki vCL parametreleri çekilerek, KK'da bulunan Priority Queue'ya (PQ) eklenir. Böylelikle bir sonraki slotun vCL parametreleri belirlenmiş olur. Daha sonra mesaj içeriği Uygulama Katmanına(AP) gönderilir.

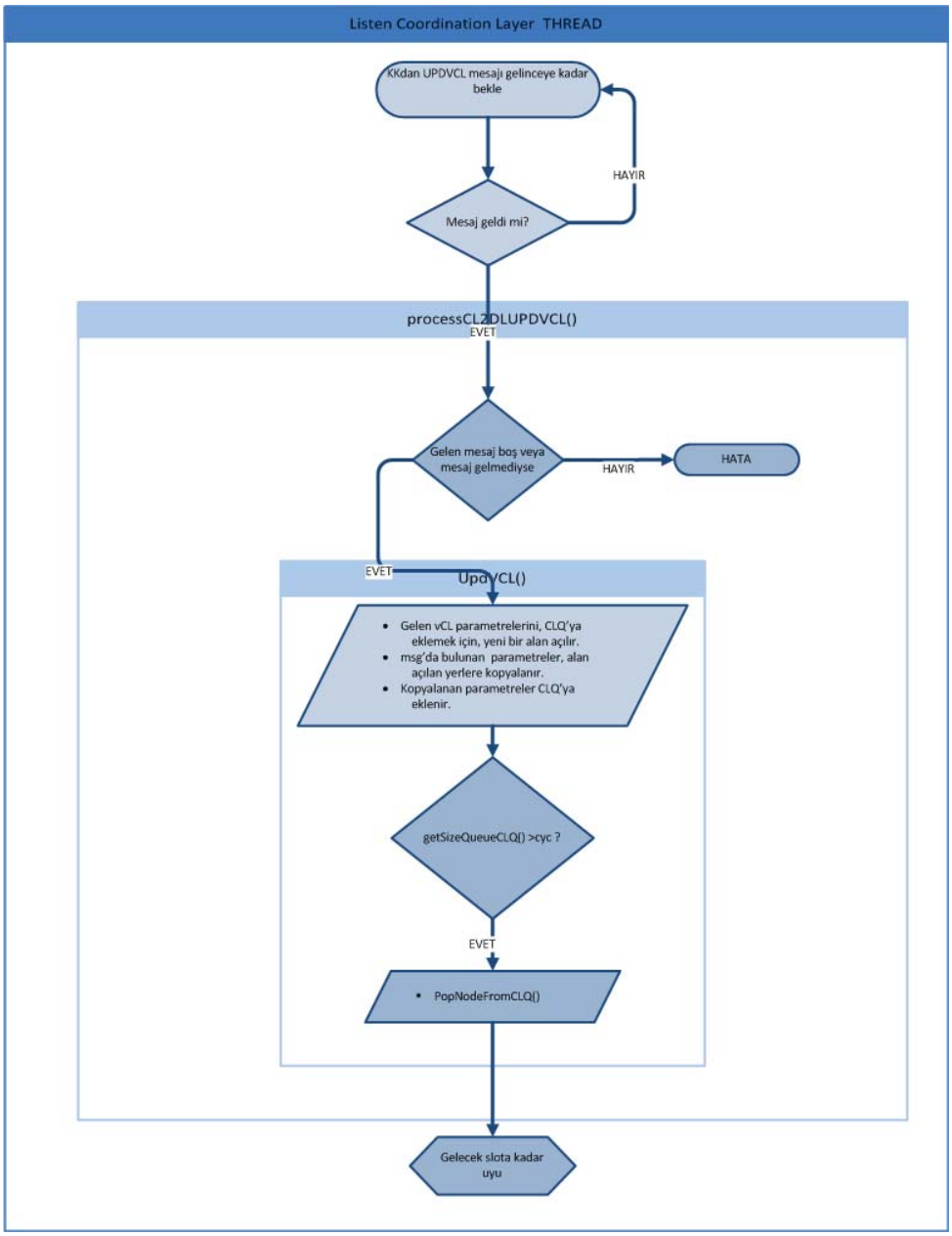
Güvenirlik Katmanı

GK'da ise SM2ILDL mesajıyla gelen parametreler ile UPDVCL ile gelen parametreler karşılaştırılır ve duruma göre RBACK ve ATEST durumlarına karar verilir.

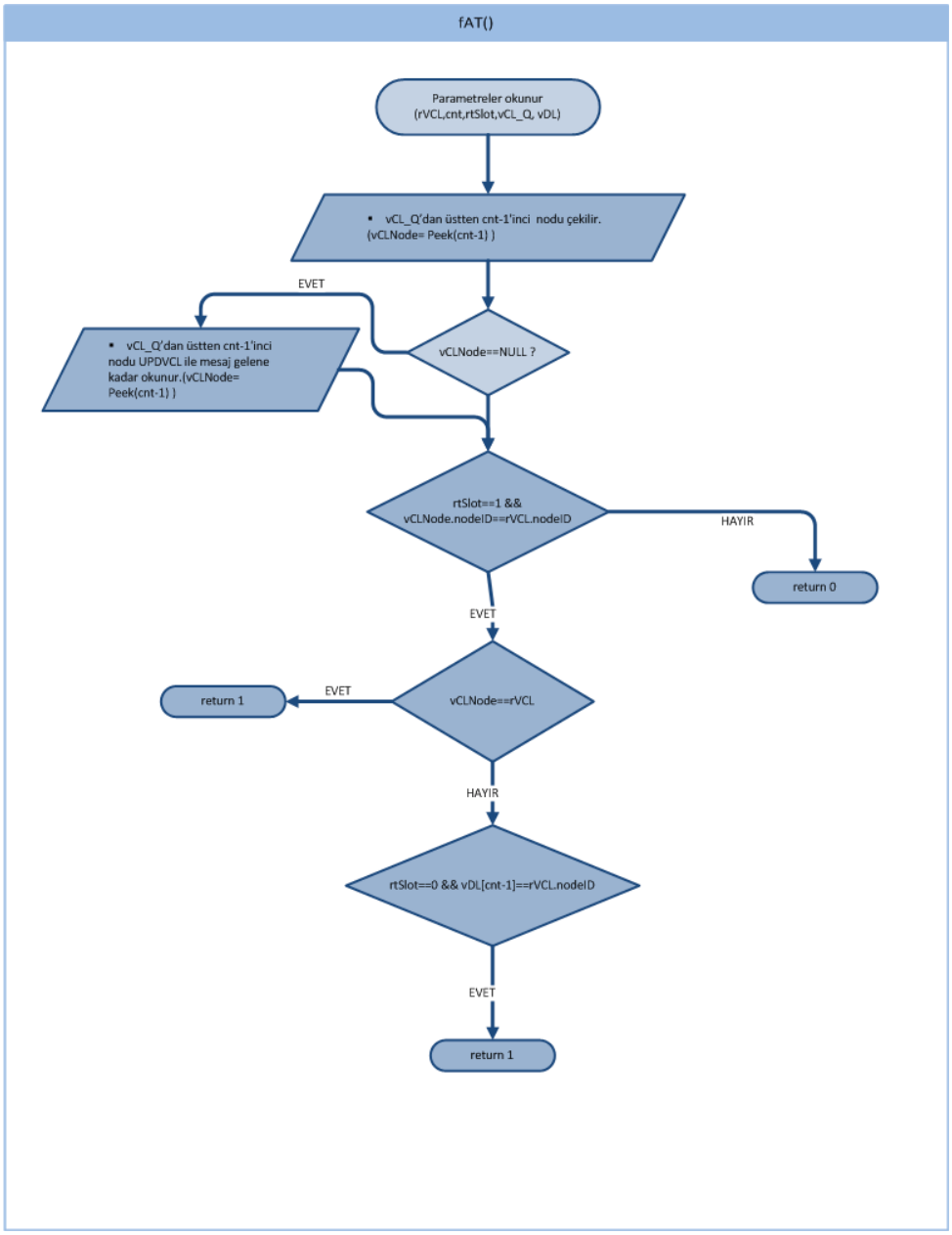
Yukarıda işleyişi özetlenen Güvenirlik Katmanının işparçaları ve modüllerinin detaylı çalışması aşağıdaki akış diyagramlarında gösterilmiştir.



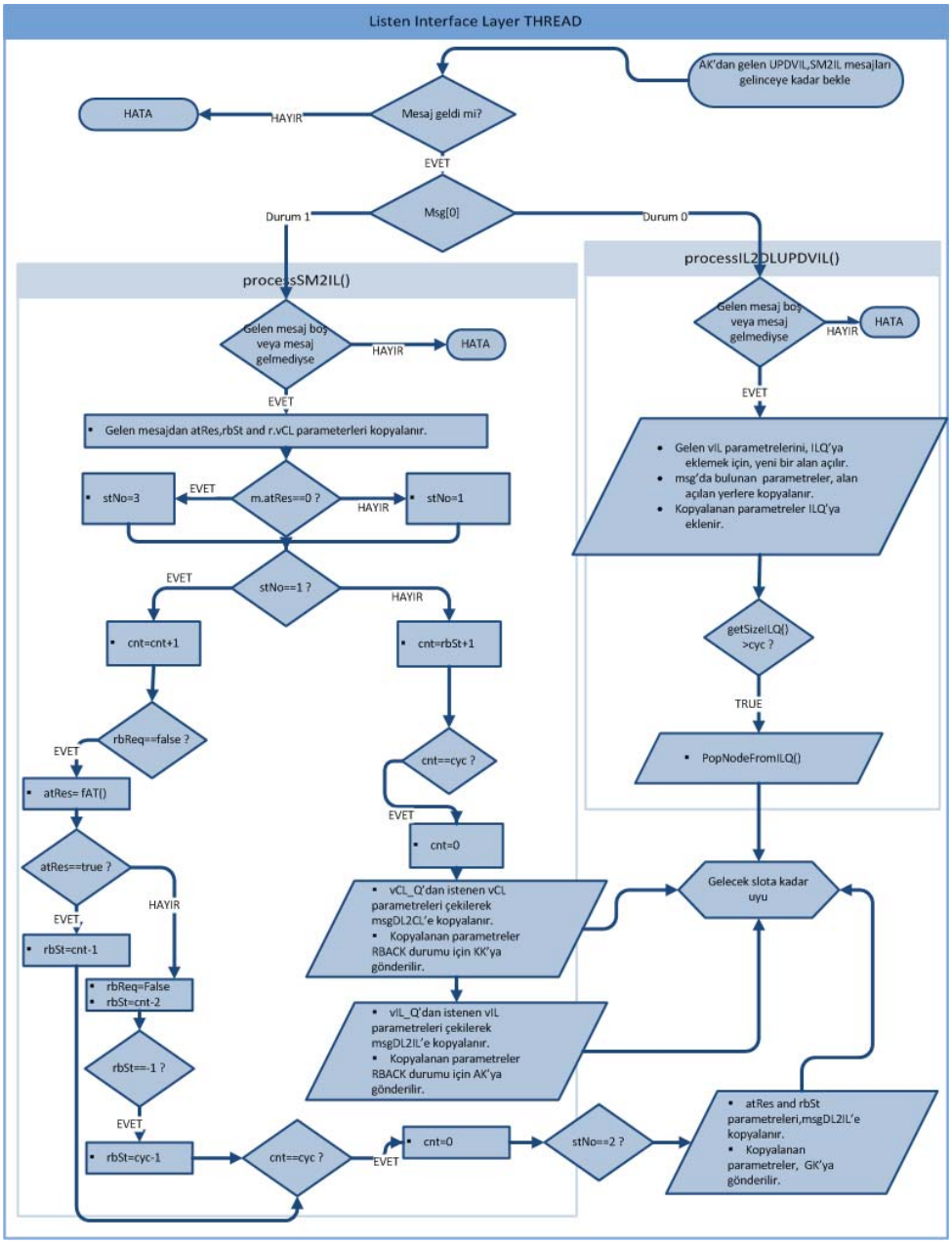
Şekil 37: Güvenirlik-Koordinasyon Katmanları Akış Diyagramı.



