T.C.
SÜLEYMAN DEMİREL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY SİNİR AĞI KULLANARAK KONTROL ALAN AĞLARI İÇİN ÇEVRİM İÇİ MESAJ ZAMANLAMASI OPTİMİZASYONU

Esin YAVUZ

Danışman
Prof. Dr. Ekrem ARTUÇ

DOKTORA TEZİ
FİZİK ANABİLİM DALI
ISPARTA - 2018

TEZ ONAYI

Esin YAVUZ tarafından hazırlanan **"Yapay Sinir Ağı kullanarak Kontrol Alan Ağları için çevrim içi mesaj zamanlaması optimizasyonu"** adlı tez çalışması aşağıdaki jüri üyeleri önünde Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü **Fizik Anabilim Dalı**'nda **DOKTORA TEZİ** olarak başarı ile savunulmuştur.

Danışman	**Prof. Dr. Ekrem ARTUÇ** Süleyman Demirel Üniversitesi	.....
Jüri Üyesi	**Prof. Dr. Fatih UCUN** Süleyman Demirel Üniversitesi	.....
Jüri Üyesi	**Prof. Dr. Selçuk ÇÖMLEKÇİ** Süleyman Demirel Üniversitesi	.....
Jüri Üyesi	**Prof. Dr. Şükrü ÖZEN** Akdeniz Üniversitesi	.....
Jüri Üyesi	**Dr. Öğr. Üyesi Gürkan BİLGİN** Mehmet Akif Ersoy Üniversitesi	.....

Enstitü Müdürü	**Prof. Dr. Yasin TUNCER**	.....

TAAHHÜTNAME

Bu tezin akademik ve etik kurallara uygun olarak yazıldığını ve kullanılan tüm literatür bilgilerinin referans gösterilerek tezde yer aldığını beyan ederim.

Esin YAVUZ

İÇİNDEKİLER

ÖZET

Doktora Tezi

YAPAY SİNİR AĞI KULLANARAK KONTROL ALAN AĞLARI İÇİN ÇEVRİM İÇİ MESAJ ZAMANLAMASI OPTİMİZASYONU

Esin YAVUZ

Süleyman Demirel Üniversitesi
Fen Bilimleri Enstitüsü
Fizik Anabilim Dalı

Danışman: Prof. Dr. Ekrem ARTUÇ

Bu tez çalışması, gerçek zamanlı veri iletişimi yapılan Kontrol Alan Ağı (CAN) için üretilen çevrim içi mesajların zamanlanarak, yapay sinir ağı ile modellenmesi aşamalarını içermektedir.

CAN ağını tasarlamak için Linux tabanlı SocketCAN uygulaması kullanılmıştır. Tasarlanan ağ için öncelik-temelli bir zamanlama algoritması uygulanmış ve mesaj önceliklerinin dinamik olarak değiştirilmesi sağlanmıştır. Optimal zamanlama için dinamik öncelikli EDF algoritması kapsamlı bir şekilde araştırılarak, modifiye edilmiştir.

Bu çalışmada, yapay sinir ağlarında zaman serisi tahminleri için kullanılan modellerden NARX (Doğrusal olmayan dışsal girdili otoregresif ağ) modelinin kullanılması uygun bulunmuştur. NARX ile optimal bir yapay sinir ağı modeli geliştirilerek, çeşitli çalışma koşulları için analiz edilmiştir. CAN ağından elde edilen çıktılar, NARX mimarisi kullanılarak modellenmiş ve bu modelleme ile ilgili en başarılı ağ yapısı belirlenmiştir.

Anahtar Kelimeler: CAN, Çevrim içi zamanlama, EDF, SocketCAN, NARX.

2018, 80 sayfa

ABSTRACT

Ph.D. Thesis

OPTIMIZATION OF ONLINE MESSAGE SCHEDULING FOR CONTROLLER AREA NETWORKS USING ARTIFICIAL NEURAL NETWORK

Esin YAVUZ

Süleyman Demirel University
Graduate School of Natural and Applied Sciences
Department of Physics

Supervisor: Prof. Dr. Ekrem ARTUÇ

This thesis contains the phases of scheduling on-line messages generated for the real-time Control Area Network (CAN) and the modeling of this network with the artificial neural network.

Linux based SocketCAN implementation has been used to design CAN network. A priority-based scheduling algorithm is implemented for the designed network and the message priorities are dynamically changed. The dynamic priority EDF algorithm for optimal scheduling has been extensively researched and modified.

In this thesis, it has been found appropriate to use NARX (Nonlinear Autoregressive Exogenous Model) model which is the one for time series estimation in artificial neural networks. An optimal artificial neural network model has been developed with NARX and analyzed for various operating conditions. Outputs from the CAN network have been modeled using the NARX architecture and the most successful network structure for this modeling has been identified.

Keywords: CAN, Online scheduling, EDF, SocketCAN, NARX.

2018, 80 pages

TEŞEKKÜR

Bu tez çalışmasında karşılaştığım zorlukları bilgi ve tecrübesi ile aşmamda yardımcı olan değerli Danışman Hocam Prof. Dr. Ekrem ARTUÇ'a teşekkürlerimi sunarım.

Çalışmama yön vermemdeki yardımlarından dolayı Tez İzleme Komitesi üyeleri Prof. Dr. Fatih UCUN'a ve Prof Dr. Selçuk ÇÖMLEKÇİ'ye teşekkürlerimi sunarım.

Çeşitli öneri ve fikirleriyle tez çalışmama olan katkılarından dolayı Öğr. Gör. Sertaç Selim SARICA'ya ve Dr. Öğr. Üyesi Övünç POLAT'a teşekkür ederim.

2205-D-10 No`lu Proje ile tezimi maddi olarak destekleyen Süleyman Demirel Üniversitesi Bilimsel Araştırma Projeleri Yönetim Birimi Başkanlığı'na teşekkür ederim.

Tezimin her aşamasında beni yalnız bırakmayan, desteklerini daima arkamda hissetiğim aileme teşekkürü bir borç bilirim.

Esin YAVUZ
ISPARTA, 2018

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

Sayfa

SİMGELER VE KISALTMALAR DİZİNİ

ACK	Onay (Acknowledgement)
API	Uygulama Programlama Arayüzü (Application Programming Interface)
ARQ	Otomatik Tekrar İstemi (Automatic Repeat Request)
ARX	Autoregressive Exogenous
CAN	Kontrol Alan Ağı (Controller Area Network)
CRC	Çevrimli Fazlalık Sınaması (Cylic Redundancy Check)
CSMA	Taşıyıcı Algılamalı Çoklu Erişim (Carrier Sense Multiple Access)
CSMA/CR	Çarpışma Çözümlemeli Taşıyıcı Algılamalı Çoklu Erişim (Carrier Sense Multiple Access with Collision Resolution)
DLC	Veri Uzunluk Kodu (Data Length Code)
DM	Tek Düze Zaman Sınırı (Deadline Monotonic)
ECU	Elektronik Kontrol Birimi (Electronik Control Unit)
EDF	En erken son tarih (Earliest Deadline First)
EOF	Çerçeve Sonu (End of Frame)
FTT-CAN	Flexible Time-Triggered communication over Controller Area Network
IFS	Çerçeveler Arası Boşluk (Inter Frame Space)
HRTS	Katı Gerçek-Zamanlı Sistemler (Hard Real-Time Systems)
ID	Tanımlayıcı (Identifier)
IDE	Tanımlayıcı Uzantı (Identifier Extention)
ISO	Uluslararası Standartlar Organizasyonu (International Standarts Organization)
LLC	Mantıksal Bağ Denetimi (Logical Link Control)
LLF	Least Laxity First
LST	Least Slack Time First
MAC	Ortak Erişim Denetimi (Medium Access Control)
MLF	Minimum Laxity First
MSE	Ortalama Hata Kareleri (Mean Square Error)
MTS	Karışık Trafik Zamanlaması (Mixed Traffic Scheduling)
NARX	Doğrusal olmayan dışsal girdili otoregresif ağ (Nonlinear Autoregressive Exogenous Model)
NRZ	Sıfıra Dönüşsüz Kodlama (Nonreturn to Zero)
OSI	Açık Sistemler Arası Bağlantı (Open Systems Interconnection)
Qdisc	Kuyruklama Disiplini (Queuing Discipline)
QoS	Servis Kalitesi (Quality of Service)
RM	Tek Düze Hız (Rate Monotonic)
RTR	Uzak İletim İstek (Remote Transmission Request)
SAE	Otomotiv Mühendisleri Birliği (Society of Automotive Engineers)
SOF	Çerçeve Başlangıcı (Start of Frame)
SRR	Yedek Uzak İstek Biti (Substitute Remote Request)
SRTS	Esnek Gerçek-Zamanlı Sistemler (Soft Real-Time Systems)
TC	Trafik Kontrolü (Traffic Control)
TTCAN	Zaman Tetiklemeli CAN (Time Triggered CAN)
UTP	Koruyucusuz Bükümlü Çift (Unshielded Twisted Pair)
VCAN	Sanal CAN (Virtual CAN)
YSA	Yapay Sinir Ağı

1. GİRİŞ

Yerine getirilmesi gereken görevlerini tam ve zamanında yapması gereken sistemler, "gerçek zamanlı" sistemler olarak tanımlanır. Görevlerin eksik ya da gecikmeli yapılması, üretim hatalarına ve hatta iş kazalarına neden olabilir. Gerçek zamanlı haberleşme sistemleri için de aynı durum geçerlidir. Gerçek zamanlı bir sistem, "kendi kontrolü altındaki süreçlerin zamanlama gereksinimlerini garanti edebilen bir sistem" olarak tanımlanabilir (Kopetz, 1997). Gerçek zamanlı haberleşme, tüm kullanıcıların anında, yani canlı olarak bilgi alışverişi yapabileceği bir telekomünikasyon şeklidir.

Gerçek zamanlı haberleşme sistemlerinde meydana gelebilecek bilgi kayıpları ya da bilgi akışındaki kesilmeler hizmet kalitesini önemli ölçüde azaltır. Gecikme, bir sistemin iletim sürecinde ağın giriş ve çıkış noktaları arasında hareket ettiği zaman dilimidir. Gecikme değişiklikleri (jitter) ise, belirli bir trafik akışında birbiri ardına gelen mesajların değişken karakterdeki gecikmelerinin ölçüsüdür. Her haberleşme sisteminde tolere edilebilir bir miktar gecikme payı bırakılır. Eğer bu süre aşılırsa, gerçek zamanlı haberleşme sistemleri gecikme süresine ve gecikmedeki değişmelere en hassas hizmetler olduğu için hatalar meydana gelir.

Gerçek zamanlı sistemler, mesajları sabit bir hızda ve aralarında sabit süreler varken almak isterler. Varış hızı değişkenlik gösterdikçe uygulamanın performansı etkilenir. Gerçek zamanlı sistemlerde, sistemin sadece çok yüksek hız performansı göstermesi değil, zamanlama performansının da kararlılık göstermesi gerekir; yani temel kriter planlanabilirlik ve öngörülebilirliktir.

Gerçek zamanlı haberleşme sistemleri, Katı Gerçek-Zamanlı Sistemler (HRTS) ve Esnek (ya da Gevşek) Gerçek-Zamanlı Sistemler (SRTS) olarak sınıflandırılırlar. Katı Gerçek-Zamanlı sistemlerde zaman kısıtlaması vardır ve zaman sınırı kesinlikle aşılmamalıdır. Hata toleransı, katı gerçek zamanlı sistemlerin gelişimi için hayati bir gerekliliktir. Bu zamanlama politikaları, geçici ya da kalıcı arızalar olsa bile, her koşulda görevlerin son teslim süreleri (deadline) içinde

iletileceğinin garantisini vermelidir. Zaman gereksinimleri ve arıza modeli, uygulamaya ve ortamına bağlıdır (Oliveira, 2003). Herhangi bir hata nedeni ile geciken veri kullanışsız bir veridir ve tüm sistemin kullanım dışı kalmasına sebep olarak felakete yol açabilir (Rouhifar ve Ravanmehr, 2015).

Esnek Gerçek-Zamanlı sistemlerde ise, daha esnek zaman kısıtları vardır; bazı zaman kısıtlarına uyulmayabilir. Oluşan hatalar nedeni ile zamanlama kısıtlamaları uygulamanın belirli bir periyodunda sağlanamazsa performans düşer fakat sistem kullanım dışı kalmaz. Kullanılan veriler halen geçerli olabilir (Livani vd., 1998; Ercek vd., 2005).

Zamanlama (scheduling), Katı Gerçek-Zamanlı sistemlerin temel problemidir. İletim önceliği, ağ modülünden çıkmakta olan trafiğin hangi sırada iletileceğini belirler. İletim önceliği olan mesaj, iletim önceliği olmayandan daha önce iletilir. Bu sebeple, mesajlar önceliklendirilir. Önceliklendirmenin sabit ya da değişken olmasına bağlı olarak, Katı Gerçek-Zamanlı iletişim uygulamaları için kullanılan çevrim içi (online) zamanlayıcı ve çevrim dışı (offline) zamanlayıcı olmak üzere iki ana zamanlama algoritması vardır. Çevrim içi zamanlayıcıda, uygulamaların farklı zamanlamaları olabilir ve mesajların öncelikleri sistemin işleyişi boyunca değişebilir. Çevrim dışı zamanlayıcıda ise uygulamaların sabit zamanlamaları vardır ve mesajlar sabit önceliklere sahiptirler.

Gerçek zamanlı bir haberleşme sistemi, farklı türdeki trafik akışları için gerekli zamanlama ihtiyaçlarını sağlamalıdır. Aşağıda, gerçek zamanlı haberleşme sistemlerinde olması istenen özellikler verilmiştir (Aras vd., 1994):

1. Jitter düşük olmalıdır.
2. Gecikme düşük olmalıdır.
3. Gerçek zamanlı olmayan trafik ile gerçek zamanlı trafik birbirlerine kolayca entegre olabilmelidir.
4. Ağ ve trafik şartlarının değişmesine dinamik olarak adapte olabilmelidir.
5. Ağın büyüklüğüne ve yüksek bağlantı sayılarına göre, performans kabul edilebilir sınırlar içinde kalmalıdır.
6. Band genişliği kullanımı efektif olmalıdır.

7. Ağ içinde ya da uç düğümlerdeki paket başına düşen işlem yükü (overhead) düşük olmalıdır.

Bu özellikleri ve gerçek zamanlı uygulamalar için gerekli koşulları sağlamak için, uygun protokoller tasarlanmış olmalıdır.

Ekonominin ve askeri uygulamaların hızlı bir şekilde gelişmesiyle, gerçek-zaman performansı; medikal cihazlar, uçak uçuş kontrolü, savunma sistemleri ve nükleer sistemler gibi çok kritik sistemlerde, en önemli problemlerden biri haline gelmiştir. Bu kritik sistemler için haberleşme ağının, katı gerçek-zaman davranışı sergilemesi beklenir. Her bir mesajın zaman sınırlamalarını karşılamak için zamanlama yapmak, bu tip güvenilir sistemler için oldukça önemlidir.

Gerçek-zamanlı bir haberleşme sisteminde iletim yapmak için mesajları zamanlamak, gerçek-zamanlı çoklu görev (multitasking) sistemindeki zamanlama görevlerinden farklıdır. Çünkü paket iletimi kesilirse, kaldığı yerden devam etmez, hepsinin yeniden gönderilmesi gerekir (Hui vd., 2008).

Bu tez çalışmasında kullanılan haberleşme protokolü, seri iletişime dayalı, çok yöneticili, çarpışma çözümleyicisine ve yüksek güvenlik özelliklerine sahip olan CAN (Controller Area Network) protokolüdür (Aysan vd., 2010). Kontrol ağlarında her bir istasyonun ağ üzerindeki diğer istasyonlarla gerçek zamanlı, hızlı ve hatasız haberleşmesi istenir. Aynı zamanda veri haberleşmesi esnasında sistemin güvenli olması, elektromanyetik gürültü gibi dış etkenlerden fazla etkilenmemesi, genişlemeye izin verir yapıda olması ve yönetilebilir olması beklenir.

CAN, yüksek performanslı ve yüksek güvenirliği olan gerçek zamanlı veri haberleşmesi için uygun bir protokoldür. Ağ performansını etkileyen anahtar faktör, CAN protokolünün doğasından kaynaklanan mesaj öncelik yapısıdır. CAN mesajlar, CAN veri yoluna önceliklerine bağlı olarak erişirler.

Bu tez çalışmasının önemi, CAN ağında mesajları önceliklendirme işleminin yapılarak, çevrim içi mesaj üretiminin ve iletiminin sağlanmasıdır. İstenen, tüm mesajların çevrim içi iletiminin en iyi şekilde gerçekleştirilmesidir. Yapay Sinir Ağları bu işlemleri bilgisayar ortamında modellemek için kullanılmıştır.

Çok karmaşık, uzun ya da çok sayıda düzensiz bilgi taşıyan verilerin çözümlenebilmesinde, insan algısının ya da var olan bilgisayar tekniklerinin sonuca ulaşmada başarılı olamayacakları benzer tüm işlemlerde, üstün yeteneklerinden dolayı Yapay Sinir Ağları (YSA) günümüzde sıklıkla kullanılmaktadır. Eğitilmiş bir sinir ağı, analiz etmesi için kendisine verilen bilgi kategorisinde bir 'uzman' olarak düşünülebilir. Bu uzman, daha sonra yeni ve tanımlanmamış durumlar için farklı çıkışlar sağlayabilir (Coşkun ve Yıldırım, 2005). YSA, insan beyninin sinir hücrelerinden oluşmuş katmanlı ve paralel olan yapısının tüm fonksiyonlarıyla beraber bilgisayar ortamına aktarılmasıyla oluşturulan ve böylece insan beyninin modellenmeye çalışıldığı sistemlerdir. Beyin fonksiyonlarını taklit edebilmek amacıyla değişik YSA modelleri geliştirilmiştir. YSA'lar öğrenme, ilişkilendirme, sınıflandırma, genelleme, özellik belirleme, optimizasyon gibi değişik problemlerin çözümünde kullanılabilmektedir (Öztemel, 2006).

Bu çalışmada kullanılan yapay sinir ağı mimarisi NARX'dir. NARX ağı, doğrusal olmayan dinamik sistemlerin giriş-çıkış modellemesi için yaygın olarak kullanılan dinamik bir yapay sinir ağı mimarisidir (Xie vd., 2009).

Bu tez çalışması, gerçek zamanlı veri iletişimi yapılan Kontrol Alan Ağı (CAN) için üretilen çevrim içi mesajların önceliklendirilerek, yapay sinir ağı ile modellenmesi aşamalarını içermektedir. Modellemenin önemi, oluşturulacak olan sistemin değişen koşullar altındaki davranışlarını incelemek, kontrol etmek ve geleceği hakkında varsayımlarda bulunmak amacıyla parametreler arasındaki bağlantıları, kelimeler ya da matematiksek terimlerle belirleyebilmektir.

2. KAYNAK ÖZETLERİ

Zuberi ve Shin (1995), mesaj kümesini yüksek hızlı ve düşük hızlı mesajlar olarak ayırmayı önermişlerdir. Uzun ID'lere gerek duyulmadan yüksek zamanlanabilirlik sağlayan MTS algoritması kullanmışlardır. Yaptıkları simülasyonlarda MTS algoritmasını, DM ve EDF algoritmaları ile karşılaştırarak, aşırı yük (overhead) ve sıkı deadline kuralları uygulanmadığı durumlarda MTS'nin performansını analiz etmişlerdir.

Lin vd. (1996), tekrarlayan sinir ağları için istenen çıktının zamanlardaki verilere bağlı olduğu problemler üzerinde yetersiz performans gösterdiği tespit ederek, NARX mimarisini önermişlerdir. NARX ağlarının geleneksel tekrarlayan sinir ağları kadar uzun süre bilgi tutabildiğini gösteren bazı deneysel sonuçlar sunmuşlardır. NARX ağlarının uzun vadeli bağımlılık sorununu atlamamasına rağmen, uzun vadeli bağımlılık problemleri üzerinde performansı artırabildiğini göstermişlerdir.

Zuberi ve Shin, (1997) bu çalışmalarında CAN için zamanlanabilirlik kullanımını artıran ve sabit öncelik planlarından daha iyi performansı olan MTS algoritmasını önermişlerdir. Simülasyon için, periyotları 20 ms ve son teslim süreleri 8 ms olan periyodik mesajlar ile son teslim süreleri 5 ms olan sporadik mesajlar seçilmiştir. Sonuçlar, ağ kullanımında %20'nin üzerinde fark olduğunu göstermektedir.

Livani vd. (1998) dinamik bir dağıtık gerçek-zamanlı sistemin gereksinimlerini karşılamak amacıyla, CAN bus kaynak için dağıtık zamanlamayı karşılamak üzere bir mekanizma sunmuştur. Burada göz önüne alınan temel sorunlar, çoklu yayın, HRT-SRT ve NRT sınırlamaları arasındaki ayrım, haberleşme kaynaklarının dinamik rezervasyonuna izin vererek, dinamik HRT hesaplamayı desteklemektir.

Natale (2000), bu çalışmasında CAN mesajlarının zamanlanmasında EDF tekniklerinin uygulanabilirliğini tartışmaktadır. Bu çalışma, etkili bir deadline

kodlama metodu ve garanti analizleri üzerindeki etkilerini anlatmaktadır. Sınırlandırılmış işlemci yüküne (overhead) rağmen (%5'den daha az CPU zamanı), önerilen EDF uygulaması (%20'nin üzerinde) uygun ağ iş yükünde artışa izin vermektedir. Çalışmada yapılan tüm simülasyonlar sonucunda EDF zamanlamanın, verimliliği kanıtlanmıştır.

Zuberi ve Shin (2000), MTS algoritmasını kullanarak, sabit öncelik planından daha iyi bir performans elde etmişlerdir. MTS algoritmasına göre, yüksek hızlı mesajlar için EDF ve düşük hızlı mesajlar için DM zamanlamaları kullanılmaktadır. EDF zamanlamasına göre, iki mesajın son teslim süresi kuantalamadan sonra ayrılabiliyorsa, daha erken son teslim süresi olan mesaj daha yüksek önceliklidir. Eğer mesajların aynı alanda son teslim süreleri varsa, DM önceliklerine göre sıralanacaklardır.

Pedreiras ve Almeida (2002), bu çalışmada FTT-CAN protokolünü baz alan EDF mesaj zamanlamasını sunmuştur. EDF zamanlama problemi CAN denetleyici kullanılarak çözülmüştür. Bu denetleyici hafızasında gönderilmeye hazır iki ya da daha fazla mesajın depolanmasına izin verir. Böylece en yüksek öncelikli mesaj, iletim için hazırdır.

Fuster vd., (2005) yaptıkları çalışmada EDF zamanlaması için CAN denetleyiciler kullanarak, farklı tipteki sensörler ile elektrikli bir aracın tahrik kontrolü arasındaki veriyi iletmek için CAN tabanlı bir iletişim sistemi kullanmışlardır. Bu çalışmada oluşturulan modelin simülasyonunda düşük öncelikli mesajlar, bus üzerinde hiç yüksek öncelikli mesaj kalmayıncaya kadar beklemektedir.

Oliveira vd., (2005) mesaj kaybını önlemek amacıyla mesaj oluşumları için ayrılmış toplam süreyi minimize etmek için bir algoritma önermişlerdir. Bu çalışmada, sabit önceliklendirme yapısı kullanılmıştır.

Tao vd., (2005) çalışmalarında fuzzy dinamik öncelikli zamanlama önermişlerdir. Bir mesajın servis kalitesi (QoS) gereksinimlerinden başka, mesaj

tanımlayıcı ve ağ durumunun geribesleme bilgisi de bir nonlineer öncelik kodlama metodu sağlamak için kullanılmıştır.

Lin vd., (2006), CAN ağındaki farklı tipteki mesajları zamanlamak için bir model sunmuşlardır. Bu mesajlar; katı gerçek-zamanlı (HRT) mesajlar, esnek gerçek-zamanlı (SRT) mesajlar ve gerçek-zamanlı olmayan (NRT) mesajlar olmak üzere üç tiptir. HRT mesajları en yüksek öncelikli, SRT mesajları daha düşük öncelikli ve NRT mesajları ise en düşük öncelikli olarak sınıflandırılmıştır. Bu çalışmada, yapay sinir ağlarının öğrenme kabiliyetinin avantajları alınarak, RBFN (Radial Basis Function Network) tarafından gerçekleştirilen bir haberci zamanlama denetleyicisi olan MSC (Message Scheduling Controller) sunulmuştur. MSC, gönderilecek olan uygun mesajı seçen bir mesaj göndericisi gibi davranır.