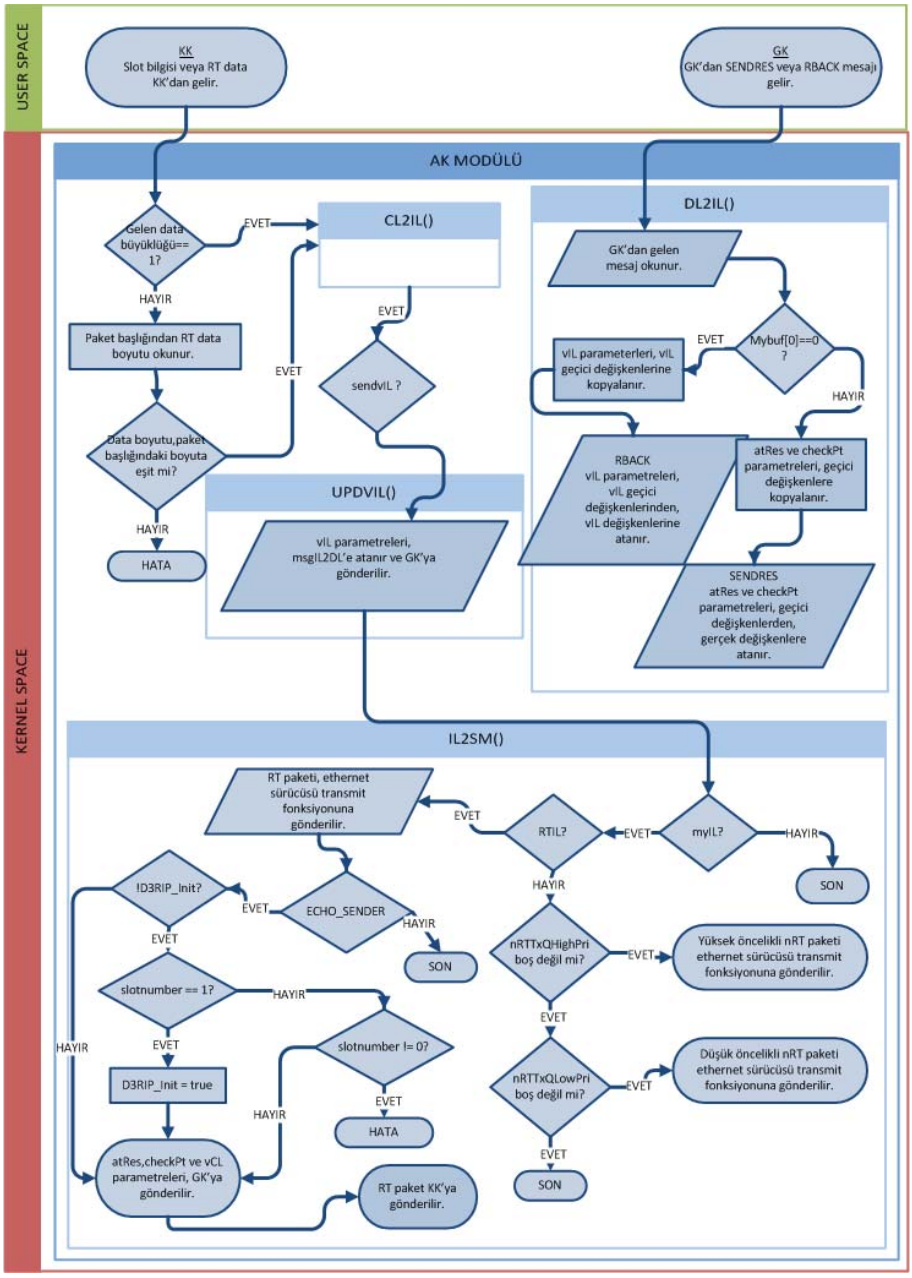
Şekil 38: Güvenirlik-Koordinasyon Katmanları Durum Değişkeni Güncellemesi Akış Diyagramı.



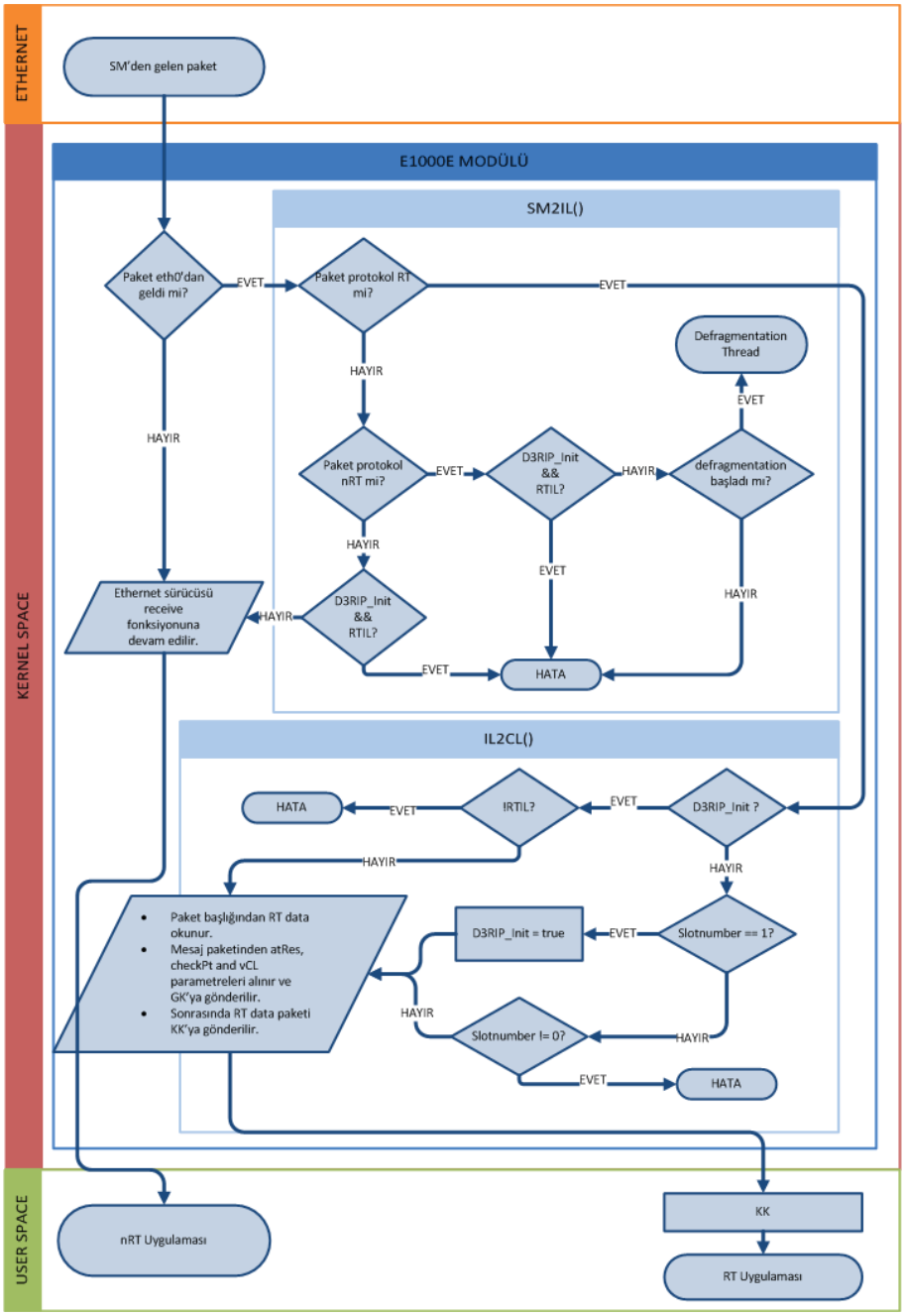
Şekil 39: Güvenirlik Katmanı Kabul Testi Akış Diyagramı.



Şekil 40: Güvenirlik Katmanı-Arayüz Katmanı Durum Değişkeni Güncellemesi Akış Diyagramı.



Şekil 41: Güvenirlik Katmanı-Arayüz Katmanı Mesaj Gönerimi Akış Diyagramı.



Şekil 42: Güvenirlik Katmanı-Arayüz Katmanı Mesaj Alımı Akış Diyagramı.

4 Bulgular

Tasarlanan ve gerçekleştirilen Koordinasyon ve Arayüz Katmanlarından oluşan ve Bölüm 3.4'de detaylı olarak anlatılan D^{2}RIP protokol yığıtının son sürümü ve bu sürüme Güvenirlik Katmanının eklenmesiyle elde edilen D^{3}RIP yığıtının başarımı labaratuvar ortamında Bölüm 3.6'daki örnek uygulamalar ile incelenmiştir. Gelişme raporlarında ilk sürüm gerçeklemeler küçük bir örnek üzerinde denenmiş ve sonuçlar sunulmuştur.

4.1 D^{2}RIP ve D^{3}RIP Deneysel İnceleme Ortamı

Bölüm 3.6'daki örnek uyarınca bir robot kol (R), iki yönde işlenen ürünleri taşıyan bir taşıma bandı (C) ve ürünleri boyayan bir boyama aracı (PD)ndan oluşan bir kontrol sistemi (Plant) ve her bileşeni kontrol eden kontrolcu cihazları (Controller R, Controller C, Controller PD) ile bu cihazları koordine eden üst düzey kontrolcu Controller S'den oluşan bir deney ortamı oluşturulmuştur.