Anwar ve Khan (2007), bu makalede mevcut CAN tabanlı ağlarla entegrasyonu bozmadan bir CAN ağındaki dinamik olarak değişen öncelikleri desteklemek için bir dizi algoritma sunmaktadır. Mesajın içeriğini tanımlayan tek bir ID kullanmak yerine mesajı tanımlamak için zaman ayarlı bir mesaj ID penceresi atanan bir şema sunulmaktadır. Ayrıca mesajın önceliği, mesajın içeriği ya da anlamı üzerinde herhangi bir etki yaratmadan çalışma zamanında ayarlanmıştır.

Duzenek vd.,, (2007) periyodik olarak gönderilen mesajların zamanlamasını optimize ederek, mesaj tepki süresini azaltmayı amaçlamışlardır. Bu çalışmada olay tetiklemeli CAN protokolü kullanılmış ve aynı zamanda gönderilen mesaj sayısı azaltılarak, mesaj kuyruk süresi azaltılmıştır.

Davis vd. (2007), CAN zamanlaması ve tepki süresi analizleri yapmışlardır. Bu makale, hatalı programlanabilirlik analizi kullanılarak tasarlanan ve geliştirilen ticari CAN sistemleri üzerindeki olası etkilerini tartışmakta ve CAN programlanabilirlik analiz araçlarının revizyonu için önerilerde bulunmaktadır.

Chen ve Yen (2012), çalışmalarında CAN ağında değişen koşullara dinamik olarak cevap veren çevrim içi adaptif MSC zamanlamayı önermişlerdir. CAN'daki mesaj zamanlama problemleri çeşitli sınıflarda ele almışlardır.

Puiu vd., (2010), bir CAN ağından mesajın iletim süresini kontrol etmek için kullanılabilecek bir dinamik mesaj zamanlama tekniği sunmaktadır. Simülasyon ortamı için MatLab kullanılmıştır. Bu çalışmada standart CAN çerçeve modeli kullanılmıştır, fakat geliştirilen model genişletilmiş çerçeve formatına uygulanabilmektedir.

Shi ve Zhang (2012) bu çalışmalarında, CAN veriyolundaki mesaj iletiminin dinamiklerini tanımlayan bir analitik model geliştirmişlerdir. Bu analitik zamanlama modeline dayanarak, mesaj akışlarının çevrim içi ayarlarının varlığında CAN veri yolunun programlanabilirliğini etkin bir şekilde kontrol eden bir çevrim içi test önermişlerdir. Simülasyonlar, çevrim içi testin CAN veriyolundaki programlanabilirlik kaybını doğru bir şekilde rapor edebildiğini göstermektedir.

Saadi (2013) bu çalışmada, iletilen mesajların son teslim sürelerini geçmeyeceğini garanti etmek için gerçek zamanlı mesaj iletim programlaması sırasında CAN protokolünün mekanizmasını göstermiştir. Analiz edilen dinamik mesaj aktarım yöntemi, birçok yazar tarafından zaman mesajı iletimi fikrini desteklemek için önerilmiş olan mesaj önceliğine dayanmaktadır.

Labde vd., (2017) yaptıkları çalışmada herhangi bir stoğun kapanış fiyatını tahmin etmek için adaptif bir NARX sinir ağına dayanan bir model önermişlerdir. Bu, girişteki gecikmeleri ve bellek yuvaları olarak hareket eden çıkışı kullanan ve böylece tahmin doğruluğunu arttıran doğrusal olmayan otomatik regresif bir eksojen giriş modelidir. Bu model kapanış fiyatını analiz etmek ve tahmin etmek için bir zaman serisi yaklaşımı kullanmaktadır. Ağı eğitmek için Levenberg-marquardt algoritması kullanılmıştır. Ağın doğruluğu, ortalama karesel hata yardımı ile belirlenir. Bu modelde, azaltılmış giriş gecikmelerinin yanı sıra tahminler yapmak için bir kapalı bir döngüden faydalanmışlar ve her bir NARX sinir ağının çalışmasını analiz etmek ve optimum konfigürasyonu belirlemek için doğruluğu tespit etmişlerdir.

3. KONTROL ALAN AĞI (CAN)

CAN, endüstriyel ortamlardaki birçok dağıtık kontrol uygulamalarında kullanılan gerçek zamanlı haberleşme sistemidir. Başlangıçta otomotiv endüstrisinde kullanılmak üzere tasarlanmış olsa da, günümüzde birçok endüstriyel kontrol uygulamalarında yaygın olarak kullanılan gerçek zamanlı bir veri haberleşme sistemidir. CAN'ın kullanıldığı değişik ağ uygulamalarına örnek olarak:

- Medikal cihazların kontrolü,
- Akıllı motor kontrolü,
- Havadaki ve sudaki yabancı kimyasal maddelerin tespiti,
- Robot kontrolü,
- Akıllı sensörler,
- Uzay araştırmaları,
- Asansörler,
- Akıllı binalar,
- Depo, sera, otopark ve laboratuar otomasyonu,
- Sel baskını ve yangın tespiti,
- Trafik denetimi,
- Doğal afet yönetimi,
- Boru hattı, çevre, sualtı ve sınır güvenliği

gösterilebilir.

3.1. CAN'ın Tarihçesi ve Gelişimi

CAN, araç ağları için basit, etkili ve sağlam haberleşme sağlamak için tasarlanmış bir seri haberleşme protokolüdür. CAN, Robert Bosch tarafından 1983 yılında geliştirilmiş ve 1986'da SAE (Society of Automotive Engineers) kongresinde sunulmuştur. 1987 yılında, Intel (82526) ve Philips (82C200) tarafından ilk CAN denetleyici çipleri üretilmiştir. 1990'ların başında, Bosch'un standardizasyon için CAN2.0A spefikasyonunu açıklamasıyla, asıl atılım gerçekleşmiştir. 1993'de CAN için ilk ISO standardı (11898) yayınlanmıştır. Mercedes, 1991'de CAN'ı ilk kullanan otomotiv imalatçısıdır.

1990'ların ortasında, otomotiv elektroniğinin karmaşıklığı, hızlı bir şekilde artmaktaydı. Mercedes, BMW, Audi ve VW arabalarda, ağa bağlı olan Elektronik Kontrol Birimlerinin (ECU) sayısı 1990'ların başında 5 ya da daha azken, milenyumun sonunda yaklaşık 40 oldu. Bu artışla birlikte, geleneksel noktadan noktaya kablolamada yüzlerce ayrı bağlantı ve onlarca kilo bakır kablo kullanılması, üretimi ve kurulumu için gerekli maliyet artarak çoğaldı.

Sonuçta CAN, maliyetin en düşük düzeyde tutulması bilinciyle, pek çok üretici tarafından otomotiv endüstrisinde hızlı bir şekilde benimsenmiştir. Bunun da sonucu olarak, CAN düğümlerinin (tümleşik CAN yan birimli 8, 16 ve 32 bit mikrodenetleyiciler) satışları, 1999'da 50 milyonun altındayken, 2003'te 340 milyonun üstüne yükselmiştir. 2004 yılında, en az 15 silikon üreticisi, toplamda 50'nin üzerinde farklı tümleşik CAN kapasiteli mikroişlemci ailesini üretmiştir.

Bugün Avrupa'da üretilen neredeyse tüm arabalar, en az bir CAN bus ile donatılmıştır. Birleşik Devletler'de, Çevre Koruma Ajansı (EPA), CAN'ın kullanımını, 2008 yılından beri göstergeden arıza tespiti için, tüm arabalarda ve kamyonetlerde şart koşmuştur.

90'lı yıllarda CAN'in gelişimi sonucu, CiA kısaltmasıyla adlandırılan "CAN in Automation" kurulmuştur. Bu bağımsız grup, CAN protokolün uluslararası standartlaştırması, tanıtımı ve geliştirilmesinden sorumludur. Kar amacı gütmeyen bu kurum, Almanya'nın Nuremberg kentinde kayıtlıdır. Birkaç şirket tarafından kurulan CiA, bugün CAN kullanıcılarını ve üreticilerini kapsayan 640 üyeye sahiptir (CAN-CiA, 2018). Bu sayı gün geçtikçe artmaktadır.

1995 yılında CAN2.0B geliştirilerek, bir CAN sistemine bağlı ünite sayısı 500 milyona çıkarılmıştır. 1996'da CANopen geliştirilmiş ve böylece CAN'ın uygulamada kullanılabilirliği daha da arttırılmıştır.

CAN protokolü, yüksek hızlı uygulamalar için ISO 11898 ve düşük hızlı uygulamalar için ISO 11519 uluslararası standardıyla tanımlanmıştır (Pedreiras ve Almeida, 2002). ISO 11898 standardı, ISO 11898-2, ISO 11898-5 (düşük güç

modu) ve ISO 11898-6 (seçmeli uyandırma) olmak üzere üç standarda bölünmüştür. Bu üç standart bir araya getirilerek ISO 11898-2: 2016 olarak yayınlanmıştır. Bu norm, ayrıca 1 Mbit/sn'nin üzerindeki bit hızları için ek dinamik parametreleri belirtir (Hell, 2016).

3.2. CAN Özellikleri

Kontrol alan ağının genel özellikleri aşağıda maddeler halinde verilmiştir:

- Bus topoloji kullanır.
- Yüksek hızlı seri arabirim sağlar.
- Düşük maliyetli ve esnek bir fiziksel ortam sunar (Yeni düğüm ekleyip, çıkarmak kolaydır).
- Yük büyüklüğü kısadır.
- Tepki zamanları hızlıdır.
- Yüksek hata bulma ve düzeltme seviyesi vardır. (Hata düzeltme mekanizmaları ile tespit edilemeyen frame sayısı 4.7x10 e-11 dir.)
- Öncelikli mesajlar kullanır. (Yüksek öncelikli mesajlar, 134 mikro saniyeden daha kısa bir sürede iletilebilir.)
- Elektromanyetik girişime karşı dayanıklığı yüksektir.
- Konfigürasyonu esnektir.

CAN, farklı sistemler için farklı veri iletim hızları kullanır. CAN düğümler, 1 Mbit/s veri iletim hızı ile 40 m ve 40 Kbit/s veri iletim hızı ile 1000 m'lik bir veri yolu üzerinden bağlanabilirler (Farsi vd., 1999). Haberleşme ortamı için kullanılacak kablo UTP, fiber optik ya da koaksiyel kablo olabilir. Fiber optik kablolar, elektromanyetik girişimden etkilenmeseler de otomotiv endüstrisindeki düşük maliyet gereksinimlerinden dolayı fizibil değildir.

CAN, NRZ5 kodlamasını kullanarak veri iletir. NRZ5 kodlamada, aynı polaritede beşten fazla ardışık bit oluşmasını engelleyen bir bit doldurma tekniği kullanılır (Nyce, 2016).

Bir CAN ağına yeni bir bağlantı noktası eklemek için donanımsal ya da yazılımsal bir değişikliğe ihtiyaç yoktur. Bu özellik, CAN'a modüler bir yapılanma şansı tanımaktadır. Çünkü verinin kim tarafından üretildiği CAN ağı içindeki istasyonlar için önemli değildir (Berkay vd., 2003).

3.2.1. CAN Katmanları

CAN mimarisi, 3 katmandan oluşmaktadır. Bu katmanlar, OSI (Open Systems Interconnection – Açık Sistemler Bağlantısı) referans modelinin Fiziksel, Veri Bağı ve Uygulama katmanlarına karşılık gelmektedir.

CAN uygulama katmanında, DeviceNet, CAN-Kingdom ve SDS (Smart Distributed System) gibi yüksek seviyeli protokoller çalışır.

CAN spesifications 2.0 Part A'da Veri Bağı katmanı, Nesne ve Transfer alt katmanları olarak ikiye ayrılmışken, Part B'de ise, LLC ve MAC alt katmanları olarak ayrılmaktadır. MAC alt katmanı, mesaj çerçeveleme, hakemlik, onaylamalar, hata denetimi ve işaretleşmeden sorumludur. LLC alt katmanının görevi ise; mesaj filtreleme, aşırı yüklenme uyarısı ve geri alma işlemlerini gerçekleştirmektir (Bosch, 1991).

Fiziksel katman, elektriksel ve mekaniksel karakteristikleri tanımlar. Bit zamanlaması, bit kodlaması ve senkronizasyonu bu katmanda gerçekleştirilir.

3.2.2. CSMA/CR

CAN protokolü, çarpışma denetimi için CSMA (Carrier Sense Multiple Access / Çarpışma Denetimi ile Taşıyıcı Algılamalı Çoklu Erişim) erişim mekanizması esasına göre çalışan CSMA/CR (Carrier Sense Multiple Access with Collision Resolution / Çarpışma Çözümü ile Taşıyıcı Algılamalı Çoklu Erişim) kullanır (Anwar ve Khan, 2007). İletime geçmek isteyen düğüm, önce hattın (bus) boş olup olmadığını kontrol eder. Eğer ağda iletim yoksa düğüm mesajı göndermeye başlar. Veri iletim yolu, bağlı olan tüm düğümlerin veri aktarımına açık olduğu

için aynı anda farklı düğümler iletime geçmeye çalışırsa çarpışma meydana gelir. Çarpışma sonrası düğümler, sıkışma (jam) sinyali göndererek, geri çekilirler.

Çarpışma durumunda tüm veriler bozulur ve yeniden aktarılması gerekir. Bu durum ağ performansını olumsuz yönce etkileyerek fazladan ağ trafiği ve gecikme oluşmasına neden olur. Bu nedenle veri gönderen bir düğümün aktarım sonrası hattı dinlemesi ve olası çarpışmaların farkına varması gerekir.

CSMA/CR erişim modelinde, her bir mesaj, tek ve benzersiz bir öncelik ile karakterize edilir. Aynı anda iki ya da daha fazla düğüm iletime geçerse, çarpışma oluşmasını önlemek için, yüksek öncelikli mesaj iletime devam ederken, diğeri yol boşalana kadar bekler. Bu sayede bir kontrol ağında var olan düğüm sayısı kadar farklı öncelik değeri verilebilir.

3.3. CAN Çerçeveleri

Bir CAN bus üzerinde mesajlar, çerçevelerle iletilir. CAN ağı iki farklı çerçeve (frame) formatında konfigüre edilebilir:

1. Standart ya da temel çerçeve formatı (CAN 2.0 A)
2. Genişletilmiş çerçeve formatı (CAN 2.0 B)

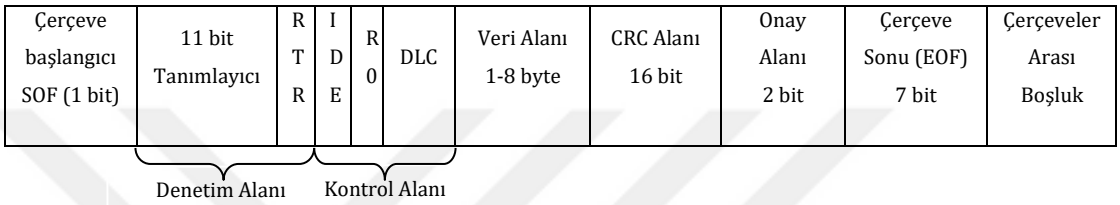
İki çerçeve formatı arasındaki tek fark, standart CAN çerçevesinin 11 bitlik tanımlayıcıyı desteklerken, genişletilmiş CAN çerçevesinin 29 (11+18) bitlik tanımlayıcıyı desteklemesidir (Bosch, 1991).

CAN çerçevelerinin içinde gönderici ya da alıcı adresleri yoktur. Onun yerine mesaj içeriğini tanımlayan bir tanımlayıcı alan kullanırlar. Alıcı istasyon bu tanımlayıcı alana bakarak mesajın kendisine gelip gelmediğini anlar. Mesajın kimden geldiğinin önemi yoktur. 11 bitlik tanımlayıcı ile 2048 farklı istasyon adreslenebilirken, 29 bitlik tanımlayıcı ile 536 milyondan fazla farklı istasyon adreslenebilmektedir. Standart çerçeve formatı daha yaygın olarak kullanılırken, genişletilmiş çerçeve formatı endüstride kamyon ve tır gibi

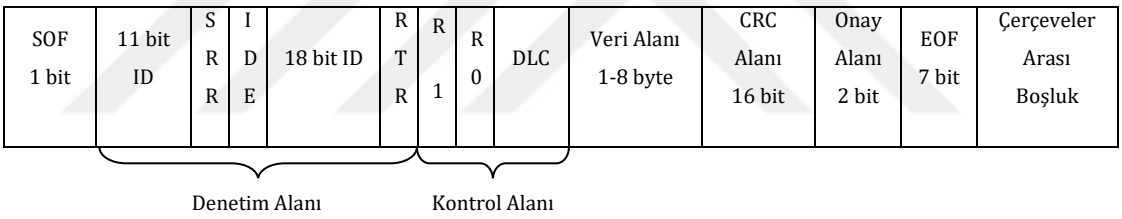
araçların elektronik aksamlarının haberleşmesinde ve CAN protokolü tabanlı yüksek seviyeli protokollerin geliştirilmesinde kullanılmaktadır.

3.3.1. CAN Çerçeve Alanları

Standart ve Genişletilmiş CAN çerçeveleri, 8 alandan oluşmaktadır. Şekil 3.1'de Standart CAN çerçeve (CAN 2.0 A) alanları, Şekil 3.2'de ise Genişletilmiş CAN çerçeve (CAN 2.0 B) alanları görülmektedir (Farsi vd., 1999).



Şekil 3.1. Standart CAN çerçeve (CAN 2.0 A) alanları



Şekil 3.2. Genişletilmiş CAN çerçeve (CAN 2.0 B) alanları

Çerçeve Başlangıcı (SOF - Start of Frame) Alanı: CAN mesajının başlangıcını belirtir. 1 bit uzunluğundadır.

Denetim Alanı (Arbitration Field): Standart veri çerçevesinde 12 bit, genişletilmiş veri çerçevesinde 32 bittir. Standart biçimde, Tanımlayıcı (ID) alanı ve RTR (Uzak İletim İstek) alanlarından oluşur. ID alanı, mesajların iletim önceliğini ve kabul edilip/edilmemesini belirlemek amacıyla kullanılır. CAN bus üzerindeki tüm mesajların kendine özgü bir tanımlayıcısı olmalıdır. Daha düşük nümerik değere sahip olan tanımlayıcı ID, daha yüksek iletim önceliklidir. Çarpışma durumunda, farklı düğümler, mesajların tanımlayıcılarını karşılaştırır. ID alanındaki ilk 7 bit ardışıl olarak çekinik olamaz.

1 bit büyüklüğündeki RTR alanı, bir düğümün diğer bir düğümden bilgi istemesi durumunda haberleşmeyi başlatmak için kullanılır. RTR biti gönderilen CAN çerçevesinin veri ya da uzak çerçeve olup olmadığını gösterir. RTR lojik '0' (dominant) ise veri çerçevesi, lojik '1' (çekinik) ise uzak çerçevesidir ve veri alanı yoktur.

Kontrol Alanı (Control Field): Mesajın tipi bilgisini içerir. Bu alanın ilk 1 biti IDE Alanını (Tanımlayıcı Uzantı), sonraki biti ileride kullanım için rezerve edilmiş 1 bitlik R0 alanını ve sonraki 4 bitlik gönderilen verinin boyutunu gösteren veri uzunluk kodunu (DLC) kapsayarak, toplamda 6 bitten oluşur. IDE lojik '0' ise gönderilecek herhangi bir tanıtıcı bilgisi olmadığını gösterir.

Veri Alanı (Data Field): Veri alanı, iletilecek veriyi içerir. DLC değerine bağlı olarak 0 ile 8 bayt arasında değişen uzunluğa sahiptir.

Çevrimli Fazlalık Sınama Alanı (Cyclic Redundancy Check, CRC): CAN protokolü, bilgi aktarımının doğruluğunu kontrol etmek için, her veri ve istek çerçeveleri ile birlikte iletilen, 15 bit CRC dizisi artı 1 bit CRC belirteci kullanır (Rufino ve Verissimo, 1995). CRC bölgesi toplam 16 bitten oluşur. Bu alana mümkün olan iletim girişimlerini sezmek için başlangıç biti, denetim alanı, kontrol alanı, veri alanı ve CRC alanlarını kapsayarak hesaplanan bir kontrol kod yazılır. Bir mesajın gönderimi tamamlandıktan sonra CRC alanı kontrol edilir. Eğer çerçevede herhangi bir hata sezilmişse, tüm çerçeve yeniden gönderilir.

ACK Alanı (ACKnowledgment Field): Kabul onayı için kullanılır. 1'er bitlik ACK slot ve ACK belirtici alanlarından oluşur (Farsi vd., 1999). Bu alan, mesajın alınıp alınmadığı ve herhangi bir hatanın sezilip sezilmediği hakkında gönderici düğümü bilgilendirir. Gönderici düğüm, ilk bitini resesif olarak gönderir. Eğer veri en az bir alıcı tarafından doğru alınmışsa alıcı yola dominant biti yazar. Böylece gönderici mesajın en az bir alıcı tarafından alındığını anlar. Eğer gönderici dominant biti okuyamazsa, ACK işaretinden kaynaklı bir hata olduğuna karar verir ve veriyi tekrar yollar. Bu alanın ikinci biti ise ACK delimiter olarak adlandırılır ve resesiftir.

Çerçeve Sonu (End of Frame) Alanı: Veri ve uzak çerçevelerinin sonlandığını belirtir. Bu alan 7 bitten oluşur ve buradaki tüm bitler çekiniktir.

Çerçeveler Arası Boşluk (Inter Frame Space, IFS): 3 bitten oluşur. Çerçeveler arasında boşluk bırakarak, bir çerçeveyi onu takip eden sonraki çerçeveden ayırmak için kullanılır. İletimi senkronize ve kontrol etmek için veri çerçeveleri arasında minimum 3 bitlik bir IFS gereklidir. Aksi takdirde hata çerçeveleri ya da aşırı yükleme çerçeveleri, çerçeve sonu belirtecinden hemen sonra başlayabilir. IFS'den sonra veriyolu yeni bir iletime kadar boş durumdadır. Bu alandaki tüm bitler çekiniktir.

Genişletilmiş çerçevede ise standart çerçevedeki alanlara ek olarak RTR bitinin işlevini yerine getiren SRR (Yedek Uzak İstek) biti ve ileride kullanım için ayrılmış R1 biti eklenmiştir. SRR, o anda gönderilmekte olan çerçevenin standart bir çerçevemi yoksa genişletilmiş bir çerçevemi olduğunu belirtir. Genişletilmiş CAN çerçevesinde SOF ve 11 bitlik standart tanımlayıcının gönderilmesinden sonra SRR biti yer alır ve devam eden mesajın, bir genişletilmiş CAN çerçevesi olduğunu belirtir, ardından da 18-bitlik genişletilmiş tanımlayıcı gelir. SRR biti aynı zamanda standart CAN çerçevesi ile genişletilmiş CAN çerçevesi arasında öncelik belirlenmesi için kullanılır.

Bir CAN ağında, standart ve genişletilmiş çerçeve kullanan iki farklı istasyon, aynı anda iletime geçmek isterse, öncelik standart CAN çerçevesi kullanan istasyonundur.

3.3.2. CAN Çerçeve Yapıları

Kontrol Alan Ağlarında kullanılan Veri Çerçevesi, İstek (Uzak) Çerçevesi, Hata Çerçevesi ve Overload (Aşırı Yükleme) Çerçevesi olmak üzere dört farklı çerçeve tipi vardır (Gao ve Li, 2011).

1. Veri Çerçevesi: Veriyi içeren mesaj çerçevesidir. Veri çerçevesi alanları Şekil 3.3'de gösterilmiştir.

Çerçeve başlangıcı (1 bit)	Hakemlik Alanı (12 ya da 32 bit)	Kontrol Alanı (6 bit)	Data Alanı (1-8 byte)	CRC Alanı (16 bit)	Onay Alanı (2 bit)	Çerçeve Sonu (7 bit)

Şekil 3.3. Veri Çerçevesi Alanları

2. İstek (Uzak) Çerçevesi: Kontrol ağındaki bir istasyon başka bir istasyondan veri istediği zaman yayınlanan mesaj çerçevesidir. Uzak çerçeve alanları Şekil 3.4'de gösterildiği gibidir.

Çerçeve başlangıcı (1 bit)	Hakemlik Alanı (12 ya da 32 bit)	Kontrol Alanı (6 bit)	CRC Alanı (16 bit)	Onay Alanı (2 bit)	Çerçeve Sonu (7 bit)

Şekil 3.4. Uzak Çerçeve Alanları

3. Hata Çerçevesi: Veri ya da İstek Çerçevesi gönderilirken ya da alınırken hata oluştuğunda, gönderen ya da alıcılar tarafından ne tip hatanın olduğunu belirten mesaj çerçevesidir. Hata çerçevesi iki alandan oluşmaktadır. Bu alanlar hata bayrakları (Error Flags) ve hata belirteci (Error Delimiter) alanıdır. Aktif ve pasif olmak üzere iki adet 6'şar bitten oluşan hata bayrağı vardır. Yolun durumuna göre aktif ya da pasif hata oluşturulmaktadır. Hata belirteci 8 bitten oluşur. Hata çerçevesi alanları Şekil 3.5'de görülmektedir.

Şekil 3.5. Hata Çerçevesi Alanları

4. Overload (Aşırı Yükleme) Çerçevesi: Ağdaki istasyon mesaj işlerken başka bir mesaj gelmesi durumunda yayınlanan mesaj çerçevesidir.

3.4. CAN Hata Mekanizması

CAN, özellikle katı gerçek-zamanlı haberleşme gereksinimleri olduğunda, sıklıkla kullanılan haberleşme protokollerinden biridir (Shokry, vd., 2009). CAN ağlarının çok verimli bir hata denetim mekanizmaları vardır. Hatayı düzeltme süresi (yeni bir çerçevenin olası başlama zamanı ile hatanın tespiti arasında) 17 ile 31 bit süresidir (Navet vd., 2000).

CAN ağında hata tespit mekanizmaları çok etkin bir şekilde çalışır. CAN bus'ta mesaj iletimi esnasında bir arıza ya da hata oluşursa, mesajın yeniden iletimi için bir hata bayrağı (error flag) sinyali gönderilerek, iletim iptal edilir. Böylece mesajın başka bir istasyon tarafından alınması ve aynı zamanda ağ içerisindeki veri doğruluğunun bozulması engellenir. Bir mesaj yeniden iletildiğinde, diğer mesajların iletiminde aksamaya sebep olmamalıdır. Eğer arıza oluşmazsa, mesaj olağan olarak iletilir ve zaman payı kullanılmaz (Oliveira, vd., 2003).

CAN üzerinde ARQ (Automatic Repeat Request) yöntemi çalışır. Bu yönteme göre, eğer mesaj hatasız bir şekilde hedefe ulaşırsa bir doğrulama mesajı (ACK) ile kaynağa onayı gönderilir. Mesaj iletimi başarısız ise, kaynağa bir olumsuz doğrulama mesajı (NAK) gönderilerek, yeniden gönderilmesi talep edilir. ARQ yöntemi ağ içerisindeki veri doğruluğunun bozulmasını engeller.

CAN protokolü kullanılarak gerçekleştirilen haberleşme ortamlarında karşılaşılabilecek beş çeşit hata yapısı vardır. Bunlar:

- Bit Hatası
- CRC (Çevrimli Fazlalık Sınaması) Hatası
- Onay (ACK) Hatası
- Çerçeve Formatı Kontrolü
- Bit Doldurma (Bit-Stuffing) Hatası

3.4.1. Bit hatası

Hattaki tüm düğümler, veriyolunu izler. İzleme bit bit yapılır. Eğer gönderici bir düğüm, dominant bir bit gönderdiğinde hatta resesif bir bit algılarsa, ya da resesif bir bit gönderip dominant bir bit algılarsa bit hatası oluşur. Bu durumda, bir hata çerçevesi üretilir.

3.4.2. CRC hatası

CRC, (Çevrimli fazlalık sınaması) bir yığın veri için, veri gönderilmeden önce hesaplanır ve veri yığını ile birlikte gönderilir. Alınan veri üzerine yeni bir CRC hesaplanır ve gelen ile karşılaştırılır. Eğer CRC'ler uyuşmaz ise, iletimde bir hata meydana gelmiş demektir. Buna CRC hatası denir.

CRC kontrolü, bit-doldurma gibi, sadece veri ve istek (remote) çerçeveleri üzerinde gerçekleştirilir (Rufino, 1995).

3.4.3. Onay (ACK) kontrolü

Kaynak tarafından bir bilgi çerçevesi yollandıktan sonra hedeften çerçevenin doğru olarak ulaşıp ulaşmadığına ilişkin bir onay (ACK) mesajı gelinceye kadar bekler. Gönderdiği çerçevenin hatasız alınması durumunda, bir sonraki çerçeveyi gönderir.

3.4.4. Çerçeve formatı kontrolü

Mesaj çerçevelerinin yapısında bir bozukluk ya da uyumsuzluk varsa çerçeve hatası oluşur ve hata mesajı üretilir.

3.4.5. Bit doldurma

Gönderici veri alanında art arda beş tane 1 bitinden sonra, bir adet 0 biti eklenir. Buna doldurulmuş bit (stuffed bit) denir (Öner, 2003). Bit doldurma işlemi

gönderici tarafında yapılır ve alıcı tarafında eklenen bitler çıkarılır (Kumar ve Ramesh, 2016).

3.5. CAN Haberleşmesi

CAN protokolü, adres bazlı bir protokol olmayıp, mesaj bazlı bir protokoldür. Gönderilen tüm mesajlar, ağda bulunan diğer tüm ağ düğümlerine broadcast (yayın) şeklinde yayılır (Bosch, 1991; Davis vd., 2007).

CAN, OSI modelinin ikinci katmanı olan Veri Bağı katmanında çalışır. Bu katmanda kullanılan veri yapısı çerçeve olduğu için, CAN ağında veriler çerçeveler ile gönderilir ve alınır. Gönderilen veride, mesajın kaynak ve hedef adreslerini yoktur. Onun yerine, her bir mesaj, ağ üzerinde tek ve benzersiz olan bir tanımlayıcı (ID) tarafından etiketlenir. Bu tanımlayıcılar mesaj içeriğini tanımladıkları gibi, aynı zamanda mesajların önceliklerini de gösterirler. CAN bus için yarışan mesajlar bu önceliklere göre haberleşmeye başlarlar. En düşük tanımlayıcı değerine sahip olan mesaj, iletim önceliğini alır (Tindell vd., 1994).

CAN mesajları önceliklendirildiğinde, en yüksek önceliğe sahip olan mesaj çarpışma meydana geldiğinde kesintiye uğramadan iletilirken, daha düşük öncelikli mesajın iletimi sonlandırılır ve ağ boşaldığında yeniden iletime geçilir. (Wen vd., 2007). Öncelikler sabit ya da değişken olabilir.

Veri yolu üzerinde haberleşmek isteyen düğümlerin veri yolunun kullanımı konusunda anlaşmazlığa düşmemeleri için bit hakemliği kavramı geliştirilmiştir. Hakemlik mekanizmasında, mesajlar gönderilirken, öncelik sıralaması yapılmakta ve böylece, yüksek önceliğe sahip olan mesaj gecikme olmadan gönderilebilmektedir.

Bit seviyesinde hakemlik işleminin gerçekleşebilmesi için lojik durumların dominant (baskın) ve çekinik (resesif) olarak tanımlanması gerekir. CAN'da, 0 biti dominant, 1 biti ise çekinik olarak tanımlanır. Bus üzerinde aynı anda farklı

düğümlerden 0 ve 1 biti gelmesi durumunda 0 biti, 1 bitine karşı baskın gelir. Yani, bit dominant ise, her zaman hakemlik yarışını kazanır (Tindell vd., 1994).

İletime geçmek isteyen bir düğüm, iletim hattını sürekli olarak dinler. Eğer kanalı boş bulursa, bir SOF (Start of Frame) biti yayınlayarak hakemlik evresine başlar. Bu noktada, gönderilecek mesajı olan her düğüm, mesajın tanımlayıcı bitlerini kanala iletmek için yarışabilir. Tanımlayıcı, iletilecek mesajın ilk kısmıdır (Natale, 2000).

Tanımlayıcı bitleri arasındaki çarpışmaların bir mantıksal "VE" işlemi ile çözüldüğü farz edilir. Eğer bir düğüm, bir değişiklik olmadan bus üzerindeki öncelik bitlerini okuyorsa, yarışmanın galibi olduğunu anlar ve iletim haklarını alır. Eğer bu bitlerden biri bus'tan okunurken daha yüksek öncelikli bir mesaj bus'a girerse, bu durumda mesaj geri çekilir.

Gönderilen mesaj, iletim hattı üzerindeki her düğüme ulaşır. Sadece bu mesajla ilişkisi olan düğümler mesajı okuyup işler. Eğer yol boşaldığında birden fazla düğüm mesaj göndermeye başlarsa düşük ID'li mesaj yolu ele geçirir ve diğer düğümler aradan çekilerek tekrar göndermek üzere yolun boşalmasını beklerler.

3.6. CAN Ağlarında Güvenlik

CAN düşük seviyeli bir protokoldür ve özünde herhangi bir güvenlik özelliğini desteklemez. Uygulamaların, kendi güvenlik mekanizmalarına göre çalışması beklenir. Eğer bus üzerindeki mesaja dışarıdan saldırı olursa, hata oluşabilir. Kontrol birimi yazılımını değiştirebilen şifre mekanizmaları veri transferi için mevcuttur, fakat genellikle standart iletişim için kullanılmaz.

4. ZAMANLAMA ALGORİTMALARI

Mesaj zamanlamasının görevi, iletim hattı üzerindeki mesaj iletiminin nasıl kontrol edileceğini belirlemektir. Zamanlama, iletim hattı üzerinde bekleyen işlemlerden hangisinin çalışacağına karar veren mekanizmadır. Bu kararı vermek için çeşitli algoritmalar tanımlanmıştır.

Güvenilir zamanlama davranışları, gerçek-zaman gereksinimleri olan kontrol sistemleri için çok önemlidir. Sistem geliştiricileri, uygun zamanlama mekanizmaları kullanarak, tasarım aşamasında, zamanlama davranışlarını planlayabilirler (Michta, 2005). Gerçek-zamanlı mesajların zamanında tamamlanabilmesi için, özellikle planlanması gerekir (Zuberi ve Shin, 2000).

Zamanlama, tahmini kaynak kullanımını sağlar ve görevlerin en kötü durum tepki zamanları ile son teslim süreleri arasındaki ilişkileri kurar. Zamanlama teorisi, gerçek-zaman gereksinimleri olan tasarlanmış bir sistemin kendi son teslim süreleri ile karşılaşıp, karşılaşmayacağı tahmini için iyi kurulmuş bir yol sağlar (Michta, 2005).

Zamanlama algoritmalarının amaçları şunlardır:

- İsteklere hızlı bir şekilde cevap verilmelidir.
- Kullanıcının beklentilerini karşılamalıdır.
- Son teslim süresine (deadline) riayet edilmelidir.
- Veri kaybından sakınılmalıdır.
- Sürecin (işlemin) sunumu ve sonlanması arasındaki süre minimum olmalıdır.
- Üretilen iş maksimum olmalıdır.

Zamanlamada görevler, periyodik, aperiyodik ya da sporadik olabilirler. Periyodik görevler, belli sabit zaman aralıklarında kendini tekrar ederken, aperiyodik görevler belli bir zaman aralığı olmadan, herhangi bir anda gerçekleştirilebilir. Aperiyodik görevler, sistem tamamlanıncaya kadar sadece bir defa aktive edilirken, periyodik görevler defalarca gerçekleştirilir ve art arda

gelen iki aktivasyon arasındaki gecikme, periyot değerine eşit sabit bir değerdir. Sporadik görevler ise periyodik görevler gibi defalarca aktive edilen görevlerdir. Fakat görevler arasında sabit zaman aralıkları yoktur.

Zamanlama politikaları üç sınıfta incelenmektedir (Schmidt ve Schmidt, 2007).

1. Öncelik-temelli (olay-tetiklemeli) zamanlama

   Genellikle gerçek zamanlı uygulamalarda kullanılan öncelik temelli zamanlama algoritmalarında, süreçlerin yönetimi, öncelikler gözetilerek yapılabilir. Değişen şart ve olaylara adapte olabilen, daha dinamik gerçek-zamanlı sistemler için kullanılır. Her sürece bir öncelik değeri atanır ve çalışabilir süreçlerden yüksek öncelikli olanların çalışmasına izin verilir. Yüksek öncelikli süreçlerin sonsuza kadar çalışmasını engellemek için, her saat sinyalinde sürecin önceliği düşürülür. Bu işlem, sürecin önceliğini en yakın rakibinin önceliğine getirdiğinde süreçler arasında geçiş işlemi yapılır. Alternatif olarak her sürece en yüksek öncelik değeri verilir. Kuantumu bittiğinde önceliği azaltılır ve kendinden sonraki sürecin çalışmasına izin verilir. Zamanlama kararları, önceliklere göre verilir. Öncelik-temelli algoritmalar, olay-temellidir (event-driven).

   Bu tip zamanlamada, işler sıraya konur ve her bir olayda, hazır olan en yüksek öncelikli iş çalıştırılır. İşlerin öncelik sıralarına atanması, sonsuz önceliklendirmeye (preemption) izin verilip verilmeyeceği gibi kurallar ile birlikte, tam bir öncelik zamanlama algoritması tanımlar.

   Gerçek-zamanlı öncelik zamanlaması, öncelikleri son teslim süresi ya da bazı zamanlama kısıtlamalarını baz alarak atar. Görevler periyodiktir. Aperiyodik ya da sporadik görev yoktur. Zamanlama kararları, yapılan iş biter bitmez, hemen verilir.

   Öncelik-temelli zamanlamanın avantajları;

   - Daha az önbilgiye ihtiyaç duyduğu için, değişen zaman ve kaynak gereksinimleri olan uygulamalar için daha uygundur.

- Çalışma süresi yükü (runtime overhead) daha düşüktür.

Öncelik-temelli zamanlayıcı, bir çevrim içi zamanlayıcıdır. Görevlerin/işlerin planını önceden hesaplamaz; işlere öncelikleri atamak yerine, yayınlandığında onları öncelik sırasına göre bir çalışma kuyruğuna yerleştirir. Her bir zamanlama kararı süresinde, zamanlayıcı, çalışma sırasını günceller ve sıranın başındaki işi çalıştırır. Bir görevdeki işler, aynı (statik) ya da farklı (dinamik) önceliklere sahip olabilir.

2. Zaman (Tablo)-temelli (zaman-tetiklemeli) zamanlama (TT-CAN ve FTT-CAN)
   Önceden zaman planlamasının bilindiği katı gerçek-zamanlı sistemlerde kullanılır. Çevrim dışı (offline) zamanlama teknikleri burada kullanılabilir. Hangi işin çalışacağı ile ilgili kararlar, belli zamanlarda anlık olarak verilir. Bu anlar, sistem çalışmaya başlamadan önce seçilir.

   Genellikle, zamanlayıcı tarafından, düzenli aralıklarda periyodik kesme uygulanır. Zamanlayıcı, her kesmeden sonra harekete geçerek, sonraki periyot için çalıştırılacak işi planlar ve sonra, bir sonraki kesme için kendini bloklar.

   Tipik olarak zaman-temelli sistemlerde;
   - Gerçek-zamanlı için tüm parametreleri sabittir ve bilinir.
   - İşlerin zamanlaması çevrim dışı hesaplanır ve çalışma anında kullanılmak üzere saklanır.
   - Basit ve açıktır.
   - Esnek değildir.

3. Hibrit (mixed) zamanlama (Hiyerarşik zamanlama)
   Hibrit zamanlama algoritmalarında, hem öncelik hem de zaman temelli algoritmalar beraber kullanılır. Livani, Kaiser ve Jia (1999) tarafından önerilmiş bir algoritma olan hibrit zamanlama algoritması, TDMA ve dinamik LLF (Least Laxity First) zamanlamasının avantajlarını bir araya

getirme amacıyla sunulmuştur. Bununla birlikte, bu metodun dezavantajı, genişletilmiş CAN çerçevesi kullanılmasıdır. Genişletilmiş çerçeve iletimi daha çok bus bandgenişliği ziyan ederek, çerçevenin etkili veri hızını düşürür. Bandgenişliğinin daha etkili kullanımı için, zamanlama algoritmalarında standart çerçeve formatı tercih edilir.

Gerçek-zamanlı bir sistemde, iki farklı mesaj tipi vardır: Zaman mesajları ve olay mesajları. Durum mesajları genelde periyodik olarak gönderilir. Data bölgesinde durum bilgisi vardır. Olay mesajları genellikle bir olay olur olmaz hemen gönderilir. Data bölgesinde olay bilgisi vardır. Farklı karakteristiklerine rağmen, birçok gerçek-zamanlı sistem, olay ve zaman tetiklemeli trafiğe beraber ihtiyaç duyar. CAN'daki CSMA, dinamik zamanlanmış ve olay-tetiklemelidir (Wanke vd., 2009).

4.1. Kesintili ve Kesintisiz Zamanlama Algoritmaları

Zamanlama algoritmaları, saat kesmelerini nasıl kullandıklarına göre kesintili (Preemptive) ve kesintisiz (Non-Preemptive) olmak üzere ikiye ayrılır (Rouhifar ve Ravanmehr, 2015). Kesintili algoritmalarda, düşük öncelikli bir mesajın iletimi esnasında yüksek öncelikli bir mesaj gelirse, düşük öncelikli mesajın iletimi askıya alınır. Yüksek öncelikli mesajın iletimi bittikten sonra, düşük öncelikli mesaj iletime kaldığı yerden devam eder. Kesintisiz (sonsuz önceliksiz) algoritmalarda ise yüksek öncelikli bir mesaj gelse bile iletim devam eder. İşlem kendi isteği ile sonlanana kadar kesintisiz çalışır.

Kesintili zamanlama politikasını uygulamak, kesintisize göre daha zordur. Çünkü durdurulan bir mesajı yeniden başlatmak için fazladan kapasite gerekir. Bir ağ paketi için tek yol, tüm içeriği yeniden göndermektir. Bu da, ağın bandgenişliğini israf etmesi anlamına gelir. Bu sebeple, genelde kesintisiz zamanlama tekniği kullanılır (Yong, 2003).

4.2. Çevrim İçi ve Çevrim Dışı Zamanlama

Zamanlama (scheduling), Katı Gerçek-Zamanlı sistemlerin temel problemidir. İletim önceliği, ağ modülünden çıkmakta olan trafiğin hangi sırada iletileceğini belirler. İletim önceliği olan mesaj, iletim önceliği olmayandan daha önce iletilir. Bu sebeple, mesajlar önceliklendirilir. Önceliklendirmenin sabit ya da değişken olmasına bağlı olarak, çevrim içi (online) ve çevrim dışı (offline) olmak üzere iki ana zamanlama algoritması vardır. Çevrim içi zamanlamada, uygulamaların farklı zamanlamaları olabilir ve mesajların öncelikleri sistemin işleyişi boyunca değişebilir. Çevrim dışı zamanlamada ise, uygulamaların sabit zamanlamaları vardır ve mesajlar sabit önceliklere sahiptirler. Görev kümesinin zamanlanabilirliği, yürütmeden önce garanti edilir. Ancak, görevlerin zamanlama karakteristikleri ile ilgili önbilgi gerektirir. Eğer görev karakteristikleri, çalışma zamanının öncesinde bilinmiyorsa, zamanlama analizi çevrim içi yapılmalıdır (Michta, 2005).

İki temel tip çevrim içi zamanlayıcı vardır:

1. Planlama bazlı zamanlayıcı: Yeni bir görev geldiğinde, zamanlayıcı, hem yeni görevin hem de önceki planlanan görevin gereksinimlerine uyumlu olabilen yeni bir plan tanımlamaya çalışır.

2. Best-effort zamanlayıcı: Yeni bir görev geldiğinde, zamanlayıcı yeni bir plan yapmaya çalışmaz. Yeni görev, yürütme için kabul edilir ve sistem en iyi son teslim süresini yerine getirmeye çalışır.

4.3. Katı Gerçek-Zamanlı Mesaj Zamanlaması

Katı gerçek-zamanlı mesajlar, gerçek-zaman kısıtlamalarını karşılamak için düzgün bir şekilde zamanlanmalıdır. Çoklu-erişimli bir endüstriyel ağda mesajları zamanlamak, ortam erişim kontrol (MAC) protokolünün bir fonksiyonudur. Bu protokol, ağa erişim için karar verir ve verilen herhangi bir zamanda ne mesajının iletileceğini belirler (Ouni ve Kamoun, 2004).

Gerçek-zamanlı haberleşmede, iletilen paketler kesilemez, aksi halde hepsinin yeniden gönderilmesi gerekir. Bu durumda mesajlar iletim don teslim sürelerini kaçırabilirler.

Katı gerçek-zamanlı mesaj zamanlamasında;

- Her paketin katı gerçek-zaman sınırlamalarını karşılaması gerekir.
- İletilen paket kesilmemelidir.
- Periyodik ve aperiyodik paketler entegre edilebilmelidir.
- Uygulama sistemi, trafik değişim için uygun olmalıdır.
- Bus hattı daha yüksek kullanılabilmelidir.
- Maliyet düşük olmalıdır.
- Kullanımı kolay olmalıdır.

Bus'a erişim, bus zamanlama tablosuyla kontrol edilir. Bus tablosundaki her zamanlama talimatı, her bir paketin çalışma bilgisini gösterir. Bu bilgi, kaynak ve hedef adresi, başlangıç zamanı ile, paket çalışmasının maksimal gecikmesini ve diğer kontrol bilgisini içerir (Hui, 2008).

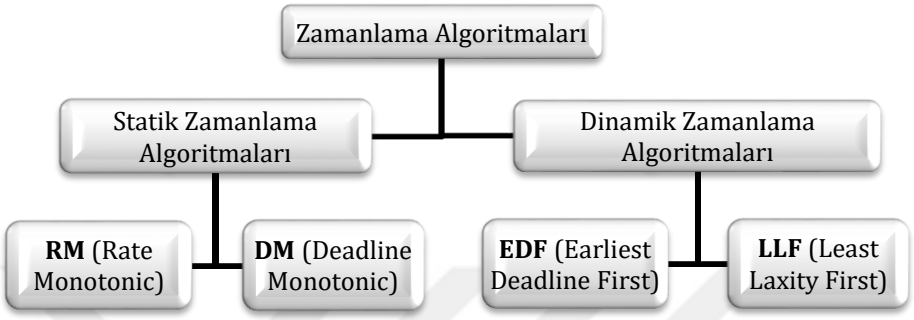
Katı gerçek-zamanlı mesajların, aşırı yüklenme durumlarında bile zamanında iletilmesini garanti altına almak için, tüm olayların öngörülebilir olması ve ilgili iletim sürelerinin önceden planlanmış olması gerekir. Katı gerçek-zamanlı mesajların son derece kritik doğası nedeniyle, planlanan iletim süreleri, en kötü durum hata gecikmelerini ve ayrıca, beklenen arıza koşulları altında yeniden iletim sürelerini içermelidir (Livani, vd., 1998).

4.4. Statik / Dinamik Zamanlama

Haberleşme sistemlerinde yaygın olarak öncelik-temelli zamanlama benimsenmiştir. Öncelik-temelli zamanlama algoritması, statik ya da dinamik olabilir. Eğer mesaj öncelikleri değiştirilebiliyorsa dinamik, değiştirilemiyorsa statik (sabit) zamanlama olarak adlandırılır.

Dinamik zamanlamada, hangi mesajın öncelikli olarak iletileceği kararı, çalışma esnasında verilir (Chen ve Yen, 2012). Statik zamanlamada ise, önceliklendirme kararları sistem çalışmaya başlamadan önce verilmiş olmalıdır.

Zamanlama algoritmalarının sınıflandırılması Şekil 4.1'de gösterildiği gibidir.



Şekil 4.1. Zamanlama algoritmalarının sınıflandırılması

4.4.1. Statik (Sabit) öncelikli zamanlama algoritmaları

Statik öncelikli zamanlama algoritmalarında, mesajların öncelikleri, yayınlanmadan önce sadece bir kez atanır ve sistem çalıştığı sürece değiştirilemez.

Sabit zamanlama algoritmalarının avantajları, kanal bandgenişliğinden verimli faydalanması, ek yüklerin (overhead) küçük olması, esnek bir yapıya sahip olması ve uygulamasının kolay olmasıdır (Dobrin ve Fohler, 2001). Dezavantajı ise nispeten yavaş olması sebebiyle, sistemin aşırı yüklenmesidir. Bununla birlikte, dinamik öncelikli zamanlamayla kıyaslandığında, yüksek bandgenişliği kullanamaz (Natale, 2000).

Sabit önceliklendirme temelli algoritma, Tek Düze Hız (Rate Monotonic – RM) ve Tek Düze Zaman Sınırı (Deadline Monotonic – DM) olmak üzere ikiye ayrılır (Shinde ve Biday, 2017).

4.4.1.1. RM zamanlama algoritması

Tek Düze Hız (Rate Monotonic - RM) zamanlama algoritması ilk olarak Liu ve Layland (1973) tarafından çalışılmıştır. Yazarlar, bu ünlü algoritmanın analizini bir katı gerçek-zaman ortamında gerçekleştirmiştir. RM'nin tüm sabit öncelikli şemalar arasında optimal olduğunu, yani bir görev kümesinin RM tarafından zamanlanamazsa, başka herhangi bir sabit öncelikli görev tarafından zamanlanamayacağını kanıtlamıştır (Bini ve Buttazzo, 2004). Çeşitli kısıtlamaları olmasına rağmen, uygulaması kolay olduğu için popüler bir algoritmadır. Sabit ve bilinen öncelik düzeni olan sabit görevler kümesi için, tüm görevlerin son teslim sürelerini karşılaması için sistem kullanımını garanti eder (Michta, 2005).