Bu deney düzeneği seçilirken Bölüm 3.6'da sıralanan sebeplerden dolayı bu bölümde anlatılan örnek uygulamanın gerçeklenmesine karar verilmiştir. Buna ek olarak Projeden önce labaratuvarımızda bulunan ve deneylerde kullanılması planlanan gömülü platformların eskimesi ve çalışmamaya başlamaları gibi bazı donanım kısıtları nedeni ile de düğüm sayısı 4 kontrolcu ve bir plant olarak belirlenmiştir. Yeni bir örnek geliştirmek, testlerini ve analitik incelemesini yapmak için gerekecek zaman uzun bir zamandır ve bu zamanı hazır örnek üzerinde yığıtın zaman başatımını optimize etmek için kullanma tercihi yapılmıştır.

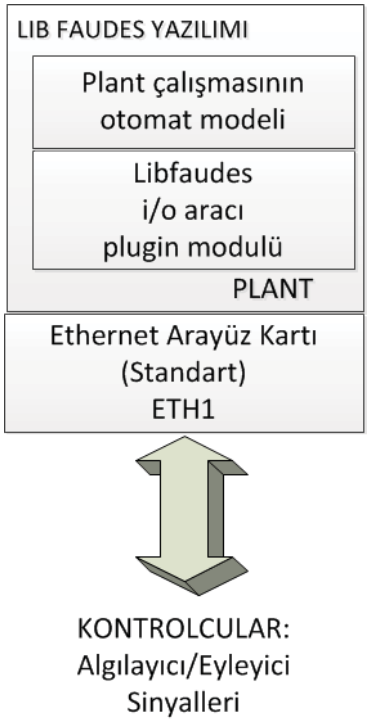
Bu ortamın iki alt konfigürasyonu bulunmaktadır. İlk konfigürasyonda bütün kontrolcu cihazları masaüstü bilgisayarlar üzerinde gerçeklenmiştir. İkinci konfigürasyonda ise iki kontrolcu cihazı WAGO PLC üzerinde, iki kontrolcu cihazı da masaüstü bilgisayar üzerinde gerçeklenmiştir. Plant gömülü platform biigisayarında gerçeklenmiştir. Örnekte tanımlanan plant mekanik hareketli parçalardan oluşmaktadır. Buna uygun olarak plant daha yavaş çalışmaktadır.

Kullanılan cihazların teknik özellikleri aşağıda sıralanmıştır:

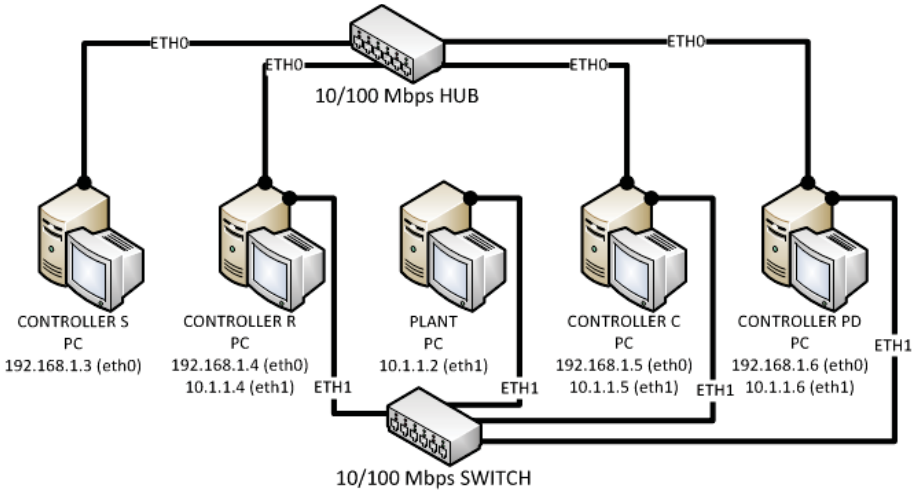
1. Kontrolcu Bilgisayarlar: QuadCore Intel(R) Core(TM) i3 CPU 550@3.20GHz, 4 GByte RAM. Donanım zaman damgalı IEEE 1588 Ethernet arayüz kartı.
2. Plant Bilgisayar: CPU Z530 @ 1.60GHz. Standart Ethernet arayüz kartı./QuadCore Intel(R) Core(TM) i3 CPU 550@3.20GHz, 4 GByte RAM. Donanım zaman damgalı IEEE 1588 Ethernet arayüz kartı.
3. WAGO PLC: Celeron 600, 256 MByte RAM, 2 x USB, RS-232, DVI-I and 2 x ETHERNET. Gömülü standart Ethernet arayüz kartı.

Plant Erlangen Üniversitesi'ndeki araştırma grubundan alınan libfaudes simulatör aracı ile gerçeklenmiştir. Bu simülatörde Plant'deki bileşenlerin ayrık olaylı sistem modelleri koşmaktadır. Bu modellerde durum değişiklikleri, Plantdeki algılayıcılardan gelen sinyaller ve kontrolculardan gelen eyleyici sinyalleri ile olmaktadır. Bu sinyallerin kontrolcular ile alşverişi Erlangen üniversitesinde geliştirilen libfaudes `io_plugin` modülü ile olmaktadır. Bu sinyaller kontrolcular ile Plant bileşenleri arasında Şekil2'de görülen T1 düzeyinde haberleşmedir. Bu sinyallerin iletimi için her kontrolcu cihazına ikinci bir Ethernet kartı takılmıştır ve D^{2}RIP çalışan

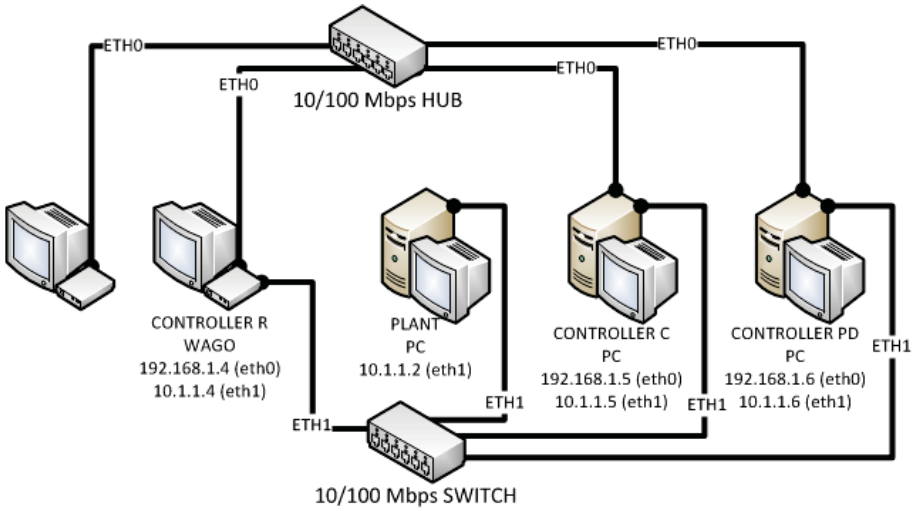
arayüzden farklı bir arayüzden haberleşilmektedir. Deneyler boyunca bu sinyal lışverişi için kullanılan Ethernet arayüzü `ETH1` olarak adlandırılacaktır. Bu arayüzler birbirlerine bir Ethernet switch ile noktadan noktaya bağlıdır. Bu ikinci ağ kontrolcu haberleşmesinin yapıldığı ve D^2RIP/D^3RIP koşan ağdan tamamen ayrı bir ağdır. Bu şekilde gerçek bir Plant'deki kontrolcu ile Plant'deki ilgili cihaz arasında doğrudan algılayıcı /eyleyici sinyal iletişimi varsayımı gerçeklenmiştir. Plant bilgisayarının yapısı Şekil 43'de Kurulan deney düzenekleri ise Şekil44 ve Şekil 45'de görülmektedir. Şekil de Konfigürasyon 1'de kontrolcular masaüstü bilgisayarlarla gerçeklenirken, Şekil Konfigürasyon 2'de iki kontrolcu WAGO PLC ile gerçeklenmiştir. Kontrolcuların arasında D^2RIP/D^3RIP protokol haberleşmesi `ETH0` Ethernet Arayüzü ile ve paylaşımlı ortam iletişimi sağlayan hub cihazı üzerinden yapılmaktadır. Controller S cihazı diğer kontrolcuları koordine eden üst seviye kontrolcudur. Bu sebeple Plant ile bağlantısı yoktur.



Şekil 43: Plant gerçeklemesi.

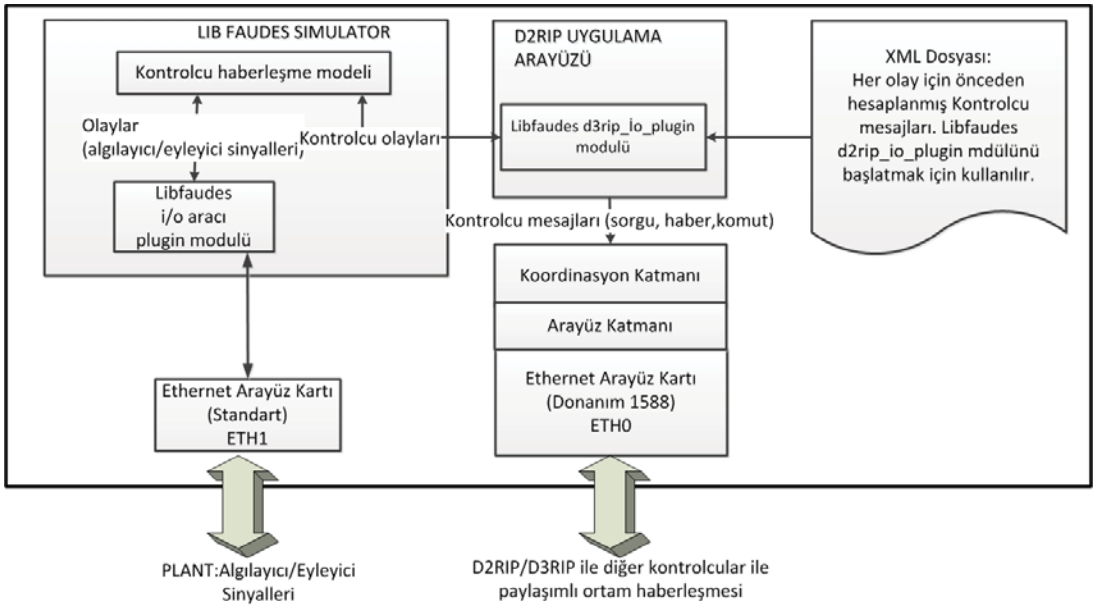


Şekil 44: Deney düzeneği. Konfigürasyon 1: Kontrolcular PC ile gerçeklenmiş.



Şekil 45: Deney düzeneği. Konfigürasyon 2: Kontrolcular PC ve PLC ile gerçeklenmiş.

Kontrolcuların gerçeklenmesi Şekil 46'de görülmektedir. Kontrolcunun ayrık olaylı çalışma modeli libafudes simulatörü ile gerçeklenmektedir. Bu simülatörden gelen kontrolcu olayları libfaudes `d3rip_io_plugin` modülü ile D^2RIP/D^3RIP üzerinde koşan RT kontrolcu uygulaması mesajlarına dönüşmektedir. Bu mesajlar `d3rip_io_plugin` modülünde yer alan bir tablodan her olay için karşılık gelen mesaj bulunarak oluşturulmaktadır. Bu tablo sistem çalışmaya başlarken `d3rip_io_plugin` modülü ilklendirilmesi sırasında bir XML dosyasından alınmaktadır.