Mesajlar periyodik olmalı ve sonraki mesaj gönderilmeden önce iletimini bitirmeli, yani hedefe ulaşmalıdır. Bir başka deyişle, mesajın son teslim süresi (deadline), periyoduna eşit olmalıdır (D=T). Tüm mesajlar, periyotlarına göre zamanlanmalıdır (Harkut vd., 2014). Pratikte, katı gerçek-zamanlı ortamlarda her zaman bu durum gerçekleşmeyebilir. Mesajın durumu, önceliğine göre ayarlanır. Kısa periyoda sahip olan mesaja en yüksek öncelik verilir. Hız, periyodun tersidir. Bu nedenle, yüksek hızlı işlerin, yüksek önceliği vardır.

RM'nin dezavantajları:

1. Gelen görev kümeleri için periyot değerlerinde oluşabilecek değişimler, görev kümelerinin önceliklendirmelerinin sabit olarak belirlenmiş olması sebebiyle önceliklendirme hesaplamalarında değerlendirmeye alınmamaktadır.
2. Zamanlanabilecek maksimum görev için organize edilebilirlik yüzdesinin %100'den küçük olması başka bir dezavantajdır. Bu nedenle, sistem kaynakları üzerinde anlık aşırı yüklenmeler olması durumunda, önemli sayıda kritik görevin reddedilmesi ve sistem kaynaklarını kullanmak üzere bu görevlere kaynak tahsisi yapılmaması söz konusu olacaktır. Bu da, sistemde sadece yüksek frekanslı ve kısa zaman dilimi içerisinde gerçekleştirilebilecek görevler haricinde görev çalıştırılamaması

anlamına gelmektedir. Düşük frekanslı ama kritik seviyedeki görevlerin gerçekleştirilememesi, gerçek-zamanlı bir sistem için istenen bir durum değildir.

4.4.1.2. DM zamanlama algoritması

DM (Deadline Monotonic) öncelik ataması, kavramda RM öncelik atamasına benzerdir. Mesajlara atanan öncelikler, ilgili bağıl son teslim süresi ile ters orantılıdır (Leung ve Whitehead, 1982). Dolayısıyla, daha kısa son teslim süresi olan mesaj, en yüksek önceliği alır (Audsley vd., 1991; Tindell vd. 1994; 1995). Bu öncelikler, her bir mesaj için ID'yi biçimlendirir. Bir mesaj iletime başladığında, bu esnada daha yüksek öncelikli mesajlar yayınlansa bile, iletim tamamlanıncaya kadar çalışmaya devam eder (Zuberi ve Shin, 1995). Görevlerin bağıl son teslim süreleri, periyotlarına eşit değildir ($D \leq T$). Eğer eşit olursa, RM ve DM aynı sonucu verir.

4.4.2. Dinamik öncelikli zamanlama algoritmaları

Dinamik zamanlama aynı zamanda, çevrim içi zamanlama olarak da anılır. Sistemdeki kararların tümü ya da bir kısmı, uygulama esnasında verilir (Chen ve Yen, 2012).

Dinamik öncelikli zamanlama algoritmaları, bandgenişliği kullanımı iyileştirmek ve ağ performansını optimize etmek amacıyla geliştirilmiştir. Dinamik zamanlama planları, her bir mesajın QoS ihtiyacına göre öncelikleri tahsis eder. Daha yüksek zamanlanabilirliği koruyabilmek ve daha iyi QoS için, daha birçok öncelik seviyesine ihtiyaç vardır. Bununla birlikte, dağıtık zamanlama modlarında, mesajın önceliği ve tanımlaması ID bölgesinde ayrı olarak kodlanmıştır. Önceliği kodlamak için kullanılan etkin bitlerin sayısı sınırlı olduğundan, sadece sınırlı sayıda öncelik seviyesi mevcuttur. Sonuç olarak öncelik tersleme kaçınılmazdır ki bu da ağ performansını düşürür. Üstelik ağın izin verilebilir hizmet aralığı, sadece ID bölgesinde kalan etkin bitlerin mevcut olması nedeniyle sınırlıdır. Diğer taraftan, merkezileştirilmiş modlarda, bus

hakemliği, her bir döngüdeki tetikleme mesajı tarafından kontrol edilir ve ağdaki bir master düğüm tarafından tahsis edilir (Tao vd., 2005).

Statik öncelikli zamanlama algoritmalarının tersine dinamik zamanlama algoritmaları, mesajlara her seferinde farklı öncelikleri atar. Dinamik önceliklendirme temelli zamanlama algoritmaları Earliest Deadline First (EDF) ve Least Laxity First (LLF) olmak üzere iki başlıkta incelenir.

4.4.2.1. EDF zamanlama algoritması

En erken son tarih (EDF – Earliest Deadline First) zamanlama algoritması, Liu ve Layland (1973) tarafından geliştirilmiş dinamik bir algoritmadır. EDF, birçok avantajı sebebiyle haberleşme sistemlerinde oldukça sık tercih edilir (Fuster vd., 2005). EDF zamanlama algoritmasında, görev öncelikleri sabit değildir (Hui, 2008). RM algoritmasında olduğu gibi mesajın son teslim süresi, periyoduna eşit olmalıdır (D=T). EDF'de ağa eş zamanlı erişim talebi geldiğinde, görevler içerisinde son teslim süresi daha erken olan görev önceliği alır ve önce gerçekleştirilir. Bir başka deyişle, gönderilecek iki mesaj arasında, bir sonraki periyodu daha yakın olan mesaj gönderilir. Mesajın gönderimi, belirlenen son teslim süresinden önce bitmelidir. Eğer sonsuz önceliğe (preemption) izin verilmişse ve mesajlar bağımsızsa, EDF optimaldir denebilir.

EDF zamanlama algoritmasının avantajları, ağda meydana gelebilecek farklı durumlara göre, önceliklendirmelerin dinamik olarak belirlenebilmesi, mevcut kanal bandgenişliğinin kullanımında verimli olması ve daha yüksek bus kullanımını sağlamasıdır. Dezavantajı ise, ağda oluşabilecek anlık aşırı yüklenme durumunda, hangi görevlerin reddedileceğinin önceden belirlenebilme imkanının olmamasıdır (Dobrin, 2005). EDF'nin başka bir dezavantajı ise, uygulamasının RM'ye göre daha kompleks olmasıdır. Dinamik öncelik, daha büyük çalışma zamanı işlem yüküne maruz bırakır ve daha fazla bellek gerektirir. EDF tekniği, her bir zamanlama döngüsünde, önceliklerin güncellenmesini gerektirir (Natale, 2000; Shokry, vd., 2009).

EDF, DM ve RM gibi sabit öncelikli algoritmalardan, daha yüksek ağ kullanımını garanti eder (Natale, 2000). Ayrıca, RM yerine EDF kullanmak, jitterin azalmasını sağlar (Pedreiras ve Almeida, 2002).

4.4.2.2. LLF zamanlama algoritması

LLF (Least Laxity First), aynı zamanda MLF (Minimum Laxity First) olarak da adlandırılan bir dinamik zamanlama algoritmasıdır. Mesaj önceliklerini, kendi zaman paylarını (slack time) ya da ertelenebilirlik (laxity) değerini baz alarak atar. Zaman payı, geçerli zamandan arta kalan boş zamandır (Hwang vd., 2011). Daha küçük zaman payı ya da ertelenebilirlik değeri olan mesaj, en yüksek önceliği alır (Rouhifar ve Ravanmehr, 2015). Bir sürecin ertelenebilirlik değeri, bir mesajın son teslim süresinden, görev için gereken çalışma süresi çıkarıldığında kalan süre olarak tanımlanır (Shinde ve Biday, 2017).

Çalışma zamanının ve son teslim sürelerinin bilinmesi gerektiğinden, daha karmaşık bir yapısı vardır. Çalışma zamanı veriye bağlı olduğundan, gerçek çalışma zamanının ön bilgisini bilmek, çoğu zaman zordur. En kötü-durum tahminlerini kullanmak gerekir.

4.5. CAN Ağında Çevrim İçi Mesaj Zamanlaması

CAN, yüksek performanslı ve yüksek güvenirliği olan gerçek zamanlı veri haberleşmesi için uygun bir protokoldür. CAN protokolü, öncelik verilmiş çözümleme mekanizmasını desteklemek üzere tasarlanmıştır.

CAN'da optimum mesaj iletimi için etken faktör, mesaj önceliğinin belirlenmesidir. Önemli mesajlardaki aşırı gecikmeleri önlemek için, mesajları önceliklerine göre ayırmak gerekir. Aynı zamanda, çarpışma durumunda önemli mesajların kaybolmasını önlemek için de mesajlara öncelik sağlamak gerekmektedir (Oliveira vd., 2005). CAN mesajlar, CAN veri yoluna önceliklerine bağlı olarak erişirler. CAN mesajları önceliklendirildiğinde, en yüksek önceliğe sahip olan mesaj çarpışma meydana geldiğinde kesintiye uğramadan iletilirken,

daha düşük öncelikli mesajın iletimi sonlandırılır ve ağ boşaldığında yeniden iletime geçilir (Wen vd., 2007).

Gerçek-zamanlı bir ortamdaki çevrim içi zamanlamanın sorunu, gerçek-zamanlı görevler için sistem kaynaklarını atayarak, dinamik olarak kararların sıralamasını yapmaktır. Her bir zamanlama kararı, gelecekte yapılacak olan görev isteğinin ön bilgisi olmadan yapılmalıdır. Zamanlama kararları, mümkün olduğunca hızlı bir şekilde verilmelidir.

Aslında, CAN protokolü, mesaj iletimindeki dağıtık zamanlamasını kontrol etmek için etkili bir hakemlik mekanizması olan sabit-öncelik planını kullanır. Bununla birlikte, sabit-öncelik planı, ağın potansiyel kapasitesinin tümünü değerlendiremez. Çoğu gerçek-zamanlı uygulama için sabit-öncelikli plan uygun ve yeterli olsa da, ağır yüklenmiş ortamlarda ağın zamanlanabilirliğini ispatlamak gerekir. Geçmişte, dinamik öncelik planlarının, yüksek maliyet ve algoritmaların karmaşıklığından kaynaklanan aşırı yük (overhead) gibi bazı problemlerinin olması, CAN denetleyicilerinin düşük hızda çalışmaları sonucunu doğurmaktaydı. Fakat son yıllarda mikroişlemcilerin hızlarının artmasıyla, aşırı yük problemi azalmıştır. Aynı zamanda maliyetler de önemli ölçüde düşmüştür. Böylece dinamik öncelik planları CAN ağlarında kullanılabilir hale gelmiştir (Yong, 2003).

CAN'de kullanılan öncelik-temelli mesaj zamanlamasının birçok avantajı vardır. Bunlar (Dobrin, 2005):

- Verimli bandgenişliği kullanımı
- Esneklik
- Yürütmenin basit olması
- Küçük işlem yükü

Çoklu-erişim ağları üzerinde mesaj iletimlerini zamanlamak, Data Link katmanının MAC alt katmanında gerçekleşir. MAC protokolü, iki işlemden sorumludur; erişim hakemliği ve iletim kontrolü (Malcolm ve Zhao, 1995). İletim kontrolü işlemi, sistemin her düğümünün iletim için yalnızca bir mesajı

yetkilendirmesini içerir. Genel olarak, nakledilen bilgi, kontrol uygulaması ile ilgilidir ve böylece mesaj uzunluğu yüz biti geçer. Erişim hakemliği, çoğu bus protokolünde temel işlemdir (Fonseca ve Almeida, 1999).
Ağ üzerinden gönderilebilecek senkron mesajlar, çoğunlukla sensörler, denetleyiciler, algılayıcılar gibi cihazlar için üretilen periyodik değişkenleri taşımak için kullanılır. Asenkron mesajlar daha ziyade, alarm sistemleri gibi geçici durumlarla ilgili mesajlardır. Her iki mesaj tipinde de, son teslim sürelerinden önce iletim için garanti gerektiren, gerçek-zaman karakteristikleri olmalıdır (Fonseca ve Almeida, 1999).

CAN ID bölgesi iki amaca hizmet eder: bus hakemliği ve veri yönlendirme. ID, mesajın önceliğini tanımlar. Her bir mesaj için tanımlanmış ID benzersiz olmalıdır (Tao vd., 2005). EDF zamanlama algoritmasında, her bus hakemlik turunun başında, CAN ID'nin güncellenmesi gerekir (Shoukry vd., 2011). ID bölgesi, aynı öncelik seviyesine sahip iki ya da daha fazla mesaj olduğu durumlarda bus hakemliği için kullanılır.

EDF dinamik algoritması, zamanlama kararlarını verirken, sistem gelişimi sırasında değişebilecek parametreleri baz alır (Pedreiras ve Almeida, 2002). CAN bus üzerinde mesaj zamanlamasında EDF kullanılması, ağ kullanımı açısından % 20 civarında daha verimlidir (Pedreiras ve Almeida, 2002; Zuberi ve Shin, 1997).

EDF'de öncelikler dinamik olduğundan, bu yaklaşım CAN tanımlayıcıyı (ID) iki alana bölerek uygulanır. Şekil 4.2'de görülen bu alanlardan biri önceliği kodlamak için, diğeri ise, mesajın kendisini tanımlamak içindir. Bu yaklaşım dezavantajları (Pedreiras ve Almeida, 2002):

- Mesaj tanımlama için mevcut bitlerin sayısının azalması,
- Dinamik öncelik bölgesini güncellemek için bazı işlemlerin periyodik olarak gerçekleştirilmesi ihtiyacı,
- Son teslim sürelerini ifade etmek için sınırlı sayıda bit kullanımı dolayısıyla öncelik terslemelerinin oluşması
- Zaman senkronizasyon mesajlarının değişim ihtiyacı

Öncelik	ID

Şekil 4.2. CAN Tanımlayıcı Alanları

29 bit ID'li genişletilmiş CAN formatının, öncelik seviyelerini temsil etmek ve düğümleri tanımlamak için yeterli bitleri vardır. Bununla birlikte, CAN'in standart 11 bitli formatı ile kıyaslandığında, daha uzun ID formatının kullanılması sebebiyle bandgenişliğinin % 20-30'u heba olur (Tao vd., 2005).

Her durumda gerçek zamanlı mesajların, gerçek zamanlı olmayan mesajlardan daha yüksek öncelik almasını sağlamak için, öncelik bölgesindeki ilk bit gerçek zamanlı mesajları, gerçek zamanlı olmayan mesajlardan ayırmak için kullanılır. ID bölgesinde kalan yedi bit ise, her bir mesajın benzersiz bir kodu olduğundan emin olmak için kullanılır (Tao vd., 2005).

5. YAPAY SİNİR AĞLARI

Çok karmaşık, uzun ya da çok sayıda düzensiz bilgi taşıyan verilerin çözümlenebilmesinde, insan algısının ya da var olan bilgisayar tekniklerinin sonuca ulaşmada başarılı olamayacakları benzer tüm işlemlerde, üstün yeteneklerinden dolayı Yapay Sinir Ağları (YSA) günümüz için vazgeçilmez olmaya başlamıştır. Eğitilmiş bir sinir ağı, analiz etmesi için kendisine verilen bilgi kategorisinde bir 'uzman' olarak düşünülebilir. Bu uzman, daha sonra yeni ve tanımlanmamış durumlar, yani spesifik girdiler için farklı çıkışlar sağlayabilir (Coşkun ve Yıldırım, 2005).

YSA, insan beyninin sinir hücrelerinden oluşmuş katmanlı ve paralel olan yapısının tüm fonksiyonlarıyla beraber bilgisayar ortamına aktarılmasıyla oluşturulan ve böylece insan beyninin modellenmeye çalışıldığı sistemlerdir. Beyin fonksiyonlarını taklit edebilmek amacıyla değişik YSA modelleri geliştirilmiştir. YSA'lar öğrenme, ilişkilendirme, sınıflandırma, genelleme, özellik belirleme, optimizasyon gibi değişik problemlerin çözümünde kullanılabilmektedir (Öztemel, 2003).

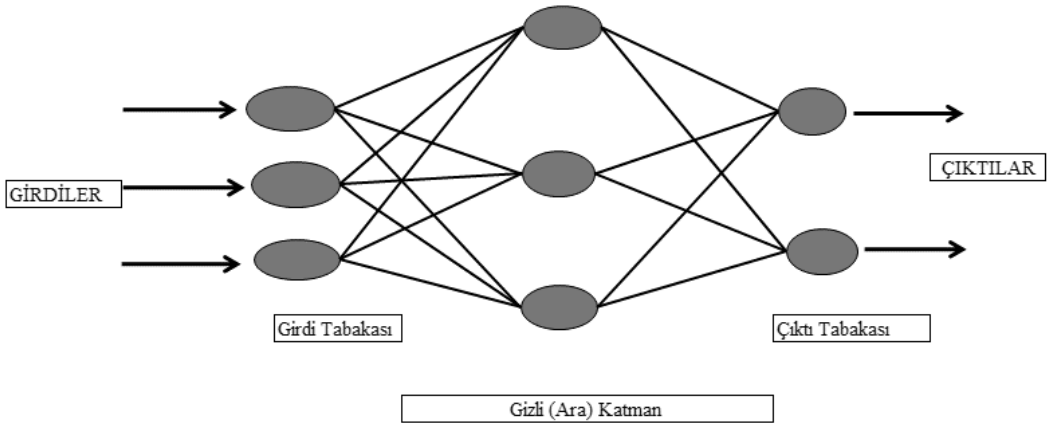
YSA'ların kilit unsurlarından biri, öğrenme yetenekleridir. Bir sinir ağı karmaşık bir uyarlama sistemidir, yani iç yapıyı girdilere ve hedeflere göre değiştirebilir. Bu YSA'ların belirli bir görev için eğitilmesi gerekir. YSA'yı eğitmek için üç tür öğrenme paradigması vardır ve bunlar aşağıdaki gibidir (Jyothi, 2015; Öztemel, 2003):

a) Öğretmenli öğrenme: Sistemin olayı öğrenebilmesi için bir öğretmen vardır. Ağı bir dizi örnek girdiyle sağlama ve çıktıyı beklenen yanıtla karşılaştıran bir süreçtir. Eğitim, ağ beklenen cevabı sağlayabilene kadar devam eder.

b) Öğretmensiz öğrenme: Bu eğitim yönteminde, girdi vektörü ve hedef çıktı bilinmemektedir. Ağ, en benzer giriş vektörünün aynı çıkış birimine tahsis edilecek şekilde değişebilir. Sistemin olayı öğrenebilmesi için yardımcı olacak bir öğretmen yoktur.

c) Destekleyici öğrenme: Ağın bir öğretmenin varlığında, ancak hedef vektörün yokluğundaki eğitim sürecidir. Öğretmen sadece doğru (1) ya da yanlış (0) olup olmadığını yanıtlar. Sistem, öğretmenden gelen bu sinyali dikkate alarak, öğrenme sürecini devam ettirir.

YSA modeli tipik olarak giriş, en az bir ara ve çıkış katmanından oluşmaktadır (Şekil 5.1). YSA hücresinde temel olarak dış ortamdan ya da diğer nöronlardan alınan veriler yani girişler, ağırlıklar, toplama fonksiyonu, aktivasyon fonksiyonu ve çıkışlar bulunmaktadır.



Şekil 5.1. Tipik YSA Modeli (Öztemel, 2006)

Bir YSA hücresine dış ortamdan ya da başka hücrelerden alınan bilgiler girilerek giriş oluşturulur. Ağırlıklar, bir yapay hücreye gelen bilginin önemini ve hücre üzerindeki etkisini gösterir. Toplama fonksiyonu, bir hücreye gelen net girişi hesaplar. Aktivasyon fonksiyonu, hücreye gelen net girişi işleyerek, hücrenin bu girişe karşılık üreteceği çıktıyı belirler. Üretilen çıktı, dış dünyaya ya da başka bir hücreye gönderilir.

Bir sinir ağı modeli oluşturulurken öncelikle kullanılacak ağ mimarisi seçilerek, ağın katman sayısı ve katmandaki nöron sayısı gibi yapısal özelliklerinin belirlenmesi gerekir. Ağdaki birimlerin kullandığı fonksiyonların karakteristik

özellikleri ve hangi öğrenme algoritmasının kullanılacağı belirlenmelidir. Sonraki aşama eğitim ve test setlerinin oluşturulmasıdır.

YSA öğrendikten sonra daha önce verilmeyen girişler verilip, sinir ağı çıkışıyla gerçek çıkışı yaklaşımı incelenir. Eğer yeni verilen örneklere de doğru yaklaşıyorsa sinir ağı doğru öğrenmiş demektir (Bulucu vd., 2007).

Yapay sinir ağlarındaki bilginin akış yönüne bağlı olarak ileri beslemeli ve geri beslemeli olmak üzere iki şekilde ağ modellemesi yapılabilir. İleri beslemeli ağlarda bilgi sadece ileri yönde hareket eder. Bu ağlarda döngüler yoktur. Katmanlar arasında tek yönlü bağlantılar bulunur. Geri beslemeli ağlarda ise bilgi sadece girdiden çıktıya doğru tek yönde geçekleşmez, aynı zamanda ters yönde de bilgi akışı olur. Bu ağlarda geri besleme bağlantıları nedeniyle döngüler görülür.

Bu tez çalışmasında geliştirilen tahmin modeli, zaman serisi modeli olduğundan, zaman serisi tahminleri için en uygun olan NARX ağının kullanılması uygun bulunmuştur.

5.1. NARX Modeli

Doğrusal olmayan dışsal girdili otoregresif ağ (NARX / Nonlinear Autoregressive Exogenous Model), istenen çıktının geçmişte zamanlardaki verilere bağlı olduğu problemlerde kullanılmak üzere Lin vd. (1996) tarafından önerilmiş bir yapay sinir ağı modelidir. NARX, birçok katman içeren geri beslemeli ve ileriye doğru hesaplamalı bir dinamik yapay sinir ağıdır. Model isminde bulunan “X”, eksojen değerleri yani, diğer dışsal değişkenlerin de modelin içerisine dahiliyetini ifade etmektedir. Dinamik yapay sinir ağlarında gelecekteki değerleri tahmin etmek için bir ya da daha fazla zaman serisinin geçmiş değerleri kullanılmaktadır. Ağ çıkışı, girişe geri besleme olarak uygulanır. Diğer geri beslemeli ağların aksine, NARX ağları bütün gizli katmanlardan değil sadece çıktı katmanındaki nöronlardan geribildirim almaktadır.

NARX modeli, zaman serisi modellemede yaygın olarak kullanılan doğrusal ARX (Autoregressive Exogenous) modeline dayanmaktadır. Bir NARX modelinde, bağımlı değişken geçmişteki değerinin bir fonksiyonudur. Birçok zaman serisi verisi de bu süreci içermektedir. NARX modeli, aşağıdaki gibi bir denklemle ifade edilebilir;

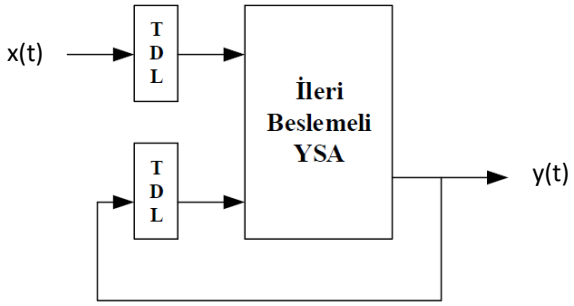
$$y(t) = f(y(t-1), y(t-2), \dots, y(t-n_y), x(t-1), x(t-2), \dots, x(t-n_x)) \quad (5.1)$$

5.1 denkleminde y(t-1), y(t-2),...,y(t-n_y) ağ çıktılarını ve x(t-1), x(t-2),...,x(t-n_x) ağ girdilerini ifade eder. n_y ve n_x ise sırasıyla geribesleme için uygulanacak geçmiş çıktıların ve geçmiş girdilerin sayısını gösterir. Çıktı sinyaline bağlı olan y(t) bir önceki çıktı değerinin sinyali ve bir önceki bağımsız girdi (dışsal) sinyalinin dönmesi ile hesaplanmaktadır (Mathworks, 2018).

5.2. NARX Mimarileri

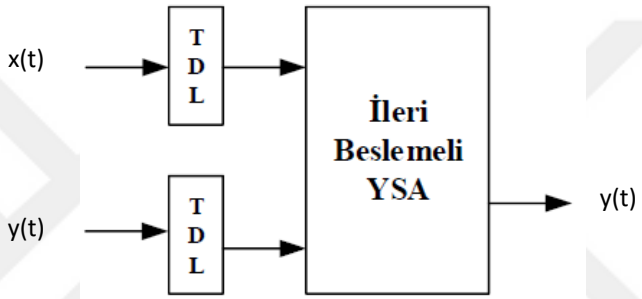
NARX ağları, paralel ya da seri-paralel mimarileri kullanılarak tasarlanabilir (Jyothi, 2015; Xie vd., 2009).

a) Paralel NARX mimarisi: Elde edilen tahmin çıkış değeri, Şekil 5.2'de görüldüğü gibi tekrar ileri beslemeli ağın girişine gönderilmektedir. Burada bir sonraki değerin tahmini, şebekeye yapılan girişlere ve önceki çıkışlara bağlıdır. TDL (Tapped Delay Line), zamansal gecikmeyi ifade eder ve ağı önceki girdi değerleriyle beslemeye yarar.



Şekil 5.2. Paralel NARX mimarisi

b) Seri-paralel NARX mimarisi: NARX ağının çıkışı, bazı doğrusal olmayan dinamik sistemin çıktısının bir tahmini olarak kabul edildiğinde kullanılır. (Şekil 5.3) Çıkış, standart NARX mimarisinin bir parçası olarak besleme iletme sinir ağının girişine geri beslenir. Ağın eğitimi sırasında gerçek çıktı mevcut olduğundan, tahmini çıktıyı geri beslemek yerine gerçek çıktının kullanıldığı bir seri-paralel mimari oluşturulur. Bunun iki avantajı vardır; ilki, ileri besleme ağına girişin daha doğru olmasıdır. İkincisi, sonuçta ortaya çıkan ağın tamamen ileriye doğru bir besleme mimarisine sahip olması ve eğitim için statik geri yayılımın kullanılabilmesidir.



Şekil 5.3. Seri-paralel NARX mimarisi

6. ARAŞTIRMA BULGULARI VE TARTIŞMA

6.1. SocketCAN

Socketcan, Volkswagen Research tarafından geliştirilmiş, Linux işletim sistemi altyapısını kullanan açık kaynak kodlu bir CAN sürücüsüdür (Sojka vd., 2010). C programlama dilinde yazılmıştır. Linux çekirdeği, tam özellikli bir CAN ağı alt sistemi içerir. Uygulamalara birkaç farklı arayüz üzerinden erişilebilir. CAN-utils paketi ile birlikte, CAN ağları ile çalışmak için çok yönlü ve kullanıcı dostudur (Sojka vd., 2014). Birden fazla uygulamanın aynı anda bir CAN aygıtına erişmesine izin verir. Ayrıca, tek bir uygulama birden fazla CAN ağına paralel olarak erişebilir. SocketCAN'ın temel bileşenleri, farklı CAN denetleyicileri için ağ aygıt sürücüleri ve CAN protokol ailesinin uygulanmasıdır.

SocketCan, sanal CAN (VCAN) cihazları oluşturarak ağ arayüzlerinin kullanılmasını sağlayan bir yazılımdır. Böylece CAN protokolleri herhangi bir donanıma gereksinim duymadan da çalıştırılabilmektedir. Sanal CAN cihazları genellikle vcan0, vcan1, vcan2… şeklinde adlandırılır. Sanal CAN sürücüsü bir modül olarak derlendiğinde, vcan.ko olarak adlandırılır (Github, 2010).

Socketcan, Berkeley soket uygulama programlama arayüzünü (API) kulanır. Programlamada API, yazılım uygulamaları inşa etmek için takip edilen rutinler, uygulanan protokoller ve kullanılan araçlar bütünüdür (Github, 2017).

CAN denetleyici donanımı için cihaz sürücüleri, kendilerini bir ağ cihazı gibi Linux ağ katmanı ile kaydederler. Böylece, denetleyiciden gelen CAN frameleri ağ katmanından ve CAN protokol modülü üzerinden geçebilir. Aynı zamanda, protokol aile modülü iletim protokolü modüllerini kaydetmek için bir API sağlar. Bu sayede herhangi bir sayıdaki iletim protokolleri dinamik olarak yüklenebilir ya da kaldırılabilir. Aslında, CAN çekirdek modülü tek başına herhangi bir protokol sağlamaz ve en az bir ek protokol modülü yüklemeden kullanılamaz. Aynı anda, farklı ya da aynı protokol modülünde birden fazla soket açılabilir. Bu soketler, farklı ya da aynı CAN ID'sine sahip çerçeveleri

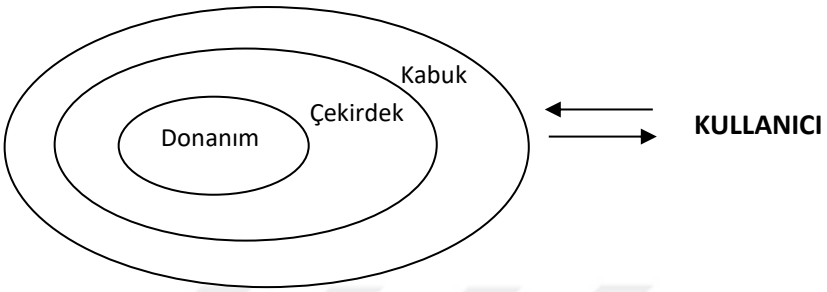
dinleyebilir ve gönderebilirler. Aynı CAN ID'si olan çerçeveler için aynı arayüzü dinleyen çeşitli soketlere, hepsi aynı alınan CAN çerçeve eşleşmeleri iletilir (Github, 2017).

Mesaj alma için kesme işleyicisi mesajı okuyarak, bu mesajı CPU softnet_data sırasına göre sıraya sokan netif_rx() öğesini çağırır ve sonraki işleme yönelik NET_RX_SOFTIRQ zamanlamaları yapar. Bu sıranın yalnızca CAN için değil, her tür ağ trafiği için paylaşıldığını bilmek önemlidir. Softirq, process_backlog() içindeki softnet_data kuyruğunu yoklayan ve orada bulunan her ileti için netif_receive_skb() öğesini çağıran net_rx_action() işlevini çalıştırır. Bu softirq'de ayrıca hız sınırlaması olduğuna dikkat edilmelidir. Bu nedenle diğer kaynaklardan gelen birçok paket varsa, CAN mesajları ek gecikmelerle ertelenebilmektedir. Socketcan can_rcv() durumunda iken, Netif_receive_skb() içinde kayıtlı paket türlerinin listesi geçilir ve mesaj işleyicisine geçirilir. Can_rcv() soket listesini geçerek, mesajı almak isteyen tüm soketlere gönderir. Raw (işlenmemiş) soketler için, soket arabelleğini klonlayan raw_rcv() öğesini çağırarak gerçekleştirilir ve verileri skb_queue_tail() tarafından soket alma kuyruğuna kopyalar ve kullanılan alanı sock_def_readable() içinde uyandıracak şekilde sock_queue_rcv_skb() çağırır.

6.2. Linux Üzerinde Kabuk Programlama

Linux sistemlerinde bütün programların donanım ile iletişimini çekirdek (kernel) kurar. Kabuk (shell) ise çekirdek üzerinde, kullanıcı ile aradaki bağlantıyı sağlayan programdır. Kabuk programlama, Linux altında çalışan hızlı ve pratik programlama yapmanın en kısa yoludur. Kabuk aracılığı ile kullanıcıdan alınan komutlar, çekirdek tarafından donanım üzerinde çalıştırılır ve sonuçları yine kabuk aracılığı kullanıcıya iletilir. Böylece kullanıcı, ne donanım ile ne de çekirdek ile iletişim kurmaz. Kabuk bir komut yorumlayıcıdır. İşletim sistemi çekirdeği ve kullanıcı arasındaki yalıtım katmanından daha fazlası, aynı zamanda oldukça güçlü bir programlama dilidir. Komut dosyası olarak adlandırılan bir kabuk programı, sistem çağrıları, araçlar, yardımcı programlar ve derlenmiş ikili dosyaları "birbirine yapıştırmak" yoluyla

uygulamalar oluşturmak için kullanılan bir araçtır. UNIX komutlarının, yardımcı programlarının ve araçlarının neredeyse tümü bir kabuk komut dosyasına çağırmak için kullanılabilir. Test ve döngü yapıları gibi iç kabuk komutları, komut dosyalarına ek güç ve esneklik kazandırır (Cooper, 2010). Donanım, çekirdek, kabuk ve kullanıcı arasındaki ilişki Şekil 6.1 'de gösterilmiştir.



Şekil 6.1. Donanım, çekirdek, kabuk ve kullanıcı arasındaki ilişki

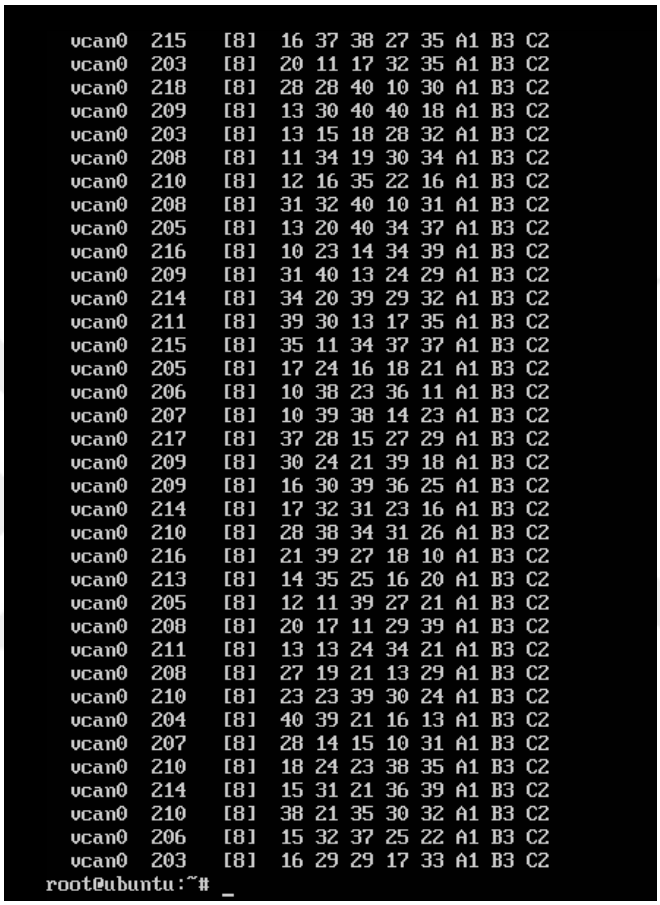
Linux sistemlerde defalarca yazmanın zor olacağı karmaşık işlemleri çok daha hızlı gerçekleştirmek için, programlama yapılarını kullanarak, komutlar bir dosyada bir araya getirilir. İşte bu sebeple bir dosyada toplanan ve çalıştırılan komutları içeren programlara kabuk scriptleri denir. En önemli ve en çok kullanılan kabukların başında tcsh, bash (Bourne Again Shell) ve ksh (Korn Shell) gelir.

6.3. SocketCAN ile Çevrimiçi (Dinamik) Önceliklendirme Uygulaması

Bu tez çalışmasında, SocketCAN üzerinde çalışan bir dizi komut içeren bash script (betik) ile kabuk programlamalar yapılarak, mesajların dinamik olarak önceliklendirilmesi sağlanmıştır. Sistem açıldıktan sonra, ilk olarak kabuk programı çalıştırılır. Bu andan sonra yapılan tüm işlemler bu kabuk programı tarafından yönetilmektedir.

İlk bash script, istenen sayı ve özellikte veri üretebilmek için tasarlanmıştır. Bu script ile herhangi bir zamanlama ayarlaması yapılmamış, mesaj sayıları ve CAN ID aralıkları belirlenmiştir. Mesaj ID'leri, 202 ile 218 arasında onaltılık (hex) değerler olarak seçilmiştir. Toplamda 17 tane farklı ID değeri olan 1000 adet

mesaj üretilmiştir. Üretilen mesajlardaki 8 byte uzunluğundaki datalar random atanmıştır. Şekil 6.2'de önceliklendirme yapılmamış 8 byte boyutundaki mesajların akış trafiğinin bir kısmı görülmektedir. Bu şekilde, birinci sütun CAN bus hattını, ikinci sütun mesajın ID değerini, üçüncü sütun verinin boyutunu ve dördüncü sütun ise iletilen mesajı belirtmektedir.



Şekil 6.2. Önceliklendirme yapılmamış 8 byte mesajların akışı

Oluşturulan diğer bir bash script ile her mesajın zaman aralıkları ve önceliklendirmelerin dinamik olarak gerçekleştirilmesi programlanmıştır. Mesajların son teslim süreleri 1 ile 50 ms arasında random olarak atanmıştır. Son teslim süresi düşük olan mesajın daha önce iletilmesi için, bu mesajın orijinal ID değerinden mesaj gönderecek düğüm sayısı çıkarılarak, yeni ID değeri hesaplanır. Eğer mesaj sayısı M ise;

Mesajın yeni ID değeri = Mesajın orijinal ID değeri – M (6.1)

şeklinde hesaplanmaktadır. Eğer sonraki mesajın öncelik değeri daha büyükse, CAN bus üzerinde mesaj orijinal ID'si ile iletilir.

Bu çalışmada 17 tane farklı mesaj gönderildiği için, son teslim süresi düşük olan mesaj hakemlik yarışmasını kazandığında öncelikli olarak iletilebilmesi için ID değerinden 17 düşülmektedir. Böylece hiçbir mesajın ID'si aynı olmaz. Bu sistem bir Elektronik Kontrol Birimi (ECU) gibi çalışarak, mesaj zamanlaması kararlarını vermektedir.

Tasarlanan CAN ağı modeli için kullanılan bu algoritma, iletilen her mesajın önceliğinin dinamik olarak düzenlemesine olanak sağlamaktadır. Geliştirilen algoritmaya göre çevrimiçi öncelikli mesaj zamanlaması akış diyagramı Şekil 6.3'de görülmektedir.

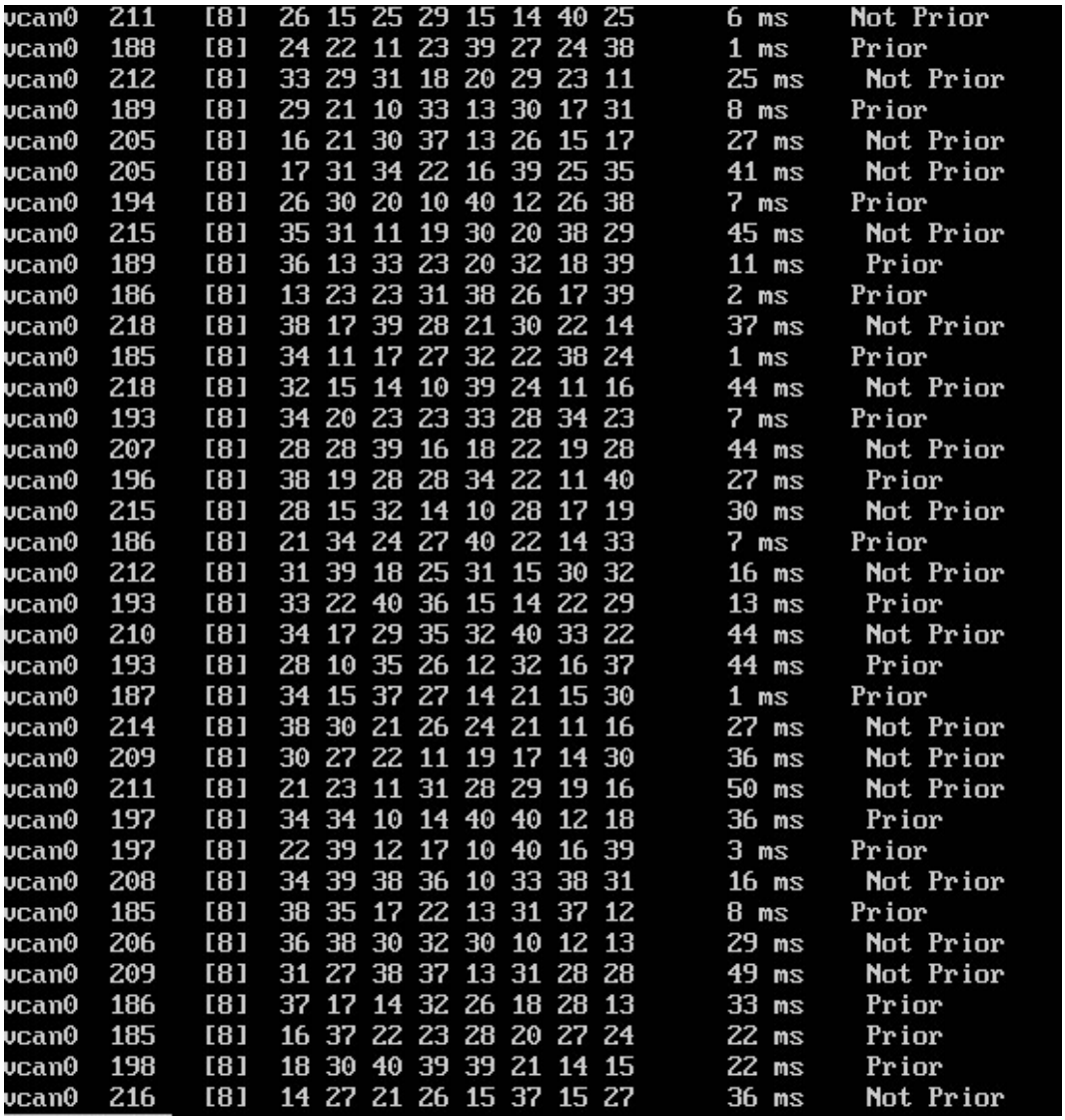
Şekil 6.3. Çevrimiçi öncelikli mesaj zamanlaması akış diyagramı

Mesajların orijinal ID'si ile önceliklendirme yapıldıktan sonra oluşan yeni ID'ler Çizelge 6.1'de verilmiştir.

Çizelge 6.1. Mesajların orijinal ve yeni ID'leri

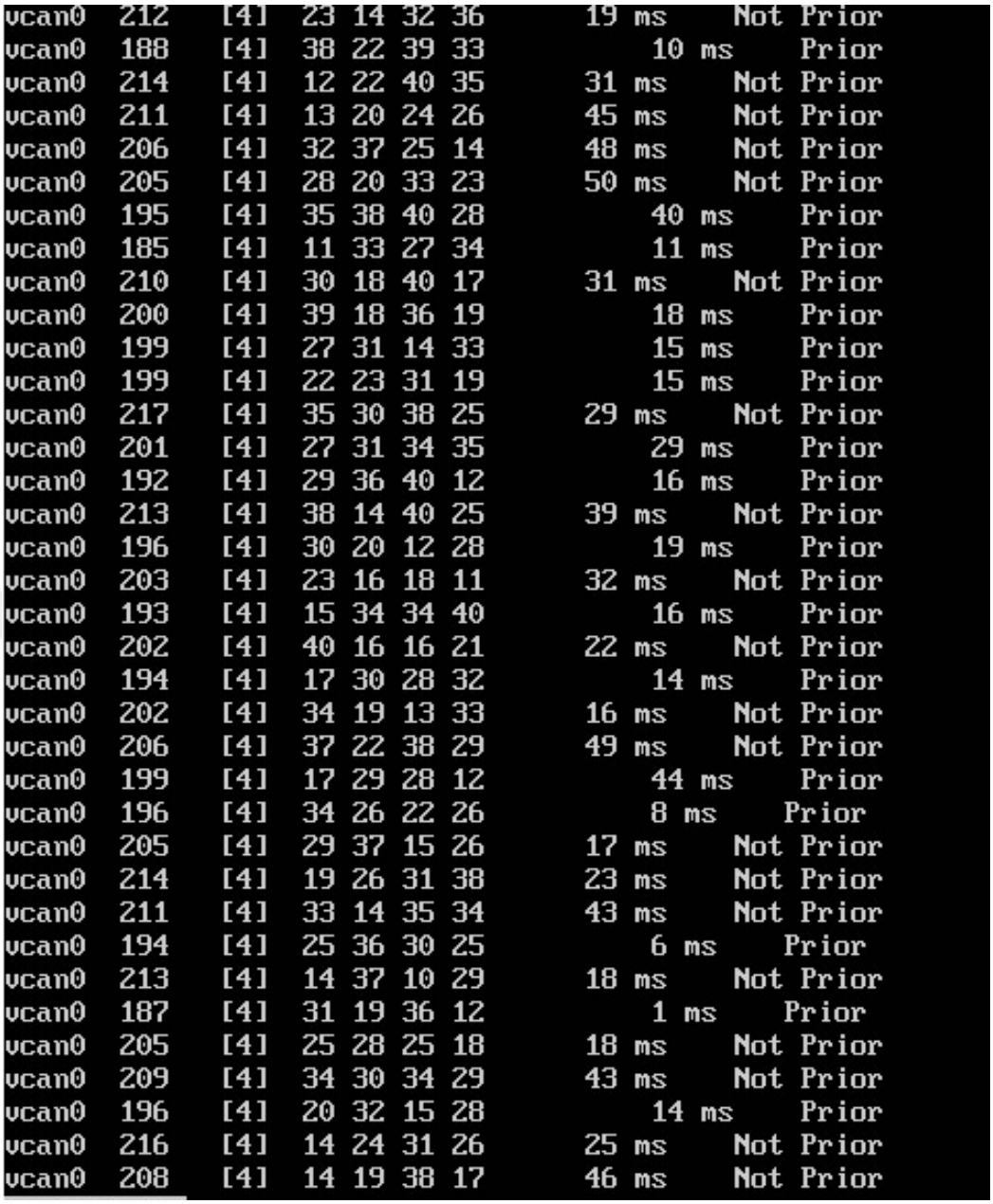
Mesajın orijinal ID'si	Önceliği alan mesajın yeni ID'si
202	185
203	186
204	187
205	188
206	189
207	190
208	191
209	192
210	193
211	194
212	195
213	196
214	197
215	198
216	199
217	200
218	201

Önceliklendirme işlemi yapıldıktan sonra oluşan akış trafiğinin bir kısmı Şekil 6.4'de görülmektedir. Son teslim süresi daha önce olan mesajın, ID değeri de düşük olacağından CAN ağlarının doğasında bulunan daha düşük ID değerli mesajın iletimi ilkesi bu şekilde sağlanmıştır.

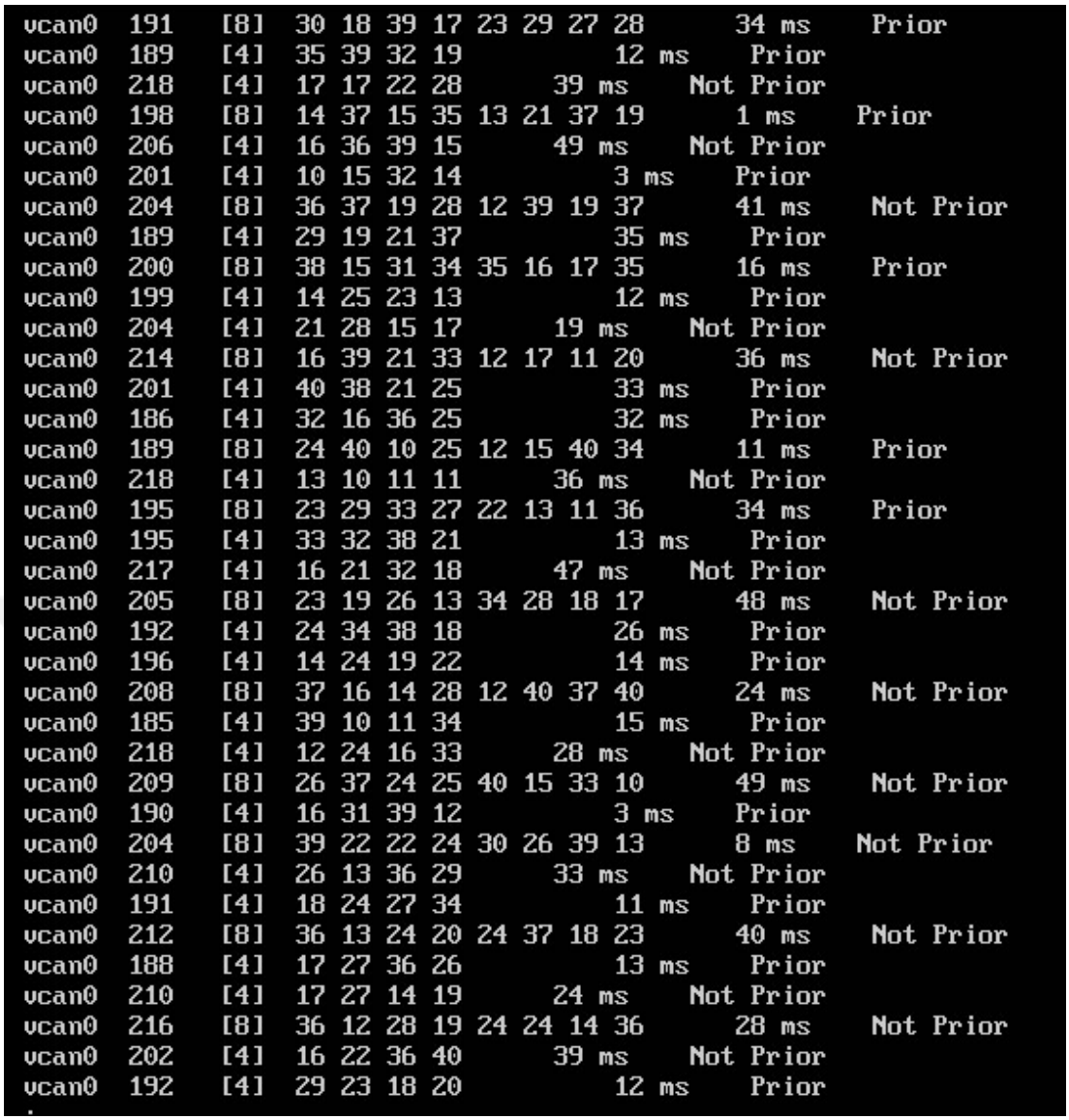


Şekil 6.4. Önceliklendirme yapılmış 8 byte mesajların akışı

Çalışmada daha sonra 4 byte uzunluğunda 1000 tane mesaj üretilerek, 8 byte uzunluğundaki mesajlar ile karşılaştırma yapılmıştır. 4 byte uzunluğundaki mesajlar için de 202 ile 218 arasında ID değerleri seçilerek, son teslim süreleri 1 ile 50 ms olarak ayarlanmıştır. Şekil 6.5'de 4 byte uzunluğundaki mesajların akışının bir kısmı görülürken, Şekil 6.6'da ise 4 ve 8 byte uzunluğundaki veriler aynı anda gönderilerek bir akış oluşturulmuştur.