Şekil 46: Kontrolcu gerçeklemesi.

Yapılan deneylerde Bölüm 3.5'da tanımlanan (5) ifadesindeki etken bant genişliğini yükseltmek ve (3) ifadesine göre desteklenebilecek son gönderilme zamanları için sıkı bir değer bulabilmek için, olabildiğince $dSlot$ küçük ve L büyütülmüştür. Buna göre bütün deneylerde $dSlot = 250\mu s$ ve $Frame_{max} = 504$ Byte seçilmiştir. Buna göre Gönderilebilecek en büyük RT verisi $L = Frame_{max} - ((14+8+4)+(4+2)) = 472$ Byte olmuş ve $B_{\text{eff}} = \frac{L}{dSlot} = \frac{3776\text{bit}}{250\mu\text{sn}} = 15.1$Mbps olmuştur. Bölüm 3.4.4'de de belirtildiği gibi RT paketlerde Ethernet çerçevesi hep $Frame_{max}$ uzunluğunda gönderilip RT verinin arkası doldurulmaktadır (padding).

Yapılan deneylerde ölçümlerde elde edilen ortalama değerler için ayrıca %99'luk *güvenirlik aralığı* bulunmuştur. Başka bir deyişle deneylerde ölçülen ve tabloda *ortalama* olarak gösterilen örnek ortalaması %99 olasılıkla gerçek ortalama μ'nün $\% \pm \Delta$ komşuluğunda bulunmaktadır. Buna ek olarak deney boyunca görülen en düşük ve en yüksek değer de görülmektedir.

4.2 D^{2}RIP Zaman Analizi

4.2.1 Eşzamanlama Hassasiyeti Analizi

Kullandığımız IEEE 1588 için donanımda zaman damgası alan Ethernet arayüz kartlarıyla elde ettiğimiz eşzamanlama hassasiyeti Tablo 6'da görülebilir. Alınan ölçüm sayısı 2048 örnektir. Ölçü birimi μsdir.

Burada önemli bir nokta, IEEE 1588 donanım zaman damgası destekleyen kartın bilgisayarlara takıllabildiğidir. WAGO PLC cihazları gömülü cihazlardır ve bu kart onlara takılamamaktadır. Bu sebeple Şekil 45'deki düzenekte 2 WAGO cihazı yazılım ile eş zamanlama yapmaktadır. Bu eşzamanlamanın kendi başına hassasiyeti Tablo 6'da görüldüğü gibi düşüktür. Buna karşılık Şekil 45'da WAGO PLC ve bilgisayarlar birlikte kullanılmaktadır. Bilgisayarlardaki donanım zaman damgalaması sayesinde sistemin genel hassasiyeti sırf yazılım ile eşzamanlamaya göre daha iyi olmaktadır.

	Ortalama	$\%\pm\Delta$	Max.	Min.
İki düğüm arasındaki zaman farkı (Donanım)	1.4	%3.71	4.2	0
İki düğüm arasındaki zaman farkı (Yazılım)	105.3	%2.13	9.4	154.7
İki düğüm arasındaki yol gecikmesi (Donanım)	1.6	%0	1.7	1.6

Tablo 6: IEEE 1588 eşzamanlama hassasiyeti.

4.2.2 Fonksiyon Zamanlamaları

Bölüm 3.4.4'de açıklanan protokol yığıtı paket yapıları Şekil 14'de gösterilmiştir. Bu paket yapıları protokolun işlemesi için gerekli başlık ve alanları içermektedir. Bunlara ek olarak aşağıda tartışılan zamanlama değerlerini ölçmek için bu veri yapılarına ek zaman alanlanları kullanılmıştır. Paket gönderilme ve alınma sırasında bu alanlara zaman damgaları yazılmış ve daha sonra gönderilip alınması tamamlanan paketler üzerindeki bu zaman damgalarından yararlanarak fonksiyon bileşenlerinin zamanları ölçülmüştür.

Yukarıda anlatılan deney ortamında Bölüm 3.4.9'de Şekiler: 22, 23, ve 24'de gösterilen D^{2}RIP fonksiyonlarının zamanlama diyagramlarındaki bileşenler ölçülmüştür. Değerler Tablo 7'da görülmektedir. Bu tabloda son satır yapılan bütün işlerin tamamlandığı zamanı göstermektedir. Bu işlevler için harcanan toplam zamana 504 Bytelık Ethernet çerçevesinin 100Mbps hızda gönderilmesi için gerekli 40μs gecikme de ilave edilmelidir. Zamanlar μs cinsindendir. Toplam 5 milyon örnek alınmıştır. Sonuçlar örnek alınırken sürekli güncellenen istatistikler olarak toplanmıştır.

Fonksiyon	Ortalama	$\%\pm\Delta$	Max.	Min.
KK İşparçacığı Uyanması	0.7	%0.21	23.5	7
`ap2cl`	3.2	%3.23	6.1	2.3
`clupdate`	1.3	%0.45	6.5	0.7
`il2sm`	15.4	%0.38	27.9	9.8
`sm2il`	24.3	%0.47	63.2	13.4
`il2cl`	9.2	%1.22	46.1	1
`cl2ap`	2.3	%4.07	12.8	4.9
Zaman dilimi işlerinin tamamlanması	94.1	%1.82	231.1	4.1

Tablo 7: Bir zaman diliminde yapılan iş bileşenlerinin ölçülen zamanları.

Yukarıdaki sonuçlar Şekil 44'da görülen kontrolcuların bilgisayarla gerçekleştirildiği düzenekte alınmıştır. Burada bilgisayarlarda donanım destekli IEEE 1588 barındıran Ethernet Kartları kullanılmıştır.

Geliştirilen protokol yığıtı WAGO PLC cihazında sorunsuz çalışmaktadır. Şekil 45'da görülen şekilde yukarıdaki deney iki kontrolcu cihazı WAGO PLClerle gerçekleştirilerek tekrarlandığında $dSlot = 400\mu$sn olmuş ve aşağıdaki sonuçlar ölçülmüştür:

4.3 D^{2}RIP Başarım İncelemesi

D^{2}RIP protokol başarımı deneysel çalışmayla incelenmiştir. Bu çalışmada Bölüm 3.6'deki kontrol uygulaması örneği üzerinde üç farklı deney yapılmıştır. Deney 1'de kontrol uygulaması ve

Fonksiyon	Ortalama	$\%\pm\Delta$	Max.	Min.
KK İsparçacığı Uyanması	25.2	%2.1	75	10.1
ap2cl	8.9	%4.2	15.1	3.9
clupdate	5.1	%1.3	11.9	3
il2sm	55.1	%1.9	70.3	45.2
sm2il	61.8	%1.7	92.1	54.8
il2cl	15.1	%2.3	64.8	3.1
cl2ap	7.9	%6.1	17.7	5.3
Zaman dilimi işlerinin tamamlanması	195.5	%3.1	395.4	145.2

Tablo 8: Bir zaman diliminde yapılan iş bileşenlerinin ölçülen zamanları. Deney Konfigürasyonu 2.

RT trafik detaylı olarak incelenmiş ve doğrulanmıştır. Deney 2'de, kanal 1'den çalışan örnek kontrol uygulamasına ek olarak kanal 2'de çalışan ek bir RT uygulaması çalıştırılmış, bu durumda örnek kontrol uygulamasının başarımı incelenmiştir. Deney 3'de ise, by kontrol uygulaması çalışırken farklı ek nRT yükleri oluşturulmuş ve bu yükler altında nRT trafik başarımı incelenmiştir.

Deney 1: D^2RIP gerçek zamanlı desteğinin detaylı incelenmesi
Bu deneyde öncelikle Bölüm 3.6'deki örnek kontrol uygulamasının doğru çalıştığı gözlemlenmiştir. Bütün kontrolcu olayları ve haberleşmesi ayrık olaylı sistem modellerine uygun olarak gerçekleşmiştir.

Örnek uygulamada Tablo 5'de sıralandığı gibi her biri sorgu, haber ve komuttan oluşan üçlü set halinde 10 set mesaj bulunmaktadır. Bu mesajların gönderildiği farklı kontrolcu düğümü çiftleri üzerinde uçtan uca farklı katmanlarda gecikme zamanları olçülmüştür. Bu ölçümler $Frame_{max} = 104$ Byte değeri için de tekrarlanmıştır. Veriler 1080 adet paket gönderimi üzerinden hesaplanmıştır. Ölçüm birimi μ saniyedir.

Tablo 9 ve benzer tablolarda görülen uygulamadan uygulamaya gecikmeler RT kontrolcu mesajlarının geçerli olmak için bekledikleri zamanı da içermektedir(elibility time). Dolayısı ile bu gecikmenin büyük bir kısmı uygulamanın çalışma şeklindendir. Mesaj geçerli olduktan sonra karşı tarafa gönderilme zamanı bir zaman diliminin (250μs) altındadır.

Ölçüm	Ortalama	$\%\pm\Delta$	Max.	Min.
Uygulamadan uygulamaya	3473.1	%1.96	4423	221
2 düğüm KK arası	89.1	%5.60	142	16
2 düğüm AK arası	63.2	%6	108	32

Tablo 9: Uçtan uca farklı katmanlarda gecikmeler.

Ölçüm	Ortalama	$\%\pm\Delta$	Max.	Min.
Uygulamadan uygulamaya	3237.2	%2.79	4229	129
2 düğüm KK arası	61.2	%5.38	108	12
2 düğüm AK arası	61.1	%3.22	92	25

Tablo 10: Uçtan uca farklı katmanlarda gecikmeler ($Frame_{max} = 104$B) .

Bu sonuçta görüldüğü üzere paket boyutu ve zaman dilimi boyutu maksimum uygulamadan uygulamaya gecikmeyi etkilememektedir. Yukarıdaki sonuçlar Şekil 44'de görülen bilgisayar tabanlı deney düzeneği ile gerçekleştirilmiştir.

Deney 1 Şekil 45'da görülen ve iki kontrolcunun WAGO PLC ile gerçekleştirildiği düzenekle de tekrarlanmıştır. Sonuçlar Tablo 11'da görülebilir.

Ölçüm	**Ortalama**	$\% \pm \Delta$	**Max.**	**Min.**
Uygulamadan uygulamaya	3617.5	%2.2	4928.9	185.2
2 düğüm KK arası	162.4	%6.4	345.7	105.3
2 düğüm AK arası	124.7	%5.8	198.9	95.3

Tablo 11: Uçtan uca farklı katmanlarda gecikmeler. Deney Konfigürasyonu 2.