Şekil 6.5. Önceliklendirme yapılmış 4 byte mesajların akışı



Şekil 6.6. Önceliklendirme yapılmış 4 byte ve 8 byte mesajların akışı

Üretilen CAN verilerinin okunarak, ekrana çıktı vermesi sağlanmıştır. Bu çıktı, bir metin dosyasına yazılmaktadır.

CAN ağı tasarlanırken, gerçek zamanlı mesaj trafiğini desteklemesi ve uygulaması temel alınmıştır. Tasarım, yüksek öncelikli mesajların son teslim sürelerini karşıladığını ve daha düşük öncelikli mesajların yüksek öncelikli mesajları engellemediğini kanıtlamıştır. Gelecekteki çalışmalarda, farklı senaryolar için yeni ağ modelleri tasarlanarak, dinamik öncelik temelli SocketCAN uygulaması pek çok endüstriyel ağda etkili olarak kullanılabilir.

6.4. Optimal YSA-NARX Mimarisinin Uygulanması

YSA-NARX modelinin optimal mimarisinin tanımlanması, eğitim ve doğrulama için veri üretilmesini ve minimum ortalama kare hatası (MSE) sağlayan mimarinin seçimini içerir. Asgari MSE'ye sahip mimari, gizli katmanın nöronlarını ve eğitim tekrarlarının sayısını değiştirerek seçilir (Manonmani vd., 2016). Ağı eğitmek için CAN ağında üretilen giriş ve çıkış verisi kullanılır. YSA-NARX modeli, hedef veri olarak CAN ağı çıktısının geçmiş değerleri ile sağlanır. Giriş katmanı, mesajların ID ve öncelik değerlerine karşılık gelen iki nöron içerir. Çıkış katmanındaki ise öncelik değerlerine karşılık gelmektedir. NARX mimarisi, aktivasyon fonksiyonu olarak bir sigmoidal fonksiyon kullanır.

CAN ağlarındaki optimum dinamik mesaj iletiminin yapay sinir ağları ile değerlendirilmesi için MatLab yazılımındaki ntstool (Neural Network Time Series Tool) aracı kullanılmıştır. Bu araç, ekonomik değişkenlerin gelecekteki değerlerini tahmin etmek, kimyasal süreçler, imalat sistemleri, robotik, havacılık araçları ve bu gibi dinamik sistemleri temsil etmek için geliştirilen modellerde de kullanılabilmektedir.

Uygulamada Levenberg-Marquardt eğitim algoritması kullanılmıştır. MatLab ntstool aracı ile kullanılabilecek eğitim algoritmaları şunlardır:

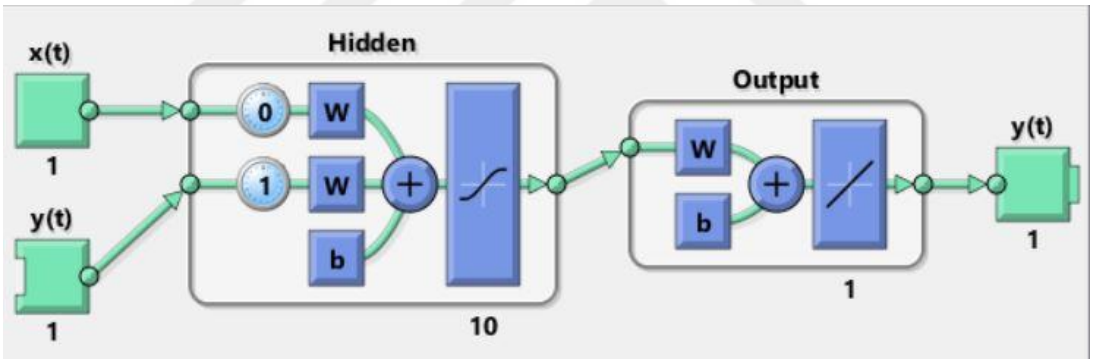
Levenberg-Marquardt: Eğitim, doğrulama örneklerinin ortalama karekökündeki bir hata ile gösterildiği gibi, genelleme durduğunda otomatik olarak durur. Bu algoritma, lineer olmayan fonksiyonların karelerinin toplamı olan bir fonksiyonun minimumunu bulmada Gauss-Newton yöntemine popüler bir alternatif olan yinelemeli bir yöntemdir. (Labde vd., 2017)

Bayesian Regularization: Bu algoritma genellikle daha fazla zaman alır, ancak zor, küçük ya da gürültülü veri kümeleri için iyi bir genelleştirmeyle sonuçlanabilir. Eğitim adaptif ağırlık minimizasyonuna göre durur.

Scaled Conjugate Gradient: Bu algoritma daha az bellek alır. Eğitim, doğrulama

örneklerinin ortalama karesel hatasında bir artışla gösterildiği gibi, genelleme durduğunda otomatik olarak durur.

Yapay sinir ağı uygulamaları için, Şekil 6.4'de mesaj akış trafiğinin bir kısmı görülen 8 byte uzunluğundaki veri seti kullanılmıştır. Yapay sinir ağı oluşturulmadan önce veri setinin bir bölümü ağın eğitimi amacıyla ayrılır. Bu uygulamada mevcut verilerin % 70'i NARX ağının eğitimi, % 15'i doğrulama ve % 15'i test işlemleri ve ağın başarısının ölçülmesi için ayrılmıştır. Şekil 6.7'de ntstool tarafından oluşturulan NARX ağı modeli görülmektedir. Bu modelde, x(t) mesajların ID değerlerini, y(t) ise mesajların önceliklerini temsil etmektedir. Oluşturulan modelde çıkışa zaman gecikmesi uygulanarak tekrar giriş olarak kullanılmaktadır. Girişe tekrar uygulanan çıkıştaki gecikme uzunluğu 1 alınmıştır. Buna göre ağ 2 giriş katmanı, 1 gizli katman ve 1 adet de çıkış katmanına sahiptir. Gizli katmanda 10, çıkış katmanında ise 1 adet nöron vardır.

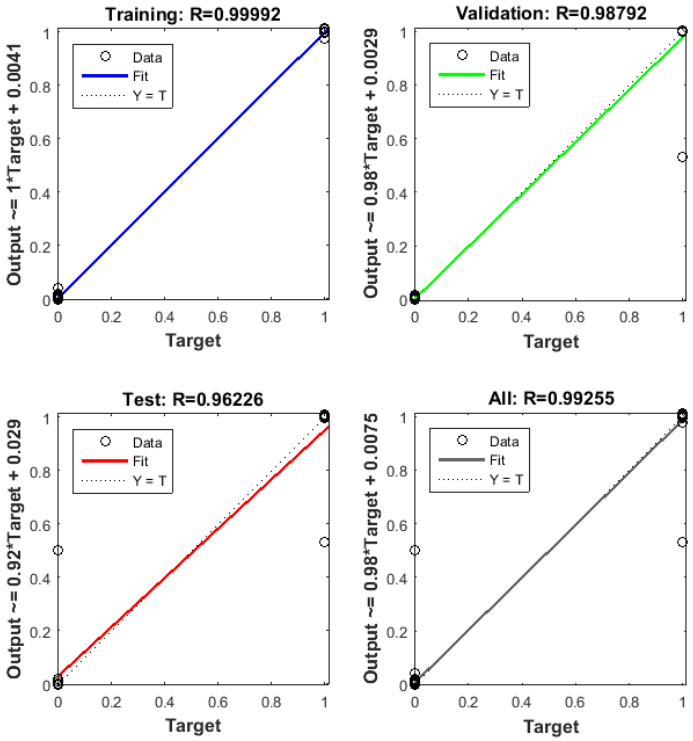


Şekil 6.7. Oluşturulan NARX modeli

Yapay sinir ağından optimum çıktı elde etmek için birkaç eğitim yinelemesi gerçekleştirilmiştir. İlk uygulamada, CAN ağından elde edilen mesajların oluşturduğu veri setinden 248'i yapay sinir ağ yapısında eğitim, test ve doğrulama amaçlı kullanılmıştır. YSA'nın başarısını gözlemlemek için ayrıca 249 mesaj da test amaçlı kullanılmıştır. Eğitim 15 iterasyon ile gerçekleştirilmiş.

Eğitim işleminin başarısı R (Regresyon) parametresi ile belirlenmekte ve hata oranı değerlendirilmektedir. İki ya da daha fazla değişken arasındaki ilişkileri

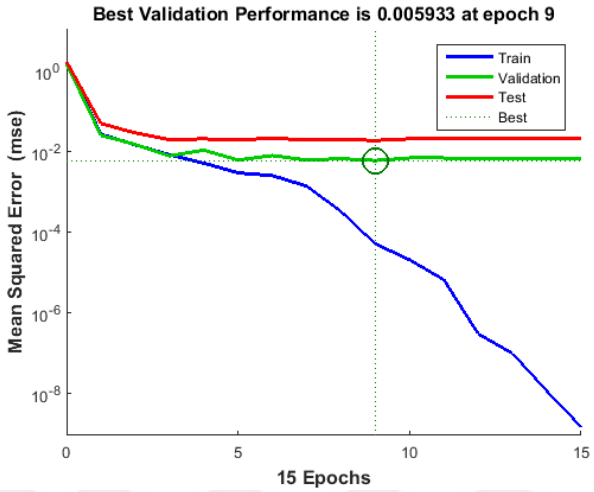
ölçmek için regresyon analizi kullanılmaktadır. Eğitim, doğrulama ve test aşamalarında elde edilen sonuçlara ilişkin regresyon analizi grafikleri Şekil 6.8'de verilmiştir. Şekilden hedeflenen değer olan 1'e oldukça yakın bir değer bulunduğu görülmektedir. Bu durumda, ağın çıktısıyla istenen çıktı arasındaki ilişki doğrudur.



Şekil 6.8. Birinci uygulamanın regresyon analizi grafiksel sonuçları

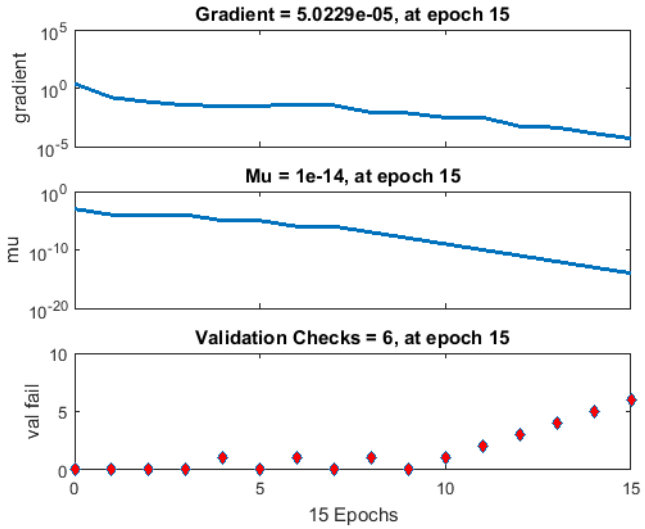
NARX ile oluşturulan modelin performans grafiği Şekil 6.9'da verilmiştir. Ağın performansı, eldeki veri seti için MSE (Ortalama Hata Kareleri) fonksiyonunu minimize eden ağırlık değerlerinin bulunmasına göre değerlendirilir. En küçük hata düzeyine ulaşılmaya çalışılır. Bu şekil, eğitim ve doğrulama hatalarının vurgulanan iterasyon sayısına (epoch) kadar düştüğünü göstermektedir. MSE, modelin test değerlerinin tahmini ile gerçek test değeri arasındaki mesafedir. MSE değeri, ağın çizdiği hiper-düzlemin ortalama olarak, doğrulama setindeki gerçek verilere kadar yakın olduğunu gösterir. MSE'nin sıfıra yakın olması, istenen çıktıların ve YSA'nın eğitim seti için çıktılarının birbirine çok yakın

olduğunu göstermektedir. Bu uygulamada en küçük MSE değeri, 9. iterasyonda elde edilmiştir. En iyi öğrenme bu aşamada gerçekleşmiştir. Her iterasyonda elde edilen MSE değerleri aşağıdaki şekilde gösterildiği gibidir.



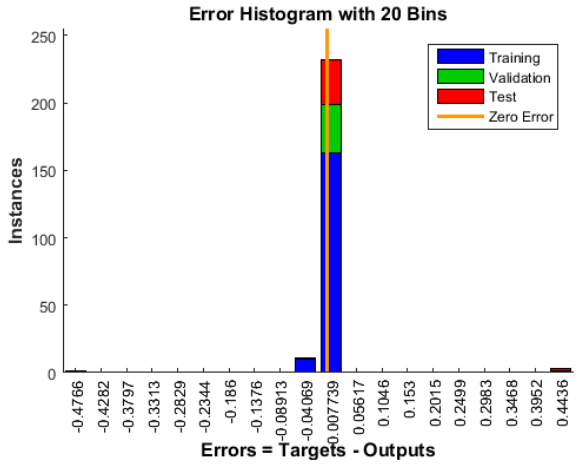
Şekil 6.9. Birinci uygulamanın NARX modelinin performans grafiği

Şekil 6.10'da NARX ağının 15 iterasyon boyunca eğitim durumu çıktıları görülmektedir. Şekil, eğitim sürecinin 15 dönemi boyunca gradyan, mu (sinir ağının ağırlık değişimleri) ve doğrulama kontrollerinin sayısını göstermektedir.



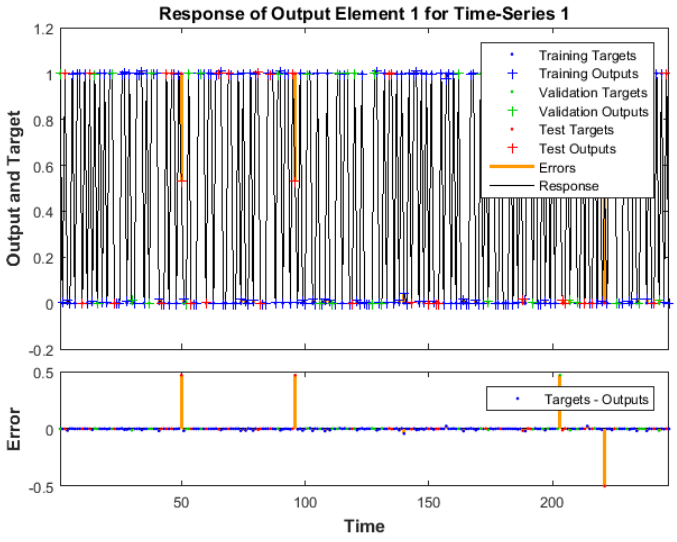
Şekil 6.10. Birinci uygulamanın eğitim durumu grafiği

Tahminler sonucu meydana gelen hata miktarları Şekil 6.11'de histogram grafiği olarak verilmiştir. Histogram grafiği, ileriye dönük bir sinir ağını eğittikten sonra hedef değerler ile tahmin edilen değerler arasındaki hataları göstermektedir. Buradaki "bin" ifadesi, grafik üzerindeki dikey çubukların sayısını verir.



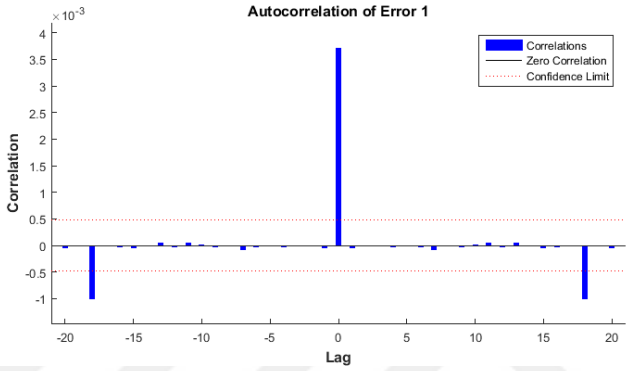
Şekil 6.11. Birinci uygulamanın hata histogram grafiği

Şekil 6.12'de zaman serisi eğitim, test ve doğrulama sonuç grafiği görülmektedir. Bu grafik, zamana karşı girdileri, hedefleri ve hataları görüntüler. Ayrıca eğitim, test ve doğrulama için hangi zaman noktalarının seçildiğini de gösterir.



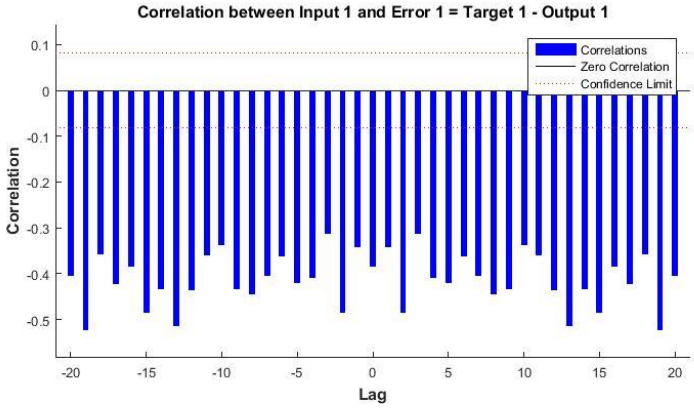
Şekil 6.12. Birinci uygulamanın zaman serisi eğitim, test ve doğrulama sonuç grafiği

Şekil 6.13'de otokorelasyon hatası grafiği görülmektedir. Bu grafik, ağ performansını doğrulamak için otomatik ilişkilendirme işlevini görüntüler. Mükemmel bir tahmin modeli için, sadece otokorelasyon fonksiyonunun sıfır olmayan bir değeri olmalı ve sıfır gecikme ile gerçekleşmelidir. Bu, tahmin hatalarının birbiriyle tamamen ilişkisiz olduğu anlamına gelecektir (beyaz gürültü). Tahmin hatalarında anlamlı bir korelasyon varsa, gecikme hatlarındaki gecikme sayısını artırarak, tahminin geliştirilmesi mümkün olabilir.



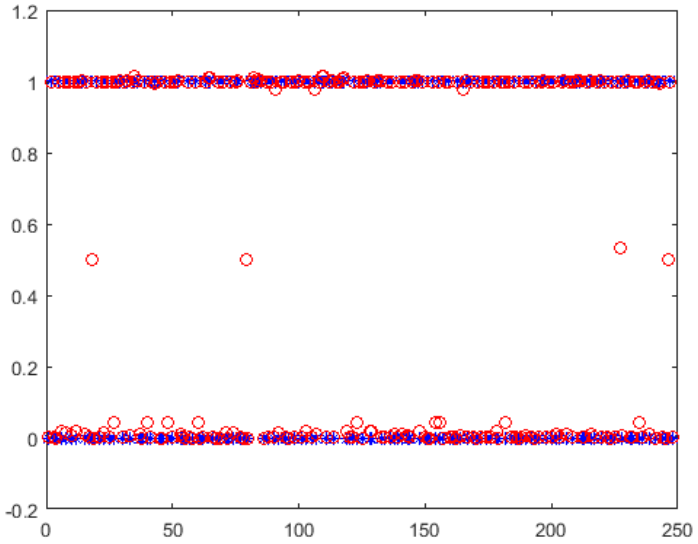
Şekil 6.13. Birinci uygulamanın otokorelasyon hatası grafiği

Şekil 6.14'de ağ performansının ek doğrulaması için giriş hatası çapraz korelasyon (açıklayıcılık) grafiği görülmektedir. Bu grafik, hataların x (t) giriş dizisi ile nasıl ilişkilendirildiğini gösterir. Mükemmel bir tahmin modeli için, tüm korelasyonlar sıfır olmalıdır.



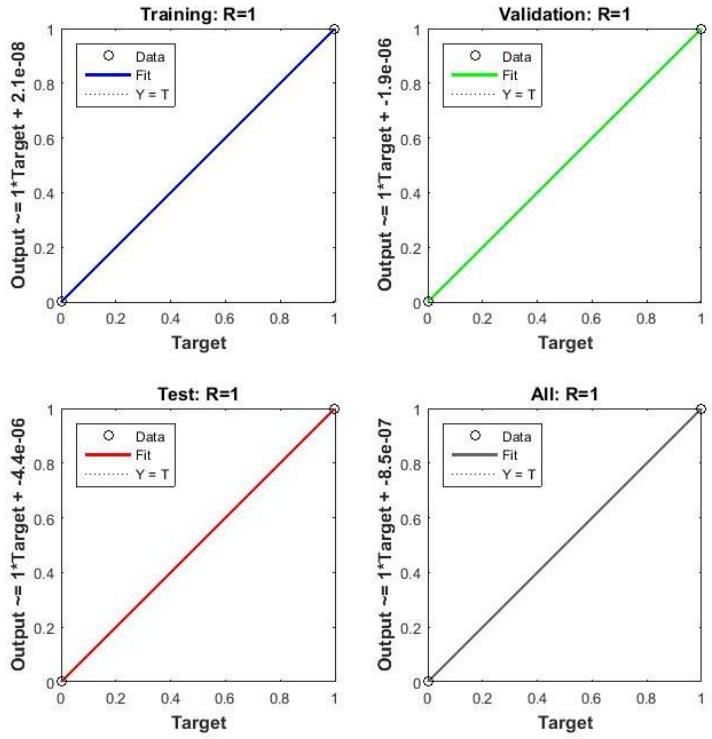
Şekil 6.14. Birinci uygulamanın korelasyon grafiği

Şekil 6.15'de oluşturulan NARX modelinin sonuç grafiği görülmektedir. Bu grafik, test için ayrılan 249 mesajı temel almıştır.

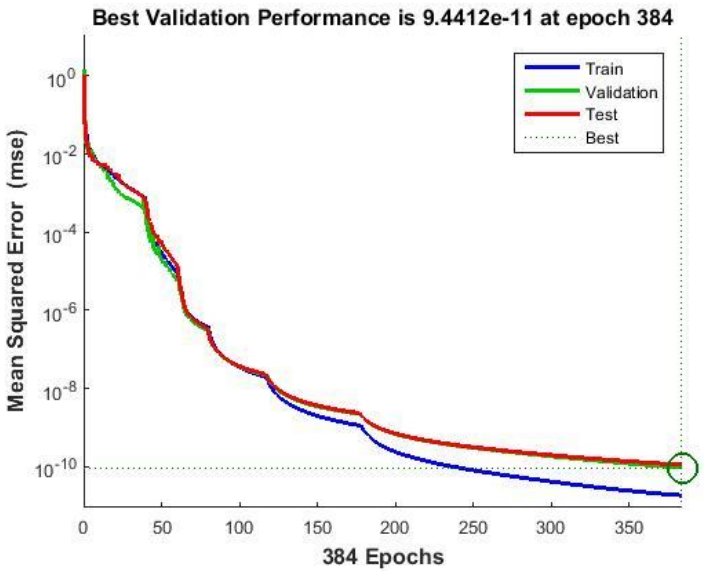


Şekil 6.15. Birinci uygulamanın NARX model test sonuç grafiği

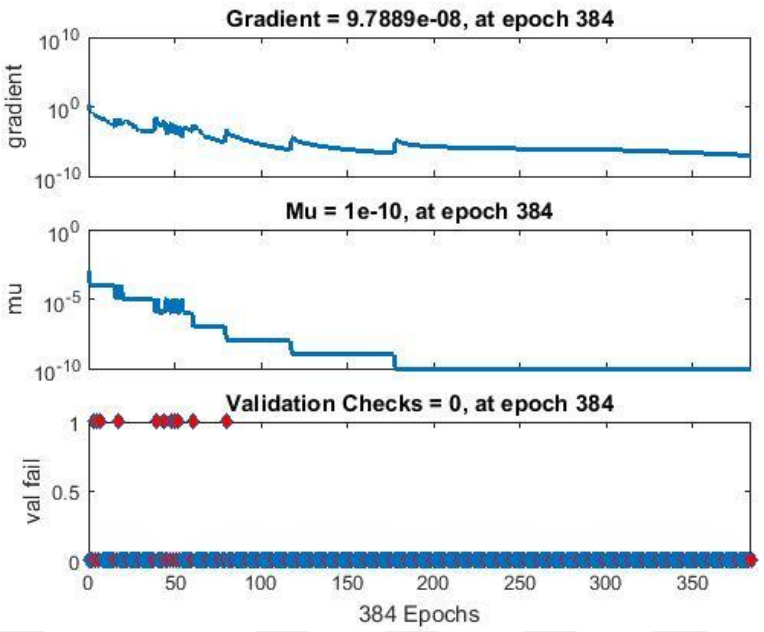
İkinci uygulamada, ilk uygulamada olduğu gibi veri setinden 248'i eğitim, test ve doğrulama için, 249'u ise ayrıca test etmek için kullanılmıştır. Eğitim 384 iterasyonla sonlanmıştır. Bu uygulama sonucu elde edilen grafikler Şekil 6.16, Şekil 6.17, Şekil 6.18, Şekil 6.19, Şekil 6.20, Şekil 6.21, Şekil 6.22 ve Şekil 6.23'de verilmiştir.



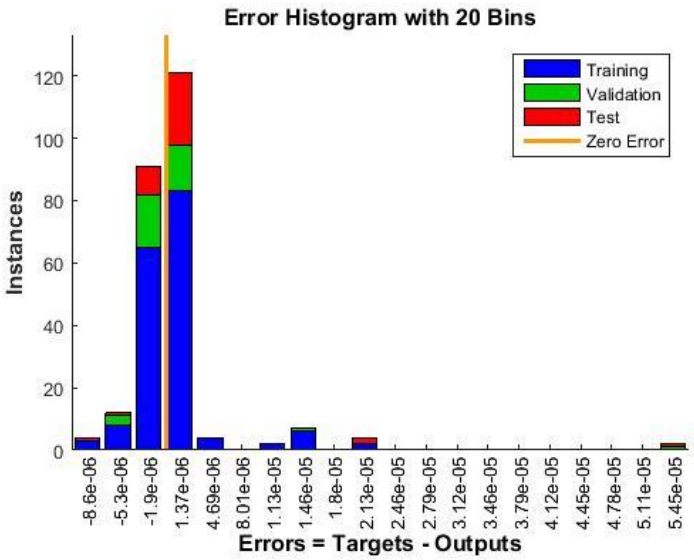
Şekil 6.16. İkinci uygulamanın regresyon analizi grafiksel sonuçları



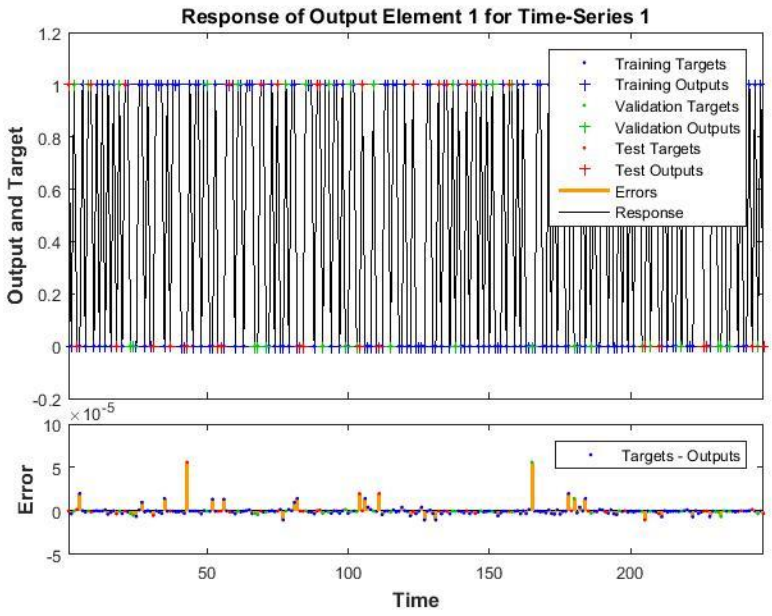
Şekil 6.17. İkinci uygulamanın NARX modelinin performans grafiği



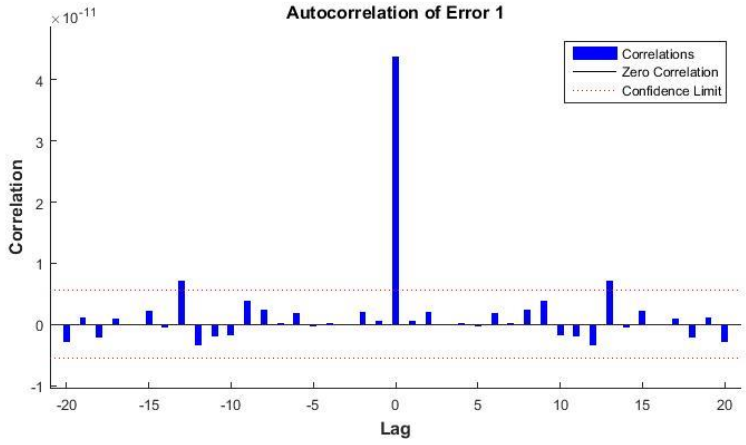
Şekil 6.18. İkinci uygulamanın eğitim durumu grafiği



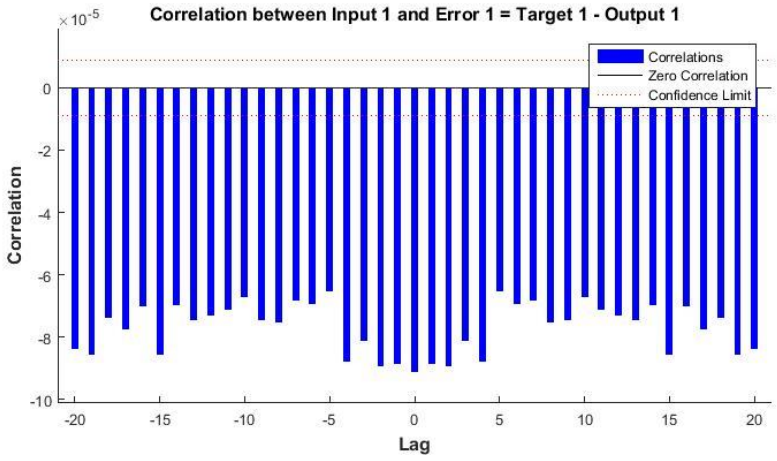
Şekil 6.19. İkinci uygulamanın hata histogram grafiği



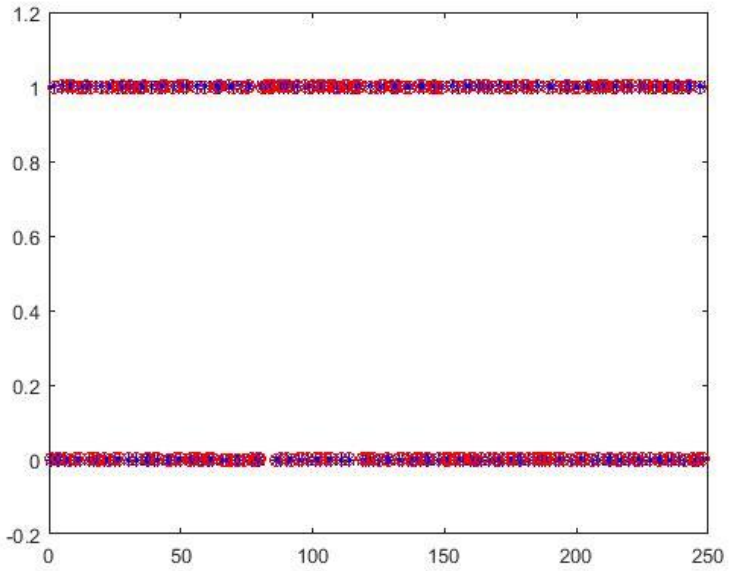
Şekil 6.20. İkinci uygulamanın zaman serisi eğitim, test ve doğrulama sonuç grafiği



Şekil 6.21. İkinci uygulamanın otokorelasyon hatası grafiği

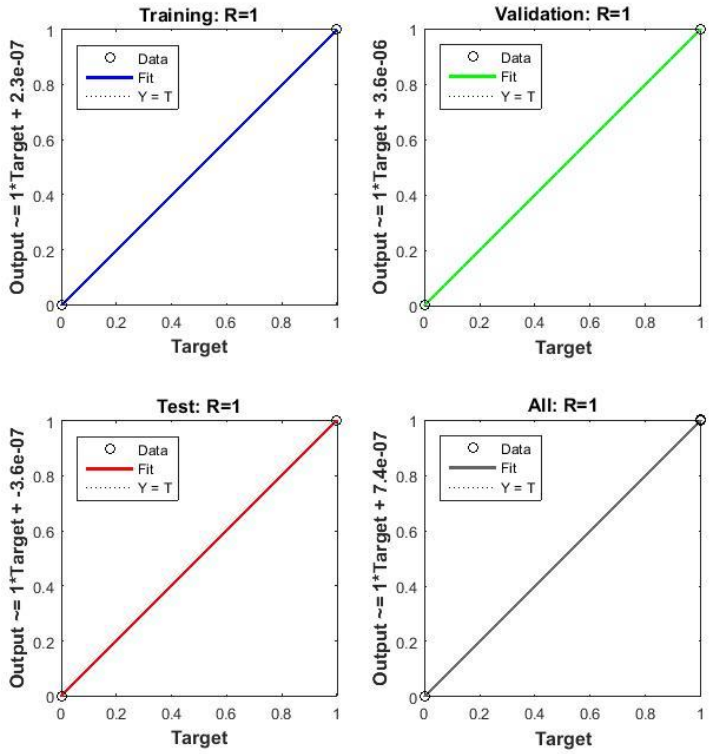


Şekil 6.22. İkinci uygulamanın korelasyon grafiği

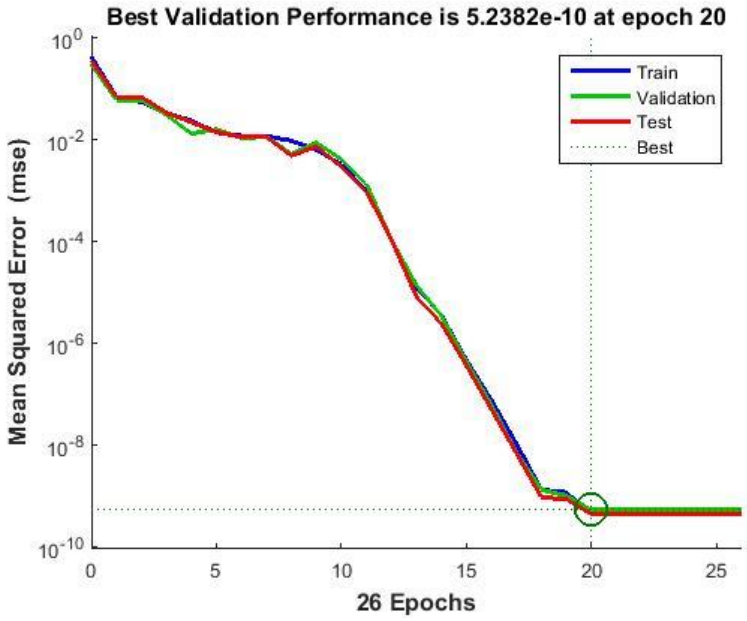


Şekil 6.23. İkinci uygulamanın NARX model test sonuç grafiği

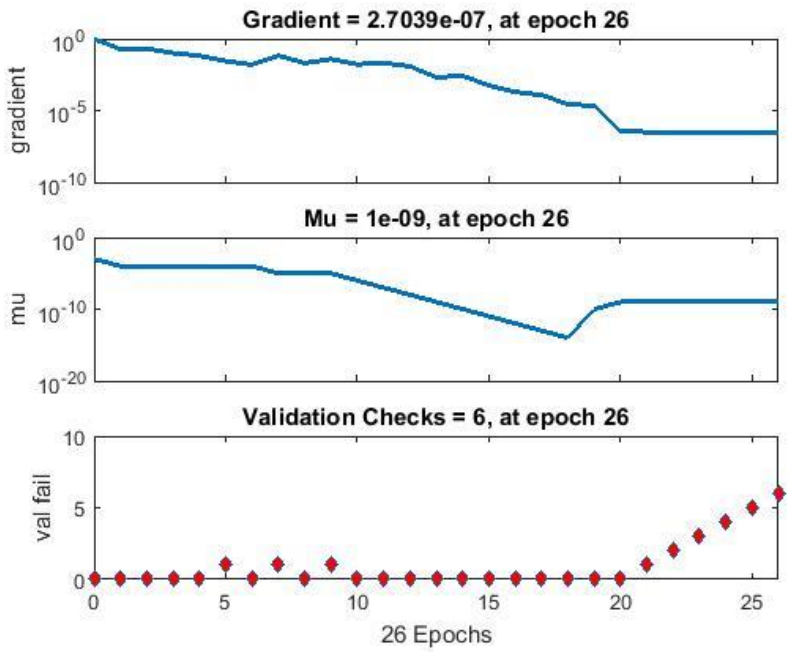
Üçüncü uygulamada, mesaj setindeki verilerin 500'ü eğitim, test ve doğrulama için, 500'ü ise ayrıca test amaçlı kullanılmıştır. Eğitim 26 iterasyonla sonlanmıştır. Bu uygulama sonucu elde edilen grafikler Şekil 6.24, Şekil 6.25, Şekil 6.26, Şekil 6.27, Şekil 6.28, Şekil 6.29, Şekil 6.30 ve Şekil 6.31'de verilmiştir.



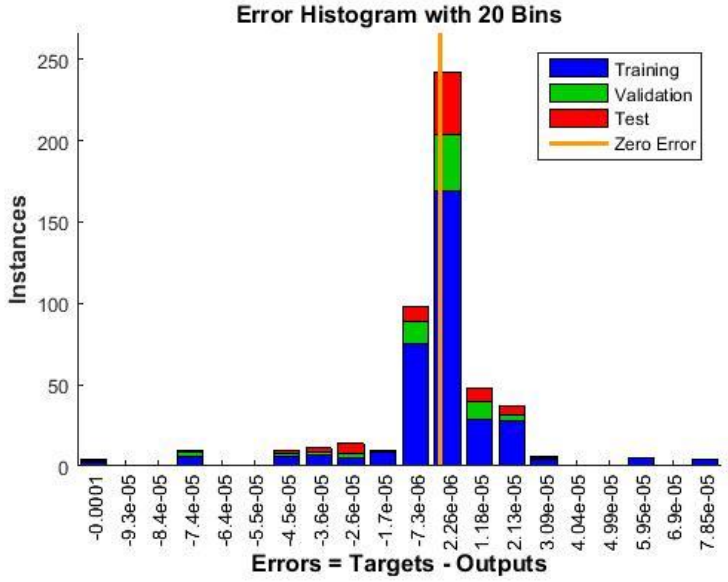
Şekil 6.24. Üçüncü uygulamanın regresyon analizi grafiksel sonuçları



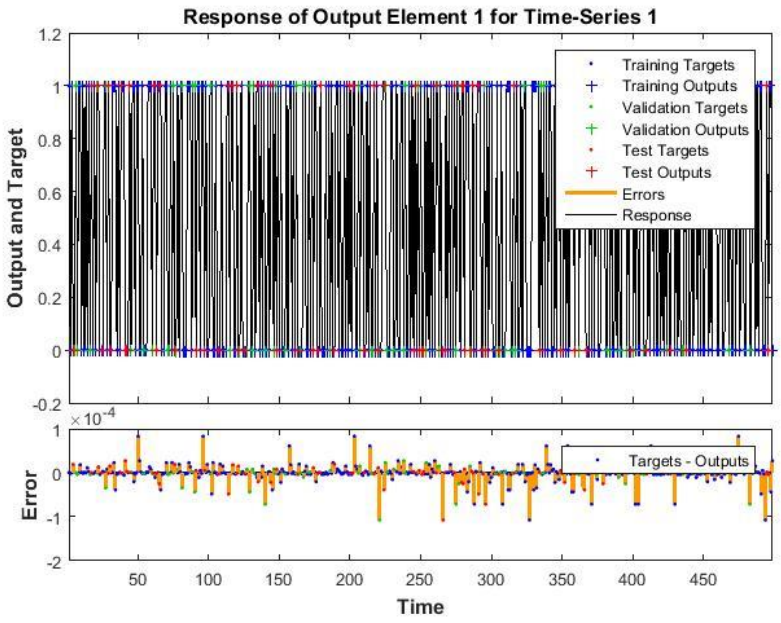
Şekil 6.25. Üçüncü uygulamanın NARX modelinin performans grafiği



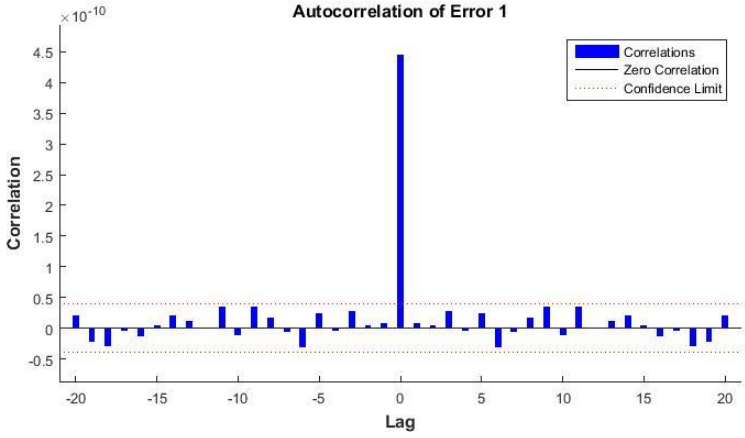
Şekil 6.26. Üçüncü uygulamanın eğitim durumu grafiği



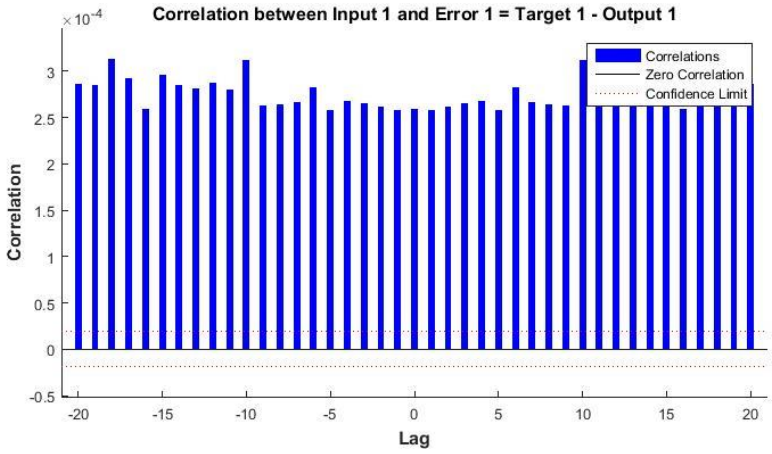
Şekil 6.27. Üçüncü uygulamanın hata histogram grafiği



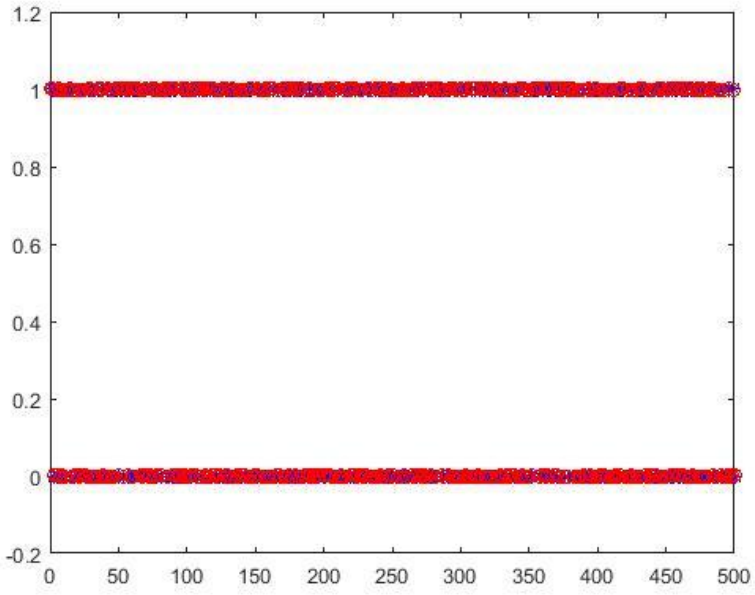
Şekil 6.28. Üçüncü uygulamanın zaman serisi eğitim, test ve doğrulama sonuç grafiği



Şekil 6.29. Üçüncü uygulamanın otokorelasyon hatası grafiği



Şekil 6.30. Üçüncü uygulamanın korelasyon grafiği



Şekil 6.31. Üçüncü uygulamanın NARX model test sonuç grafiği

Doğrusal olmayan oto regresif sistem tanılama metodunun (NARX) yapay sinir ağına uygulanması ile kararlı ve hızlı sistem modellemesi sağlanmıştır. Elde edilen veriler, yapay sinir ağı kullanılarak incelenmiş ve meydana getirilen zaman serisi üzerinden ileriye yönelik kestirimlerde bulunulmuştur. Yapılan her üç uygulamada da ağın başarılı performans gösterdiği belirlenmiştir.

7. TARTIŞMA VE SONUÇLAR

Endüstriyel otomasyon ve otomotiv endüstrisinde zamanlamanın kritik önem taşıdığı uygulamalar gittikçe artmaktadır. Mesajların iletimi esnasında dikkate alınması gereken mesaj gecikmesi ve iletimin güvenilirliği gibi parametreler verimlilik açısından önemlidir. CAN öncelik tabanlı olması, erişim kontrolü sağlaması, sınırlandırılmış mesaj uzunluğu ve olumlu/olumsuz onay mekanizması olması gibi özellikleri sayesinde, gerçek zamanlı kompleks uygulamalar için uygun bir iletişim ağıdır.

Bir CAN ağı, iki ya da daha fazla düğümün, CAN protokolü ile bir CAN bus'a bağlanmasıyla oluşur. CAN protokolü, seri haberleşmeyi kullanır. İletilecek mesajlara öncelik atanarak, hattı kimin öncelikli olarak kullanacağı belirlenir. Bunun için, CAN mesajlara tanımlayıcılar (ID) atanır. ID'si daha küçük olan mesaj, diğerlerinden daha önceliklidir. CAN ağının performansı, öncelik karar mekanizmasına bağlıdır.

CAN, gerçek zamanlı bir iletişim ağıdır. Gerçek zamanlı haberleşme sistemlerinde, iletimin zamanında ve aksamadan gerçekleştirilebilmesi için zamanlanması gerekir. Gerçek zamanlı sistemlerde zamanlama politikaları, her koşulda mesajların son teslim süreleri içinde iletilmesini sağlamalıdır.

Bir CAN ağı tasarlanırken, sistemin tüm özellikleri her zaman bilinmeyebilir. Çalışma esnasında sistemde mesaj akışında değişiklikler olabilir. Bu durumda sistemin dinamik bir ortamda sağlıklı çalışabilmesi için gerekli değişikliklerin yapılabilmesine izin vermesi gerekmektedir. Bu durumda dinamik zamanlama kullanılmalıdır.

Dinamik zamanlama algoritmaları, iletilecek bir sonraki mesajı çevrim içi belirler. Bunun için bazı öncelikleri baz alır. İletim öncelikleri, işlemlerin kritik ve önemli görevleri içermesine göre değerlendirilir. Sistemde herhangi bir değişiklik olduğunda, zamanlayıcı yeni duruma göre yeniden işleme alıp mesajları sıralar. Bu özellik, haberleşme sistemine yüksek seviyede esneklik

sağlar. Esneklik, sistemdeki bileşenlerin birlikte düzgün olarak çalışabilmeleri için önemlidir. Kullanıcılar, sadece farklı üreticilerin ekipmanlarını kullanmayı değil, aynı zamanda ekipman ekleme ya da çıkarmayı ya da konfigürasyonunu sistemi kapatmandan anında değiştirebilmeyi isterler.

Önceliklerin en uygun şekilde atanması, ağ bant genişliğinin maksimum düzeyde kullanılmasını sağlarken, ağın zamanlanabilir olmasını, yani tüm mesajların zaman kısıtlamalarını ya da son teslim sürelerini karşılamasını sağlar. Bununla birlikte, dinamik zamanlamanın dezavantajı, sistemin sürekli hattı daha yüksek öncelikli mesaj için taraması sebebiyle aşırı çalışma yükü getirmesidir.

Bu tez çalışmasında, dinamik öncelikli EDF zamanlama algoritması baz alınarak yeni bir yaklaşım ortaya konulmuştur. Bu zamanlama tekniğinde mesajın son teslim süresi, periyoduna eşit olmalıdır (D=T). EDF öncelik atama düzeni, mutlak son teslim süresine göre mesaj önceliklerini atar. Daha önceki son teslim süresine sahip olan mesaj, daha yüksek önceliklidir. Bu mesaj, önceliği alır ve önce iletilir.