Deney 2: D^2RIP, ek gerçek zamanlı trafik altında başarım

Bu deneyde Bölüm 3.6'deki örnekteki kontrolcular tarafından üretilen trafiğe ek gerçek zamanlı veri üretilmiştir. Kontrol uygulaması verisi Bölüm 3.4.1'da tanımlandığı şekilde kanal 1'den gönderilirken bu ek veri kanal 2'den gönderilmiştir.

Bu ek gerçek zamanlı trafik altında yapılan ölçümler Tablo 12'daki gibidir.

Ölçüm	**Ortalama**	$\% \pm \Delta$	**Max.**	**Min.**
Uygulamadan uygulamaya	3542,1	%2	4765	211
2 düğüm KK arası	86,2	%5.18	136	17
2 düğüm AK arası	71.3	%8.23	116	34

Tablo 12: Ek gerçek zamanlı trafik altında uçtan uca farklı katmanlarda gecikmeler.

Deney 3: D^2RIP gerçek zamanlı olmayan trafik başarımı

Bu deneyde Örnek kontrol uygulaması calışırken ağdaki düğümler tarafından farklı yüklerde ve paket uzunluklarında nRT trafik yollanmıştır. Aşağıdaki Tablolarda bu farklı özelliklerdeki nRT trafik için gecikme değerleri incelenmektedir. Bütün zaman ölçümleri μsec cinsindendir.

Gönderim periyodu	**Ortalama**	$\% \pm \Delta$	**Max.**	**Min.**
500 μsec	*Çokuzun*	$\% - -$	*Çokuzun*	$- - -$
1 msec	1530.1	%0.83	5940	90
10 msec	636.2	%2.03	3295	57
100 msec	620.1	%6.92	3124	64

Tablo 13: Gerçek zamanlı olmayan trafik. 40 Byte paketler.

Gönderim periyodu	**Ortalama**	$\% \pm \Delta$	**Max.**	**Min.**
500 μsec	*Çokuzun*	$\% - -$	*Çokuzun*	$- -$
1 msec	*Çokuzun*	$\% - -$	*Çokuzun*	$- -$
2 msec	2312	%0.86	5563	124
10 msec	845,2	%1,91	4235	86
100 msec	735.6	%6.14	3564	78

Tablo 14: Gerçek zamanlı olmayan trafik. 40 Byte paketler. Kanal 2 aktif.

Paket uzunluğu arttıkça gecikmeler artmaktadır. Bunun sebebi uzun nRT paketlerin parçalanması ve bütün parçalar gönderildikten sonra paketin gönderilmiş olmasıdır. Her parça gönderimi için düğüm sıranın kendisine gelmesini beklediği için toplam gecikme artmaktadır. Öte yandan düğümler kendilerine nR Tpaket göndermek için sıra geldiğinde sadece tek bir paket yollamaktadırlar. Eğer bu paket zaman dilimine göre çok küçükse zaman dilimi kapasitesini kullanamamakta ve efektif nRT kapasite de düşmektedir.

Gönderim periyodu	Ortalama	$\% \pm \Delta$	Max.	Min.
1 msec	$--$	$\% --$	*Çokuzun*	$--$
2 msec	$--$	$\% --$	*Çokuzun*	$--$
2.5 msec	2153	%0.47	10235	1756
5 msec	1728.4	%0.59	5354	1081
10 msec	1683.2	%0.82	4320	1082
100 msec	1696.3	%2.65	3853	1079

Tablo 15: Gerçek zamanlı olmayan trafik. 576 Byte paketler.

Gönderim periyodu	Ortalama	$\% \pm \Delta$	Max.	Min.
2 msec	$--$	$\% --$	*Çokuzun*	$--$
2.5 msec	$--$	$\% --$	*Çokuzun*	$--$
5 msec	3842.3	%0.28	6698	3101
10 msec	3844.3	%0.38	6541	3079
100 msec	3827.5	%1,26	6118	3087

Tablo 16: Gerçek zamanlı olmayan trafik. 1500 Byte paketler.

4.4 D^{3}RIP Başarım İncelemesi

Güvenirlik katmanı gerçekleştiriminden sonra örnek sistem için güvenirlik ölçümleri yapılmıştır. Bu ölçümlerde güvenirlik için yapılan ek işlemleri kapsamak üzere zaman dilimi uzunluğu 300 μsn seçilmiştir.

Örnek sistemde iki paket gönderimi arasındaki en yüksek zaman (Δ_I) bir düğüme ayrılmış iki zaman dilimi arasındaki en fazla zaman diliminden bulunur. Bu parametre örnek uygulamada $\Delta_I = 6$ zaman dilimi olarak bulunabilir. Arka arkaya gelen ve herhangi bir düğüme eşlenmemiş zaman dilimleri, bütün zaman dilimleri kullanıldığı için $\Delta_S = 0$dır. Olabilecek en fazla arka arkaya paylaşımlı ortam üzerinde çarpışma sayısı $\Delta_C = 4$ varsayılmıştır. Bir düğüm için maksimum arka arkaya gönderim sayısı bir RT bir nRT zaman dilimi olarak $\Delta_N = 2$'dir. Bu varsayımlara göre en uzun hata tespit zamanı Δ_F aşağıdaki gibi hesaplanır.

$$\Delta_F = \Delta_I + \Delta_S + \Delta_C + \Delta_N . = 12. \tag{6}$$

Buna göre Güvenirlik Katmanının $\Delta_F = 12$ durum değişkeni değerlendirmesini belleğinde tutması gerekir. Ayrıca [KARTAL et al. (2013a)]'de belirtildiği üzere, eğer zaman dilimi x'de kabul testi yoluyla bir hata tespit edildiyse geri toplama (rollback) zaman dilimi $x - \Delta_I$ olacaktır.

Yukarıdaki duruma uygun olarak iki hatalı deney yapılmıştır.

İlk deneyde R (Robot) bileşeninde `vCL` veri yapısında bir hata oluşturulmuştur. Bu hata nedeni ile R zaman dilimi 40'da yanlış durum değişkenlerini yollamaktadır. Bu hata kabul testi yoluyla R harç bütün düğümlerde tespit edilmiştir. Buna göre geri toplama zaman dilimi olarak $40-6=34$ zaman dilimine karar verilmiştir. Deneyde bunu takip eden zaman dilimi de R'ye ayırılmıştır. Bu şekilde geri toplama isteği ancak bir zaman dilimi sonra S tarafından yollanabilmiştir. Bu zaman diliminde bütün düğümler doğru olarak zaman dilimi 34'e geri dönmüşler ve çalışmaya devam etmişlerdir.

Ölçülen hatadan geri dönme (fault recovery) zamanı 8 zaman dilimi olarak $8 \cdot 0.3\,\text{ms} = 2.4\,\text{ms}$ ölçülmüştür. Bu gözlenen zaman bu örnek için bulunan maksimum zaman olan $12 \cdot 0.3\,\text{ms} = 3.6\,\text{ms}$ den küçüktür.

İkinci deneyde PD cihazı protokol tarafından düğüm S'ye ayrılmış olan 160. zaman diliminde mesaj göndererek bir çarpışmaya neden olmuştur. Gönderilmesi gereken mesaj çarpışma nedeniyle gelmeyince bütün düğümlerde kabul testı false sonuç vermiştir. Buna göre geri dönülen zaman dilimi $160-6=154$ olarak belirlenmiştir. Buna ek olarak S geri dönüş isteği paketini yollamadan önce ağda 3 çarpışma daha olduğu görülmüştür. Takiben, bütün düğümler zaman dilimi 154'e dönüp normal operasyonlarına devam etmişlerdir. Bu deney için hatadan geri dönme zamanı $6+4=10$ zaman dilimi olarak gözlemlenmiştir. Bu da $10 \cdot 0.3\,\text{ms} = 3\,\text{ms}$. olarak ölçülmüştür.

Bu deney sonuçları [FELSER (2009)]'da endüstriyel ortamda sıklıkla görüldüğü belirtilen 20 ms'lik zamana göre uygundur.

4.5 D^{2}RIP Simülatör Yazılımı Sonuçları

Bu sonuçlar detaylı olarak 4. Dönem Raporunda sunulmaktadır.

D^{2}RIP Simülatörü'nün çıktıları, gerçek bir D^{2}RIP düzeneği çıktılarıyla kıyaslanmıştır. Bu düzenek 2 kontrolcu (*Operasyon kontrolcusu G*1 ve *Hata Kontrolcusu G*2) ve bir kontrol edilen sistemden oluşmaktadır.

Bu örnek sistem için D^{2}RIP işçerçevesi içinde statik gerçek zamanlı ve gerçek zamanlı olmayan zaman dilimleri olan bir AK gerçekleştirilmiştir. KK için URT kullanılmıştır. Buna göre Gerçek zaman dilimlerinin hangi düğüm tarafında kullanılacağına gönderilecek mesajların son geçerlilik zamanlarına (deadline) göre dağıtık bir öncelikli kuyruk yapısıyla karar veren bir gerçekleştirme yapılmıştır. Zaman dilimi 3ms, gerçek zamanlı veri boyutu 136Byte, AK hesap zamanı (cmp) 0.1 ms, KK hesap zamanı ($rem-cmp$) 0.4 ms seçilmiştir. Sistemde AK protokolü olarak RAIL koşturulmaktadır.

Kontrol edilen sistem Intel Atom Işlemci tabanlı gömülü sistem geliştirme platformunda, *G*1 ve *G*2 kontrolcuları ise birbirlerine 10 Mbps Ethernet Hub ile bağlı iki masaüstü bilgisayarda gerçekleştirilmiştir. Her iki kontrolcu da 10 ms'de bir 150 Bytelık gerçek zamanlı olmayan trafik üretmektedirler. IEEE 1588 eşzamanlama protokolu da yüksek öncelikli gerçek zamanlı olmayan uygulama olarak çalışmaktadır. Kontrol edilen sistem ile kontrolcular arasındaki bağlantı ise seri kanaldan yapılmaktadır. Kontrolcu uygulaması ve KK ile arayüzü, Erlangen Üniversitesi, Kontrol Teknolojisi Enstitüsü, Almanya ile yapılan ortak çalışma kapsamında libFAUDES [LIB (2006–2013)] yazılım kütüphanesi ile gerçekleştirilmiştir.

Benzeştirimde ise $dSlot = 3000$, $rem = 500$ ve $cmp = 100$ zaman birimi olarak alınmıştır. Dolayısıyla 1 zaman birimi 1μs olarak belirlenmiştir. Bu değişkenlerin farklı seçimleri de mümkündür. Ancak her bir seçimin sistem performansı üzerinde belirli bir sınırı vardır. Örneğin *dSlot'*a bağlı olarak bir zaman aralığında gönderilebilecek en fazla mesaj uzunluğu değişebilecek, ancak çok yüksek olur ise, bant genişliği boşa harcanmış olacaktır.

RT haberleşme benzetimi için, öncelikle gerçek sistemde AP2CL olaylarının hangi zaman aralıklarının hangi anlarında olduğu kaydedilmiştir. Daha sonra bu bilgiler, benzeştirim için girdi olarak kullanılmış ve yapma olaylar yaratılmıştır. Bu olayların benzeştirimde yer almasından sonra yapılacak tek şey, sistemin akışını izlemek ve analiz etmektir. Örneğin, gerçek sistemde oluşan AP2CL olayından sonra hangi paketin hangi zaman aralığında, hangi anda hangi olay ile nereden nereye iletildiğini, hangi anda hangi olayın meydana geldiğini vb. inceleyerek D^{3}RIP Benzeştiricisi'nin gerçek sistemle özdeş olup olmadığı kontrol edilecektir.

Gerçek Zamanlı Paketler

Öncelikle, gerçek sistem çalıştırılmıştır ve yaratılmış olan mesajların zamanlamaları kaydedilmiştir. Daha sonra D^{3}RIP Benzeştiricisi, o zamanlarda aynı mesajları yaratır ve ÖK'ye ekler. Dolayısıyla benzeştirim süreci tetiklenmiştir. Daha sonra bu mesajların ilgili CL2IL ve CL2AP olaylarının ÖK'den çıkma zamanlamaları kaydedilerek gerçek sistemdeki zamanlamalar ile kıyaslanacaktır. Dolayısıyla, aynı anda yaratılmış olan bir mesajın D^{3}RIP Yığını içerisinde hareketinin D^{3}RIP Benzeştiricisi kullanılarak birebir benzeştirildiği an be an incelenebilecektir.

Toplam olarak 31 gerçek zamanlı paket incelenmiştir. Her bir paket için 6 olayın zamanlaması kaydedilmiştir ve toplamda 186 adet olay kıyaslanmıştır. Bu kıyaslamanın sonucunda, bir olayın gerçek sistemde gerçekleşmesi ile benzeştirimde gerçekleşmesi arasında ortalama 122.6 μs fark olduğu gözlemlenmiştir. Tüm olaylar göz önüne alındığında bu fark, en çok 401 μs; en az ise 0 olarak ortaya çıkmıştır.

Gerçek Zamanlı Olmayan Paketler

Gerçek zamanlı paketlere benzer olarak, toplam 95 gerçek zamanlı olmayan paket iletimi incelenmiştir. Her bir gerçek zamanlı olmayan paket için de 6 adet olayın zamanlaması incelenmiştir ve gerçek sistem zamanlamaları ile kıyaslanmıştır. Bir olayın gerçek sistemde gerçekleşmesi ile benzeştirimde gerçekleşmesi arasında ortalama 139.6 μs fark olduğu gözlemlenmiştir. Tüm olaylar göz önüne alındığında bu fark, en çok 344 μs; en az ise 0 olarak ortaya çıkmıştır.

Şu sonuca ulaşılabilir ki, bütün kıyaslamalar arasında en fazla fark 401 μs olduğuna göre, paketler ve olaylar, arada herhangi bir eşzaman farklılığına yol açmaksızın aynı zaman aralığında servis görmektedir. Yani, bir zaman aralığında gerçek sistemde işlem görmekte olan bir olay, benzeştirimde de aynı zaman aralığında işlem görmektedir. Sonuç olarak, gerçek zamanlı olmayan mesaj tamponları, zaman aralığı sayıları, olay zamanlamaları, herhangi bir değişkenin veya sistemin durumunun benzeştirim ve gerçek düzenekte farklılık göstermediği görülmüştür.

Dikkat edilmelidir ki, benzeştirim bazı varsayımlar ve soyutlamalar altında tutarlı olmaktadır. Bunlar öncelikle KHY ve zamanlayıcı gecikmeleri ile RT Linux'ın sınırlı eşzaman çalışmasıdır

Daha önce belirtildiği gibi, denetleyiciler 500 μs'ye kadar eşzamanlı iken gerçek sistemden veri toplanmaya başlanmıştır. Buna göre, gerçek sistem ve benzeştirimdeki aynı olaylar arasındaki en yüksek fark 401 μs olduğu görülmüştür. En iyi durumda, bu fark bu kadar yüksek çıkmayacaktır. Genel olarak, veriler denetleyicilerin 100 μs ile en yüksek eşlemeye sahip olduklarında alındığı takdirde, en yüksek 300 μs fark gözlenecektir.

Son olarak, 35,3 μs standart sapma ile toplam 126 paket, 756 olay incelemesi yapılmıştır.

Yani, eğer kıyaslama işlemi farklı veriler üzerinde yapılmaya devam ettikçe, yapılan kıyaslamaların %90,%95 ve %99'unda standart sapma belirtilen aralıklarda çıkacaktır.

Toplam 756 olay için, ortalama fark 135,42 μs olarak ölçülmüştür. Sabit standart sapma için, 0.95 güvenilirlik aralığı, (134,994 μs 135,841 μs) olarak hesaplanmıştır. Buradan, yapılacak olan farklı benzeştirim sonuçlarının %95'inde ortaya çıkacak olan fark, bu aralık içerisindedir.

D^{2}RIP Benzeştiricisi'nin karmaşıklığı öncelikli olay kuyruğu yönetimine bağlıdır. Sistemin en yoğun durumunda, yani her zaman aralığında bir aygıtın iletime hazır olduğu durumda, sisteme en çok yük binecektir. Her bir RT paket için, AP2CL CL2ILRT IL2SM olayları ÖK'ye itile-

cektir. Bu mesajın alımları esnasında ise, SM2IL IL2CLRT CL2AP olayları ÖK'ye itilecektir. nRT mesajlar için ise, iletim için AP2ILnRT IL2SM, alım için SM2IL IL2APnRT olayları ÖK'ye itilecektir. Yani, her bir gerçek zamanlı mesajı iletimi ve alımı için 3, nRT mesajı iletimi ve alımı için 2 olay ÖK'ye itilir.

Böylece, Denklem (7)'de görülen 4 zaman aralığında 2 gerçek zamanlı, 2 gerçek zamanlı olmayan mesaj iletilecektir. Bu mesajlar bütün aygıtlar tarafından alınacaktır. Toplam olarak,

$$2 \cdot RT_{iletim} * 3 \cdot \frac{olay}{RT_{iletim}} + 2 \cdot RT_{alım} * 3 \cdot \frac{olay}{RT_{alım}} + 2 \cdot nRT_{iletim} * 2 \cdot \frac{olay}{nRT_{iletim}} + 2 \cdot nRT_{alım} * 2 \cdot \frac{olay}{nRT_{alım}} = 20 \quad (7)$$

olay ÖK'ye itilecektir ve servis edilecektir. Eğer yük daha fazla olmaz ise, ÖK boyutu her döngü sonrası aynı kalmış olacaktır. Eğer yük daha fazla olur ise ÖK boyutu sürekli artacak ve belirli bir zaman sonra sistem hata verecektir. Bu durum, D^3RIP Benzeştiricisi için bir sınırlayıcı faktör olarak düşünülebilir.

5 Tartışma ve Sonuç

Bu projede dağıtık endüstriyel sistemlerin paylaşımlı ortam Ethernet ağları üzerinde gerçek zamanlı haberleşmeleri için bir protokol yığıtı işçerçevesi tasarlanmış ve bu işçerçevesine uygun olarak *Dynamic Distributed Realtime Industrial Communication Protocol (D^3RIP)-Dinamik Dağıtık Güvenilir Gerçek Zamanlı Endüstriyel İletişim Protokolu (D^2G^2EP)* adını verdiğimiz protokol yığıtı gerçekleştirmeleri yapılmıştır.

Projenin ana çıktısı endüstriyel gerçek zamanlı Ethernet ağlarında kontrol uygulamaları trafik özelliklerini dikkate alan ve formal modellemeye dayalı yaklaşımımızdır. Yaptığımız gerçekleştirim bu yaklaşımın avantajlarını göstermeye yönelik ve akademik bir çalışmadır. Buna rağmen, bu gerçekleştirimde örnek üzerinde elde edilen 15Mbps efektif gerçek zamanlı trafik kapasitesi ve yaklaşık 3.5 msn uygulamadan uygulamaya gecikme değerleri Tablo 1'da ki değerlere bakıldığında literatürdeki profesyonel protokol başarımlarından iyi ya da çok yakındır. Önemli bir nokta; ölçülen gecikmelerin RT kontrolcu mesajlarının geçerli olmak için bekledikleri zamanı da içermesidir(elibility time). Dolayısı ile bu gecikmenin büyük bir kısmı uygulamanın çalışma şeklindendir. Mesaj geçerli olduktan sonra karşı tarafa gönderilme zamanı bir zaman diliminin (250μs) altındadır. Gerçekleştirim profesyonel bir gerçek zamanlı işletim sistemi ile daha iyi başarımlar yakalayabilecektir.

İşçerçevesinin tasarımı TIOA formal modeliyle, doğrulaması UPPAAL doğrulama yazılım aracıyla yapılmıştır. D^3RIP yığıtının Güvenirlik Katmanı olmadan ana gerçeklemesi olan D^2RIP sıkı gerçek zamanlı garantiler verebilecek şekilde gerçekleştirilmış, D^3RIP'de ise geri toplamaya dayalı bir yöntem uygulanmıştır.

Projede akademik yenilik içeren pek çok çıktı elde edilmiştir. Bu çıktıların bir kısmı yayınlanmış bir kısmı hakem değerlendirmesinde ya da dergiye sunuma hazır durumdadır. Projede yarı zamanlı bursiyer ve gönüllü olarak çalışan bir doktora ve beş yüksek lisans öğrencilerinin tezleri tamamen bu projede yapılan işlerden oluşmuştur. Bu sonuçlarıyla proje yürütücüsünün kariyerine çok önemli bir destek olmuştur.

Proje yürütülmesi sırasında bazı teknik sorunlarla karşılaşılmıştır. IEEE 1588 protokolunun Ethernet tabanlı Endüstriyel ağlarda eşzamanlama için kullanılacağı proje önerilirken öngörülmüş ve eşzamanlama için IEEE 1588 kullanılmasına karar verilmiştir. İlgili iş paketi yapılırken o sırada piyasada nadir bulunan IEEE 1588 donanım zaman damgası destekli kartlardan iki adet alınmıştır. Bu kartlar bir üniversite laboratuvarında geliştirilen ve profesyonel desteği olmayan kartlar oldukları için gerçek zamanlı işletim sistemi ile çıkan sorunlar çözülememiştir. Eş zamanlama yazılımda yapıldığında hassasiyet az olduğu için ilk geliştirimlerde istenen başarımları elde edememiştir. Teknolojinin gelişmesiyle 2013 yılı başında Intel tarafından seri üretim IEEE 1588 donanım zaman damgası destekli kartlar piyasaya çıkmış ve bu kartların katkısıyla başarım çok iyileştirilmıştır. Buna ek olarak proje önerilirken öngörülmeyen pek çok gerçek zamanlı işletim sistemi optimizasyonu ve ayarları gerekmiştir.