Bu tez çalışması çeşitli aşamalardan oluşmaktadır. İlk aşamada, CAN ağını simule etmek için Linux tabanlı SocketCAN uygulaması kullanılmıştır. SocketCAN'ın temel amacı, Linux ağ katmanı üzerine kurulan kullanıcı alanı uygulamalarına bir soket arabirimi sağlamaktır. Yaygın olarak bilinen TCP / IP ve Ethernet ağlarının aksine, CAN veri yolu Ethernet gibi hiçbir MAC katmanı adresleme özelliğine sahip olmayan bir yayın (broadcast) ortamıdır. Mesajların ID'leri, CAN-bus'ta hakemlik için kullanılır. Bu nedenle, tüm mesajların ID değerleri birbirinden farklı olacak şekilde seçilmiştir.

İkinci aşamada oluşturulan ağ üzerindeki mesajlara öncelik-temelli bir zamanlama algoritması uygulanarak, mesaj önceliklerinin dinamik olarak değiştirilmesi sağlanmıştır. CAN ağı tasarlanırken, gerçek zamanlı mesaj trafiğini desteklemesi ve uygulaması temel alınmıştır. Tasarlanan ağ tek bir CAN bus üzerinde, 17 farklı düğümden oluşmaktadır. 17 farklı ID değerine sahip olan

mesajlar, ağ üzerinde random olarak üretilmiştir. Sistem esnek bir yapıya sahip olduğu için ağdaki düğüm ve CAN bus sayısı artırılabilir özelliktedir.

Geliştirilen algoritmada mesajların önceliklendirmesinin dinamik olması ve CAN'ın doğasından kaynaklanan en düşük ID değerine sahip olan mesajın daha önce iletilebilmesi için, son teslim süresi en erken olan mesajın ID'sinden CAN bus üzerinde mesaj gönderen düğüm sayısı kadar eksiltme yapılmıştır.

Üçüncü aşamada CAN ağının mesaj gönderme koşullarını modellemek ve kontrol etmek için yapay sinir ağı kullanılmıştır. SocketCAN uygulaması ile elde edilen CAN trafiği mesajlarından oluşan veri seti kullanılarak yapay sinir ağı eğitilmiş ve modellenmiştir. Farklı kaotik zaman serileri modelleri ile başa çıkmak ve CAN ağındaki mesajların dinamik zamanlanmasının benzetimini yapmak için NARX modelinin kullanılması uygun olarak görülmüştür. NARX modelinin dinamik bir yapay sinir ağı olması gelecekteki değerleri tahmin etmek için bir ya da daha fazla zaman serisinin geçmiş değerlerini kullanmasından kaynaklanmaktadır. NARX, dışsal bağımlılıkları modellemek suretiyle sonuçları iyileştirmeye yardımcı olacak dışsal girdiyi modelleme esnekliğine sahiptir. NARX modeli, tahmin için yararlı olabilecek ekstra bilgilerin kullanılmasına izin verir. Geleneksel geribeslemeli ağ yapılarına göre NARX ağları, daha hızlı yakınsamakta ve daha etkili bir öğrenme ortaya koymaktadır.

Bu tez çalışması, gerçek zamanlı sistem tasarımcıları için uygulama karmaşıklığına karşı verimliliği dengelemek üzere geniş bir pratik çözüm yelpazesi sunmaktır. Simülasyon sonuçlarının, sunulan çeşitli yükler ve trafik koşulları altında elde edilen analitik sonuçlar ile tutarlı olduğu görülmektedir.

Tasarlanan CAN ağı, yüksek öncelikli mesajların son teslim sürelerini karşıladığını ve daha düşük öncelikli mesajların yüksek öncelikli mesajları engellemediğini kanıtlamıştır. Geliştirilen bu yöntem ile kullanıcılar, sistemi kapatmadan konfigürasyon değişiklikleri yapabileceklerdir. Böylece zaman kaybı söz

konusu olmaz. Tasarlanan sistem esnek bir yapıya sahip olduğu için ağdaki düğüm ve CAN bus sayısı artırılabilir özelliktedir.

Bu çalışmada, bir CAN ağının Linux ortamında zamanlanması ve çevrim içi zamanlanmış bir CAN ağının NARX modeli kullanılarak modellenmesi literatürde ilk kez çalışılmıştır. NARX metodunun yapay sinir ağına uygulanması ile kararlı ve hızlı sistem modellemesi sağlanmıştır. Modelleme sonuçları, öncelik gibi harici verilerin dahil edilmesinin, CAN ağlarındaki mesaj trafiği ile ilgili doğru bir tahmin elde edilebileceğini göstermektedir.

Gelecekteki çalışmalarda, farklı senaryolar için önerilen algoritma geliştirilerek, düşük öncelikli mesaj sorununun üstesinden gelecek şekilde yeni bir kuyruk modeli tasarlanabilir. Geliştirilen öncelik temelli SocketCAN uygulamaları pek çok endüstriyel ağda etkili olarak kullanılabilir. Bununla birlikte, yapılan çalışma farklı mesaj öncelikleri ile karşılaştırılarak test edilebilir ve simülasyon verileri, gerçek verilerle değiştirilerek, sistemin performansı test edilebilir. Ayrıca, modelleme için genetik algoritma, bulanık mantık ve benzeri metodlarla alternatif yöntemler denenebilir.

KAYNAKLAR

Anwar, K., Khan, Z.A., 2007. Dynamic Priority Based Message Scheduling on Controller Area Network. Proceedings of International Conference on Electrical Engineering (ICEE), 1–6 April.

Aras, Ç.M., Kurose, J.F., Douglas, S.R., Schulzrinne, H., 1994. Real Time Communication in Packet-Switched Networks. Proceedings of The IEEE, 82(1), 122-139.

Audsley, N.C., A. Burns, M.F. Richardson, A.J. Wellings, 1991. Hard Real-Time Scheduling: The Deadline Monotonic Approach. Proceedings of 8th IEEE Workshop on Real-Time Operating Systems and Software.

Ayeni, J.A., Odion, A.E., Ogbormor-Odikayor, I.F., 2014. An Analytical Survey of Real Time System Scheduling Techniques. International Journal of Science and Research, 3(10), 1774- 1778.

Aysan, H., Thekkilakattil, A., Dobrin, R., Punnekkat, S., 2010. Efficient fault tolerant scheduling on controller area network (CAN). 15th International Conference on Emerging Technologies and Factory Automation, 13-16 September.

Berkay, A., Şeker, M., Esin, E., 2003. Kontrolör Alan Ağı için Delphi Programı Aracılığı ile Geliştirilecek Olan Yazılımlarda Kullanılmak Üzere Hazırlanan Delphi Bileşenleri. M.Elektrik - Elektronik - Bilgisayar Mühendisliği 10. Ulusal Kongresi, 18-20 Eylül, İstanbul, 476-479.

Bini, E., Buttazzo, G.C., 2004. Schedulability analysis of periodic fixed priority systems. IEEE Transactions on Computers, 53(11), 1462-1473.

Bosch, R., CAN Specification Version 2.0, Stuttgart, Germany, 1991.

CAN-CiA, 2018. Erişim Tarihi: 19.07.2018. https://www.can-cia.org/about-us/

Chen, M., Yen, H.W., 2012. An Online RBF Network Approach for Adaptive Message Scheduling on Controller Area Networks. Journal of Information Science and Engineering, 28, 503-519.

Coşkun, N., Yıldırım, T., 2005. Yapay Sinir Ağları İle Sayısal Devrelerde Gecikme Kestirimi. EMO II. İletişim Teknolojileri Ulusal Sempozyumu, 17-19 Kasım, Adana.

Cooper, M., 2010. Advanced Bash-Scripting Guide. Published by Lulu.com, 520p.

Davis, R., Burns, A., Bril, R.J., Lukkien, J.J., 2007. Controller Area Network (CAN) Schedulability Analysis: Refuted Revisited and Revised. Real-Time Systems, 35(3), 239-272.

Dobrin, R., Fohler, G., 2001. Implementing off-line message scheduling on controller area network (CAN). Proceedings of Emerging Technologies and Factory Automation. Antibes-Juan Les Pins, 241-245.

Dobrin, R., 2005. Combining Off-line Schedule Construction and Fixed Priority Scheduling in Real-Time Computer Systems. Mälardalen University Doctoral Thesis, 138p, Västerås, Sweden.

Duzenek, D., Stare, Z., Kovacic, Z., 2007. GA-based on-line optimization of CAN message scheduling. The 15th Mediterranean Conference on Control and Automation, 27-29 June, Greece, 1-6.

Ercek, E., Erdoğdu, S., Uluat, M. F., 2005. Gerçek Zamanlı Sistemlerde UML Uygulamaları. EMO II. İletişim Teknolojileri Ulusal Sempozyumu, Adana.

Farsi, M., Ratcliff, K., Barbosa, M, 1999. An overview of controller area network, Computing & Control Engineering Journal, 10(3), 113-120.

Fohler, G., Lenvall, T., Buttazzo, L.G., 2001. Improved Handling of Soft Aperiodic Tasks in Offline Scheduled Real-Time Systems using Total Bandwidth Server, 8th IEEE International Conference on Emerging Technologies and Factory Automation, 15-18 October, France, 151-157.

Fonseca, J.A., Almeida, L.M., 1999. Using a Planning Scheduler in the CAN Network. Proc. IEEE Emerging Technologies and Factory Automation ETFA'99, Barcelona, Spain, October, 815-821.

Fuster, S., Rodríguez, F., Bonastre, A., 2005. Software-Based EDF Message Scheduling on CAN Networks. Proceedings of the Second International Conference on Embedded Software and Systems (ICESS'05), Xian, China.

Gao, X., Li, L.S., 2011. Analysis and Research of Real Time Ability of Message Transmission in CAN Bus, International Conference on Control Automation and Systems Engineering (CASE), 1-3, Singapore.

Github, 2010. Erişim Tarihi: 15.09.2017. https://github.com/rhyttr/Socket CAN/commit/db198f35c3e9ab17c62 ed63d238f1c4afa9acb19.

Github, 2017. Erişim Tarihi: 15.09.2017. https://github.com/rhyttr/SocketCAN.

Harkut, D.G., Ali, M.S., Lohiya, P., 2014. Real-time scheduling algorithms for wireless sensor network. Circuits and Systems: An International Journal CSIJ, 1(1), 11-18.

Hell, M., M., 2016. The new dynamic parameters of CAN FD. CAN Newletter.

Hui, C., Zhidong, Q., Xiaoming, J., 2008. An On-Line Hard Real-Time Communication Scheduler for Ultra-dependable System. International

Conference on Embedded Software and Systems (ICESS2008), IEEE, 49 – 56.

Hwang, M., Choi, D., Kim, P., 2011. Least Slack Time Rate First: an Efficient Scheduling Algorithm for Pervasive Computing Environment. Journal of Universal Computer Science, 17(6), 912-925.

Jyothi, M.N, Dinakar, V., Teja, N.S.S.R., Kishore, K.N., 2015. NARX Based Short Term Wind Power Forecasting Model. TELKOMNIKA Indonesian Journal of Electrical Engineering, 15(1), 20-25.

Kopetz, H., 1997. Real-Time Systems, Design principles for Distributed Embedded Applications, Kluwer Academic Publishers, Boston, 0-7923-9894-7.

Kumar, B. V., Ramesh, J., 2016. Improved Automotive CAN Protocol Based on Payload Reduction and Selective Bit Stuffing. Circuits and Systems, 7, 3415-3429.

Labde, S., Patel, S., Shukla, M., 2017. Time Series Regression Model for Prediction Of Closing Values of the Stock using an Adaptive NARX Neural Network. International Journal of Computer Applications, 158(10), 29-34.

Lin, C.K., Yen, H., Chen, M., Hwang, C., 2006. A Neural Network Approach for Controller Area Network Message Scheduling Control. IAENG International Journal of Computer Science, 33(1), 36-41.

Lin, T., Horne, B.G., Tino, P., Giles, C.L., 1996. Learning long-term dependencies in NARX recurrent neural networks. IEEE Transactions on Neural Networks, 7(6), 1329-1338.

Liu, C.L., Layland, J.W. 1973. Scheduling algorithms for multiprogramming in a hard real-time environment. Journal of ACM 20(1), 40–61.

Leung, J.,Y.,T., Whitehead, J., 1982. On the complexity of fixed-priority scheduling of periodic, real-time tasks. Performance Evaluation, 2(4), 237-250.

Livani, M.A., Kaiser, J., Jia, W.J., 1998. Scheduling Hard and Soft Real-Time Communication in The Controller Area Network (CAN). Proceedings of the 23rd IFAC Workshop on Real-Time Programming, June, 13-18, China.

Livani M.A., Kaiser J., 1998. EDF consensus on CAN bus access for dynamic real-time applications. Rolim J. (eds) Parallel and Distributed Processing. IPPS 1998. Lecture Notes in Computer Science, 1388. Springer, Berlin, Heidelberg.

Malcolm, N., Zhao, W., 1995. Hard real-time communication in multiple Access Networks. The International Journal of Time-Critical Computing Systems, Real-Time Systems, 8(1), 35-77.

Manonmani, A., Thyagarajan, T., Elango, M., Sutha S., 2016. Modelling and control of greenhouse system using neural networks. Transactions of the Institute of Measurement and Control, 40(3), 918 – 929.

Mathworks, 2018. Design Time Series NARX Feedback Neural Networks. Erişim Tarihi: 05.04.2018. https://www.mathworks.com/help/nnet/ug/design-time-series-narx-feedback-neural-networks.html.

Michta, E., 2005. Scheduling Systems. Sydenham, P.H., Thorn, R., (Ed.), Handbook of Measuring System Design, John Wiley and Sons, 1648p.

Natale, M.D,. 2000. Scheduling the CAN Bus with Earliest Deadline Techniques. In Proceedings of the 21st IEEE Real-Time Systems Symposium, December 2000, 259–268.

Navet, N., Songa, Y.-Q., Simonot, F., 2000. Worst-case deadline failure probability in real-time applications distributed over controller area network. Journal of Systems Architecture, 46, 607-617.

Nyce, D., D., 2016. Position Sensors, John Wiley & Sons, 400p, New Jersey.

Nyguen, V-A-Q., Tran, T-V., 2017. Analysis of packet loss on scheduling over wireless real-time control system. Seventh International Conference on Information Science and Technology (ICIST), 29-31, Da Nang, Vietnam.

Oliveira, M.P., Fernandes, A.O., Campos, S.V.A., Zuquim, A.L.A.P., Mata, J.M., 2003. Guaranteeing Fault Tolerance through Scheduling on a CAN bus. The International CAN Conference (ICC2003), 15-22.

Oliveira, C.A.S., Pardalos, P.M., Querido, T.M., 2005. A Combinatorial Algoritm for Message Scheduling on Controller Area Networks, International Journal of Operational Research, 1, 160-171.

Ouni, S., Kamoun, F., 2004. Real Time Message Guarantees with EDF Based Access Protocols over Industrial Networks. International Conference on Computing, Communication and Control Technologies (CCCT'04), August, USA.

Öner D., 2003. Bilgisayar Ağları, Papatya Yayıncılık, 312s, İstanbul.

Öztemel, E., 2006. Yapay Sinir Ağları, Papatya Yayıncılık, 238s, İstanbul.

Pedreiras, P.,, Almeida, L., 2002. EDF Message Scheduling on Controller Area Network. Computing & Control Engineering Journal, 13(4), 163-170.

Pinho, L.M., Vasques, F., 2003. Reliable Real-Time Communication in CAN Networks. IEEE TRANSACTIONS ON COMPUTERS, 52(12), 1594-1607.

Puiu, D., Moldoveanu, F., Cernat, M., 2010. Message Scheduling For Real-Time Communications. The 5th International Conference on Interdisciplinarity in Education ICIE'10, June 17-19, Tallinn, Estonia, 321-326.

Rufino, J., Verissimo, P., 1995. A study on the inaccessibility characteristics of the controller area network. Proceedings of the second international CAN conference, October, London, UK. 7.12-7.21.

Rouhifar, M., Ravanmehr, R., 2015. A Survey on Scheduling Approaches for Hard Real-Time Systems. International Journal of Computer Applications, 131(17), 41-48.

Saadi, I.M.A., 2013. Dynamic Message Transmission Scheduling Using Can Protocol. International Journal of Scientific & Technology Research, 2(9), 158-163.

Schmidt, K., Schmidt, E.G., 2007. Systematic Message Schedule Construction for Time-Triggered CAN. IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, 56(6), 3431-3441.

Shi Z., Zhang F., 2012. An analytical model of the CAN bus for online schedulability test. Proceedings of the third analytic virtual integration of cyber-physical systems workshop (AVICPS), held inconjunction with the 33rd IEEE real-time systems symposium (RTSS2012)

Shinde, V., Biday, S.C., 2017. Comparison of Real Time Task Scheduling Algorithms. International Journal of Computer Applications, 158(6), 37-41.

Shokry, H., Shedeed, M., Hammad, S., Shalan, M., Wahdan, A., 2009. Hardware EDF scheduler implementation on controller area network controller. In Proceedings of 4th International IEEE Design and Test Workshop (IDT) 2009, Riyadh, Saudi Arabia.

Shoukry, Y.,, Shokry, H.,, Hammad, S., 2011. Distributed Dynamic Scheduling of Controller Area Network Messages for Networked Embedded Control System. 18th The International Federation of Automatic Control, Milano, Italy, August 28 - September 2, 1959-1964.

Sojka, M., Píša, P., Petera, M., Špinka, O., Hanzálek, Z., 2010. A Comparison of Linux CAN Drivers and their Applications. 5th IEEE International Symposium on Industrial Embedded Systems (SIES), 7-9 July, 18-27.

Sojka, M., Píša, P., Hanzálek, Z., 2014. Performance evaluation of Linux CAN-related system calls. 10th IEEE Workshop on Factory Communication Systems (WFCS 2014), 5-7 May, Toulouse, France, 1-8.

Tindell, K., Burns, A., 1994. Guaranteeing Message Latencies on Control Area Network (CAN). 1st International CAN Conference 94, 1-11.

Tindell, K.W., Hansson, A.H., Wellings. A. J., 1994. Analysing Real-Time Communications: Controller Area Network (CAN). IEEE Real-Time Systems Symposium, 259-265.

Tindell, K., Burns, A., Wellings, A., 1995. Calculating Controller Area Network (CAN) Message Response Times. Control Engineering Practice, 3(8), 1163-1169.

Tao, B., Hu, L., Wu, Z., Yang, G., 2005. Flexible Fuzzy Priority Scheduling of The CAN Bus. Asian Journal of Control, December, 401-413.

Wanke, C., Cheng, L., Wei, Z., Fengchun, S., 2009. A Real-Time Planning-Based Scheduling Policy with CAN for Automotive Communication Systems. World Congress on Computer Science and Information Engineering, 31 March-2 April, Los Angeles, CA, USA, 186-191.

Wen, P., Cao, J. Li, Y., 2007. Design of High-performance Networked Real-time Control Systems, IEEE IET Control Theory and Applications, 1(5), 1329-1335.

Xie, H., Tang, H., Liao, Y.-H., 2009. Time series prediction based on NARX neural networks: An advanced approach. International Conference on Machine Learning and Cybernetics, China, 3, 1275–1279.

Yong, W., 2003. A scheduling algorithm for CAN bus. National University of Singapore, M.Sc. Thesis, 94p, Singapore.

Zuberi, K.M., Shin, K.G., 1995. Non-Preemptive Scheduling of Messages on Controller Area Network for Real-Time Control Applications. Proceedings of I995 IEEE Real-Time Technology and Applications Symp., May 1995, Chicago.

Zuberi, K.,M.,, Shin, K.G., 1997. Scheduling Messages on Controller Area Network for Real-Time CIM Applications. IEEE Transactions On Robotics and Automation, 13(2), 310-314.

Zuberi, K.M., Shin, K.G., 2000. Design and implementation of efficient message scheduling for controller area network. IEEE Transactions on Computers, 49(2), 182-188.

EKLER

EK A. Birinci ve İkinci Uygulamada Kullanılan MatLab Kodları

EK B. Üçüncü Uygulamada Kullanılan MatLab Kodları

EK C. Tasarlanan CAN Ağı Topolojisi

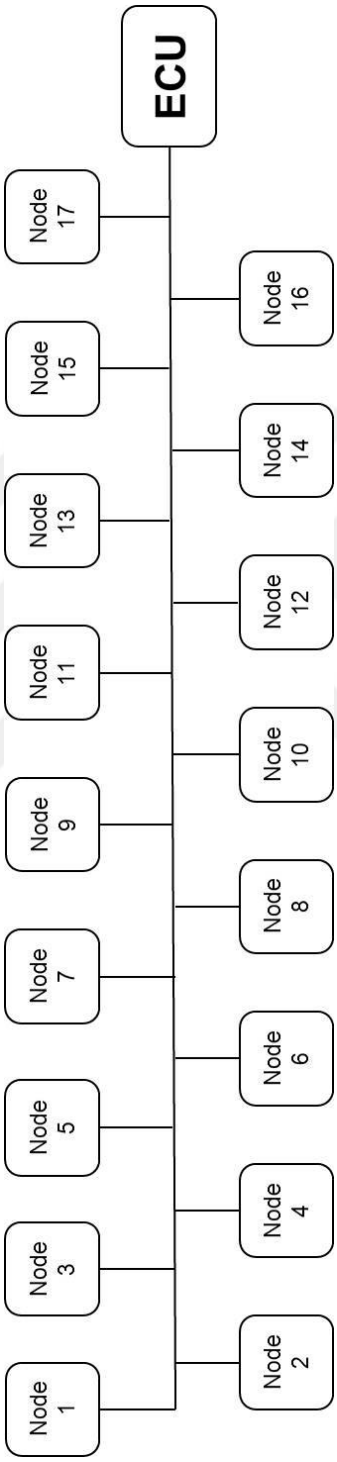
EK A. Birinci ve İkinci Uygulamada Kullanılan MatLab Kodları

```
close all;
clear all

load X.txt
load T.txt

x=X(1:248)';
y=T(1:248)';

xtest=X(249:497)';
ytest=T(249:497)';

xo=x;
yo=y;

xot=xtest;
yot=ytest;

inputDelays = 0:0;
feedbackDelays = 1:1;
hiddenLayerSize = 10;

net = narxnet(inputDelays,feedbackDelays,hiddenLayerSize);
[x,xi,ai,t] = preparets(net,num2cell(xo),{},num2cell(yo));

net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
[net,tr] = train(net,x,t,xi,ai);
y = net(x,xi,ai);

view(net)

[xt,xit,ait,tt] = preparets(net,num2cell(xot),{},num2cell(yot));

ytest = net(xt,xit,ait);

TS = size(tt,2);

plot(1:TS,cell2mat(tt),'b*',1:TS,cell2mat(ytest),'ro');
```

EK B. Üçüncü Uygulamada Kullanılan MatLab Kodları

```
close all;
clear all

load Xa.txt
load Ta.txt

x=Xa(1:500)';
y=Ta(1:500)';

xtest=Xa(500:1000)';
ytest=Ta(500:1000)';

xo=x;
yo=y;

xot=xtest;
yot=ytest;

inputDelays = 0:0;
feedbackDelays = 1:1;
hiddenLayerSize = 10;

net = narxnet(inputDelays,feedbackDelays,hiddenLayerSize);
[x,xi,ai,t] = preparets(net,num2cell(xo),{},num2cell(yo));

net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
[net,tr] = train(net,x,t,xi,ai);
y = net(x,xi,ai);

view(net)

[xt,xit,ait,tt] = preparets(net,num2cell(xot),{},num2cell(yot));

ytest = net(xt,xit,ait);

TS = size(tt,2);

plot(1:TS,cell2mat(tt),'b*',1:TS,cell2mat(ytest),'ro');
```

EK C. Tasarlanan CAN Ağı Topolojisi



Şekil A.1. Tasarlanan CAN Ağı Topolojisi

ÖZGEÇMİŞ

Adı Soyadı : Esin YAVUZ

Doğum Yeri ve Yılı : Isparta, 1977

Medeni Hali : Bekar

Yabancı Dili : İngilizce

E-posta : esinyavuz@sdu.edu.tr

Eğitim Durumu

Lise : Isparta ŞAİK Lisesi, 1995

Lisans : Süleyman Demirel Üniversitesi, Mühendislik Fakültesi, Elektronik & Haberleşme Mühendisliği, 2000

Yüksek Lisans : Süleyman Demirel Üniversitesi, Fen Bilimleri Enstitüsü, Elektronik & Haberleşme Mühendisliği ABD, 2004

Mesleki Deneyim

SDÜ Bilgi İşlem Daire Başkanlığı	2000-2007
SDÜ Mühendislik Fakültesi	2007-…… (devam ediyor)