Proje ekibi açık kaynak gerçek zamanlı işletim sistemleri ile yapılan gerçekleştirimlerin başarımlarının iyileştirilmesinin oldukça güç olduğunu gözlemlemiştir. Bu nedenle proje yapılan deneylerde iyi sonuç elde etmek için Linux işletim sistemi ile ilgili çok detay bilgilerin bulunması gerekmiştir. Projedeki bazı iyileştirmeler Linux ya da güncel Ethernet arayüz kartı sürücülerinde son bir kaç haftada olan teknik gelişmelerle mümkün olmuştur.

Gerçek zamanlı Ethernet ile ilgili bu hızlı gelişme süreci proje yöneticisi ve ekibinin geleceğe yönelik gelişme potansiyeli çok olan bir alanda çalışma yaptıklarına dair inançlarını güçlendirmiştir.

Proje bütçesinde bursiyerler için ayırılan paradan artma olmuştur. TÜBİTAK tarafından sağlanan tam zamanlı burs yüksek bir miktar olmasına karşılık, projenin yürütüldüğü Orta Doğu Teknik Üniversitesi Elektrik Mühendisliği Bölümü'nde tam zamanlı burslu yüksek lisans ve doktora öğrencisi bulmak oldukça zordur. Buna karşılık beş öğrencinin gerçek zamanlı işletim sistemleri, gerçek zamanlı Ethernet, protokol yığıtı tasarımı ve gerçekleştirimi, formal modelleme ve doğrulama teknikleri, güvenirlik gibi önemli konular hakkında akademik ve prtaik olarak bilgi ve tecrübe sahibi olmaları olumlu bir kazanımdır.

Projenin bitişini takiben yeni bir yüksek lisans öğrencisi ile birlikte D^3RIP düğümlerinden ıluşan bir otomasyon sistemi ağı ile standart ev-ofis ortamlardında kullanılan Ethernet arasında iletişimi ve gerekli paket çevirimlerini ve çizelgelemeyi sağlayacak bir gateway (ağ kapısı) düğümü tasarımına başlancaktır.

Proje yöneticisi kariyerinde önemli bir aşama sağlayan bu destek için TÜBİTAK'a teşekkür eder.

Referanslar

[AVIZIENIS et al. (2004)] AVIZIENIS A., Laprie J.-C., Randell B., Landwehr C., Basic concepts and taxonomy of dependable and secure computing, *Dependable and Secure Computing, IEEE Transactions on*, 1, 11–33, (2004).

[RANDELL (1975)] RANDELL B., System structure for software fault tolerance, *SIGPLAN Not.*, 10, 437–449, (1975).
URL http://doi.acm.org/10.1145/390016.808467

[RAMANATHAN (1993)] RAMANATHAN P., Shin K. G., Use of common time base for checkpointing and rollback recovery in a distributed system, *IEEE Trans. Softw. Eng.*, 19, 571–583, (1993).
URL http://dx.doi.org/10.1109/32.232022

[SCHMIDT (2012)] SCHMIDT K. W., Schmidt E. G., Distributed real-time protocols for industrial control systems: Framework and examples, *Parallel and Distributed Systems, IEEE Transactions on*, 23, 1856–1866, (2012).

[LARSEN et al. (1994)] LARSEN K. G., Pettersson P., Yi W., *Uppaal in a nutshell*,.

[KARTAL et al. (2013)] KARTAL Y. B., Schmidt K. W., Schmidt E. G., Modeling and formal verification of TIOA based distributed real-time systems, *Software Engineering, IEEE Transactions on*, Submitted.

[KARTAL et al. (2013a)] KARTAL Y. B., Schmidt K. W., Schmidt E. G., Dependability design for a distributed real-time protocol family, *Parallel and Distributed Systems, IEEE Transactions on*, Submitted.

[KAYA et al. (2013)] KAYA A., Schmidt E. G., Schmidt K. W., Moor T., Dynamic distributed realtime industrial communication protocol (D^2RIP): Architecture, implementation and experimental evaluation, *Computer Standards and Interfaces*, Submitted.

[BAILLIEUL (2007)] BAILLIEUL J., Antsaklis P., Control and communication challenges in networked real-time systems, *Proceedings of the IEEE*, 95, 9–28, (2007).

[MOYNE (2007)] MOYNE J., Tilbury D., The emergence of industrial control networks for manufacturing control, diagnostics, and safety data, *Proceedings of the IEEE*, 95, 29–47, (2007).

[DECOTIGNIE (2005)] DECOTIGNIE J.-D., Ethernet-based real-time and industrial communications, *Proceedings of the IEEE*, 93, 1102–1117, (2005).

[DECOTIGNIE (2009)] DECOTIGNIE J.-D., The many faces of industrial Ethernet [past , present], *Industrial Electronics Magazine, IEEE*, 3, 8–19, (2009).

[THOMESSE (2005)] THOMESSE J.-P., Fieldbus technology in industrial automation, *Proceedings of the IEEE*, 93, 1073–1101, (2005).

[FELSER (2009)] FELSER M., Real-time Ethernet for automation applications, in: Zurawski R., (Ed.), Embedded Systems Handbook, Second Edition: Networked Embedded Systems, CRC Press, Inc., Boca Raton, FL, USA, , (2009), pp. 21–1–21–20 2nd edition.

[IEEE (2002)] IEEE 1588 standard for a precision clock synchronization protocol for networked measurement and control systems, `http://ieee1588.nist.gov`, (2002).

[EIDSON (2006)] EIDSON J. C., *Measurement, Control, and Communication Using IEEE 1588* , Springer, (2006).

[JANSEN (2004)] JANSEN D., Buttner H., Real-time Ethernet the EtherCAT solution, *Computing & Control Engineering Journal*, 15, 16–21, (2004).

[SCHEMM (2004)] SCHEMM E., Sercos to link with Ethernet for its third generation, *Computing & Control Engineering Journal*, 15, 30–33, (2004).

[NESIC et al. (2009)] NESIC D., Teel A., Carnevale D., Explicit computation of the sampling period in emulation of controllers for nonlinear sampled-data systems, *Automatic Control, IEEE Transactions on*, 54, 619–624, (2009).

[DEP (2007)] 1st IFAC workshop on dependable control of discrete systems, (2007).

[FELSER (2004)] FELSER M., Sauter T., Standardization of industrial Ethernet - the next battlefield?, *Factory Communication Systems, 2004. Proceedings. 2004 IEEE International Workshop on*, 413–420, (2004).

[PRO (2004)] Real-time Ethernet: Profinet IO: Proposal for a publicly available specification for real-time Ethernet, Doc. IEC 65C/359/NP, (2004).

[MOD (2002)] Schneider automation – modbus messaging on TCP/IP implementation guide, http://www.modbus.org/, (2002).

[KWEON (2003)] KWEON S.-K., Shin K. G., Statistical real-time communication over Ethernet, *Parallel , Distributed Systems, IEEE Transactions on*, 14, 322–335, (2003).

[KWEON et al. (2004)] KWEON S.-K., Cho M.-G., Shin K. G., Soft real-time communication over Ethernet with adaptive traffic smoothing, *Parallel and Distributed Systems, IEEE Transactions on*, 15, 946–959, (2004).

[FELSER (2005)] FELSER M., Real-time Ethernet - industry prospective, *Proceedings of the IEEE*, 93, 1118–1129, (2005).

[LIU (2007)] LIU L., Frey G., Simulation approach for evaluating response times in networked automation systems, *Emerging Technologies & Factory Automation, 2007. ETFA. IEEE Conference on*, 1061–1068, (2007).

[SCHARBARG (2007)] SCHARBARG J.-L., Fraboul C., Simulation for end-to-end delays distribution on a switched Ethernet, *Emerging Technologies & Factory Automation, 2007. ETFA. IEEE Conference on*, 1092–1099, (2007).

[WANG et al. (2002)Wang, Song, Chen, , Sun] WANG Z., Song Y., Chen J., Sun Y.,Real time characteristics of Ethernet and its improvement, , (in: 4th World Congr. Intelligent Control , Automation), (2002), pp. 1311–1318.

[EIP (2004)] Real-time Ethernet: Ethernet/IP with time synchronization: Proposal for a publicly available specification for real-time Ethernet, Doc. IEC 65C/361/NP, (2004).

[MORAES et al. (2011)] MORAES R., Carreiro F., Bartolomeu P., Silva V., Fonseca J., Vasques F., Enforcing the timing behavior of real-time stations in legacy bus-based industrial ethernet networks, *Computer Standards and Interfaces*, 33, 249–261, (2011).

[TCN (2004)] Real-time Ethernet: TCnet (Time-Critical Control Network): Proposal for a publicly available specification for real-time Ethernet, Doc. IEC 65C/353/NP, (2004).

[EPL (2004)] Real-time Ethernet: EPL (Ethernet powerlink): Proposal for a publicly available specification for real-time Ethernet, Doc. IEC 65C/356a/NP, (2004).

[EPA (2004)] Real-time Ethernet: EPA (Ethernet for plant automation): Proposal for a publicly available specification for real-time Ethernet, Doc. IEC 65C/357/NP, (2004).

[PEDREIRAS et al. (2005)] PEDREIRAS P., Gai P., Almeida L., Buttazzo G., FTT-Ethernet: a flexible real-time communication protocol that supports dynamic QoS management on Ethernet-based systems, *Industrial Informatics, IEEE Transactions on*, 1, 162–172, (2005).

[KAYNAR et al. (2003)] KAYNAR D. E., Lynch N., Segala R., Vaandrager F., The theory of timed I/O automata, Tech. Rep. MIT-LCS-TR-917, MIT Laboratory for Computer Science, (Cambridge, MA).

[KAYNAR et al. (2003a)] KAYNAR D. E., Lynch N., Segala R., Vaandrager F.,Timed I/O automata: A mathematical framework for modeling and analyzing real-time systems, , (in: 24th IEEE International Real-Time Systems Symposium), (2003), .

[WANG YI, (1997)] WANG YI P. P., Daniels M., *Automatic verification of real-time communicating systems by constraint solving*.

[NELSON (1990)] NELSON V., Fault-tolerant computing: fundamental concepts, *Computer*, 23, 19 –25, (1990).

[AVIZIENIS et al. (2001)] AVIZIENIS A., Laprie J. C., Randell B., Fundamental concepts of dependability, *Technical Report Series University Of Newcastle*, 1145, 7–12, (2001).

[MEYER (1999)] MEYER B., Every little bit counts: toward more reliable software, *Computer*, 32, 131 –135, (1999).

[CHILLAREGE et al. (1992)] CHILLAREGE R., Bhandari I., Chaar J., Halliday M., Moebus D., Ray B., Wong M.-Y., Orthogonal defect classification-a concept for in-process measurements, *IEEE Transactions on Software Engineering*, 18, 943–956, (1992).

[PAULK et al. (1993)] PAULK M. C., Curtis B., Chrissis M. B., Weber C. V., Capability maturity model, version 1.1, *IEEE Softw.*, 10, 18–27, (1993).
URL `http://dx.doi.org/10.1109/52.219617`

[AVIZIENIS (1967)] AVIZIENIS A., Design of fault-tolerant computers, in: Proceedings of the November 14-16, 1967, fall joint computer conference, AFIPS '67 (Fall), (ACM, New York, NY, USA,), (1967), pp. 733–743.
URL `http://doi.acm.org/10.1145/1465611.1465708`

[HORNING et al. (1974)] HORNING J. J., Lauer H. C., Melliar-Smith P. M., Randell B., A program structure for error detection and recovery, in: Operating Systems, Proceedings of an International Symposium, (Springer-Verlag, London, UK, UK,), (1974), pp. 171–187.
URL http://dl.acm.org/citation.cfm?id=647641.733522

[SHIN (1984)] SHIN K. G., Lee Y. H., Evaluation of error recovery blocks used for cooperating processes, *IEEE Trans Soft Eng*, SE-10, 692–700, (1984).

[GÖZCÜ (2011)] GÖZCÜ A. K., Implementation and evaluation of a synchronous time-slotted medium access protocol for networked industrial embedded systems, Master's thesis, Middle East Technical University, (Turkey).

[TURAN (2011)] TURAN U., Implementing and evaluating the coordination layer, time-synchronization of a new protocol for industrial communication networks, Master's thesis, Middle East Technical University, (Turkey).

[AYBAR (2011)] AYBAR G., Simulation and performance evaluation of a distributed real-time communication protocol for industrial embedded systems, Master's thesis, Middle East Technical University, (Turkey).

[GOZCU et al. (2012)] GOZCU A. K., Turan U., Schmidt E. G., Schmidt K. W.,The implementation of dynamic distributed real time industrial communication protocol (D2RIP), (in: Signal Processing and Communications Applications Conference (SIU), 2012 20th), (2012), .

[KARTAL et al. (2012)] KARTAL Y. B., Schmidt K. W., Schmidt E. G., The verification of a novel framework for real-time shared medium communication network protocols, (in: Signal Processing and Communications Applications Conference (SIU), 2012 20th), (2012), .

[ERL (2006–2013)] FG DES manufacturing system, www.rt.e-technik.uni-erlangen.de/FGdes/productionline.html, (2006–2013).

[MOOR et al. (2008)] MOOR T., Schmidt K., Perk S., libFAUDES - an open source C++ library for discrete event systems, *Discrete Event Systems, 9th International Workshop on*, 125–130, (2008).

[LIB (2006–2013)] libFAUDES software library for discrete event systems, www.rt.eei.uni-erlangen.de/FGdes/faudes, (2006–2013).

[CORRELL (2006)] Correll K., Barendt N.,Design considerations for software only implementations of the IEEE 1588 precision time protocol, , (in: In Conference on IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems), (2006), .

[LUB (2011)] Lubuntu, http://lubuntu.net/, (2011).

[INT (2012)] Intel 82574l Gigabit Ethernet Controller, http://ark.intel.com/products/32209/Intel-82574L-Gigabit-Ethernet-Controller, (2012).

[PTP (2011)] The Linux Ptp Project, http://linuxptp.sourceforge.net/, (2011).

[SCHMIDT et al. (2008)] SCHMIDT K., Schmidt E., Zaddach J.,Safe operation of distributed discrete-event controllers: A networked implementation with real-time guarantees, (in: IFAC World Congress), (2008), .

[SCHMIDT et al. (2007)] SCHMIDT K., Schmidt E., Zaddach J., A shared-medium communication architecture for distributed discrete event systems, *Control & Automation, Mediterranean Conference on*, 1–6, (2007).

[GARLAND et al. (2005)] GARLAND S. J., Kaynar D., Lynch N. A., Tauber J. A., Vaziri M., TIOA tutorial, (2005).

[XIAOWAN HUANG 1997)] XIAOWAN HUANG S. A. S., Anu Singh, *Using integer clocks to verify the timing-sync sensor network protocol*,.

[Gómez (2007)] GÓMEZ R., Bowman H.,Efficient detection of zeno runs in timed automata, in: Proceedings of the 5th international conference on Formal modeling , analysis of timed systems, FORMATS'07, (Springer-Verlag, Berlin, Heidelberg,), (2007), pp. 195–210.
URL `http://dl.acm.org/citation.cfm?id=1779879.1779894`

Ekler

Bu bölüm TÜBİTAK Proje sistemi dosya boyutları nedeni ile ikinci bir döküman olarak yüklenmiştir.

Erlangen Çalışma Ziyareti Hakkında Mektup

D^2RIP Başarım İyileştirimi Karşılaştırılması

Dynamic Distributed Realtime Industrial Communication Protocol (D^2RIP): Architecture, Implementation and Experimental Evaluation

Modeling and Formal Verification of Distributed Real-Time Systems Using TIOA and UPPAAL

Dependability Design for a Distributed Real-time Protocol Family

TÜBİTAK
PROJE ÖZET BİLGİ FORMU

Proje Yürütücüsü:	Doç. Dr. ŞENAN ECE SCHMİDT
Proje No:	109E261
Proje Başlığı:	Ethernet Üzerinde Dinamik, Dağıtılmış ve Güvenilir Endüstriyel Haberleşme Protokolları: Genel Tasarım işçerçevesi, Gerçekleştirim ve Deneysel Çalışma
Proje Türü:	Kariyer
Proje Süresi:	36
Araştırmacılar:	
Danışmanlar:	
Projenin Yürütüldüğü Kuruluş ve Adresi:	ORTA DOĞU TEKNİK Ü. MÜHENDİSLİK F. ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ B.
Projenin Başlangıç ve Bitiş Tarihleri:	01/03/2010 - 01/03/2013
Onaylanan Bütçe:	145836.0
Harcanan Bütçe:	110377.85

Öz:	Günümüz otomasyon sistemlerinde kontrol uygulamaları pek çok sayısal cihaz üzerinde dağıtılmış gömülü sistemler olarak gerçekleştirilmektedir. Bu cihazların koordinasyonu ve haberleşmesi endüstriyel haberleşme ağları üzerinden yapılmaktadır. Kontrol uygulamalarının gereklerine göre periyodik veya rasgele mesajlar gönderilmekte ve bu mesajlar gerçek zamanlı garantilere ihtiyaç duyabilmektedir. Buna ek olarak otomasyon sistemlerinin uzaktan idaresi ve çalışmalarının izlenmesi için gerçek zamanlı olmayan mesajların da ağ üzerinde taşınması gereklidir. Endüstriyel ağlar üzerlerinde çalışan kontrol uygulamaları insan hayatına doğrudan etki edebilecek özelliklerdedirler. Bu ağların her koşulda tanımlandıkları şekilde çalışmaları ve gerçek zamanlı garantileri sağlamaları gerekmektedir. Bu nedenle bu ağlar için protokolların formal olarak geliştirilebilmesi, incelenebilmesi ve çalışmalarının doğrulanabilmesi önemlidir. Literatürdeki endüstriyel ağ protokolları sistematik bir yaklaşım olmadan uygulama ihtiyaçları doğrultusunda ev ofis ağları için olan yaklaşımlara ekler ve değişiklikler yaparak, kontrol uygulamalarının davranışı hakkındaki bilgilerden faydalanmadan geliştirilmişlerdir. Pek çok var olan protokol için güvenirlik desteği bulunmamaktadır. Günümüzde ev ofis ağlarında çok yaygın olarak kullanılan fakat gerçek zamanlı haberleşmeyi desteklemeyen Standart Ethernet teknolojisinin endüstriyel haberleşme ağlarında kullanılması önemli bir araştırma konusudur. Bu projede yukarıdaki gereklere uygun olarak, gerçek zamanlı paylaşımlı ortam haberleşme protokolları için formal bir yaklaşımla genel bir işçerçevesi oluşturulmuş ve bu işçerçevesine uygun olarak bir protokol yığıtı geliştirilmiş ve gerçekleştirilmiştir. Projede geliştirilen formal işçerçevesi ve buna uygun olarak yapılan protokol gerçekleştirimi tamamen dağıtılmış bir mimariye sahiptir. Endüstriyel kontrol sistemlerinde gönderilen mesajların kontrol sistem modeline göre önceden belirlenmesi mümkündür. Bu özellikten yararlanarak ağ kaynakları anlık gerçek zamanlı haberleşme ihtiyaçlarına göre dinamik bir biçimde düğümlere dağıtılmakta ve artan ağ kapasitesi gerçek zamanlı olmayan trafik için kullanılmaktadır. Son olarak mimaride olası hata durumlarına karşı geri toplama mekanizmaları da mevcuttur. Bu özelliklerinden dolayı geliştirdiğimiz formal işçerçevesine ve bu işçerçevesine göre gerçekleştirilen üç katmanlı protokol yığıtına D3RIP (Dynamic Distributed Dependable Real-time Industrial Protocol) adı verilmiştir. Projemizin çıktısı olan formal işçerçevesi bu katmanların ve katman arayüzlerinin Timed IO Automata (TIOA) yöntemi ile ayrık olarak modellenmesi ve bu yönetmin sağladığı çeşitli özellikler ve operasyonlar kullanılarak bütün katman yığıtı modelinin elde edilmesine dayalıdır. Analitik olarak oluşturulan TIOA modelleri daha sonra UPPAAL yazılım aracıyla da benzetim yoluyla doğrulanmıştır. Bu aşamada TIOA modellerini UPPAAL modellerine algoritmik çeviren bir yöntem geliştirilmiştir. Formal işçerçevesine uygun olarak tamamen standart Ethernet donanımı üzerinde, farklı platformlara taşınabilir bir D3RIP Katman mimarisi gerçeklemesi yapılmış ve başarımı deneylerle incelenmiştir. Proje kapsamında 1 dergi makalesi, 2 konferans bildirisi yayınlanmıştır. İki dergi makalesi hakem değerlendirmesindedir. Proje kapsamında üç yüksek lisans tezi tamamlanmıştır. 2 yüksek lisans ve bir doktora tezi 2013 bahar dönemi sonunda tamamlanacaktır.
Anahtar Kelimeler:	Gerçek-zamanlı Ethernet, Dağıtık Hesaplama, Zamanlı giriş çıkışlı otomat, Formal model, Güvenirlik
Fikri Ürün Bildirim Formu Sunuldu Mu?:	Hayır
Projeden Yapılan Yayınlar:	1- Paylaşımlı Ethernet Üzerinde ÇalışanÖzgün Bir Gerçek--?Zamanlı Haberleşme Protokolü İşçerçevesinin Doğrulanması (Bildiri), 2- Dinamik Dağıtık Gerçek Zamanlı Endüstriyel İletişim Protokolu (d2gep) Gerçekleştirimi (Bildiri), 1- Distributed Real-Time Protocols for Industrial Control Systems: Framework and Examples (Makale/Kitap/Kitapta Bölüm)