T.C.
KONYA TEKNİK ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

DAĞITIK SİSTEM YAZILIMLARININ GÜNCELLENMESİNE YÖNELİK YENİ BİR YAKLAŞIM

Abdurrahman KURTVURAN

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Anabilim Dalı

Mayıs-2023
KONYA

TEZ KABUL VE ONAYI

Abdurrahman KURTVURAN tarafından hazırlanan "Dağıtık Sistemlere ait Yazılımların Güncellenmesine Yönelik Yaklaşımlar" adlı tez çalışması …/…/… tarihinde aşağıdaki jüri tarafından oy birliği / oy çokluğu ile Konya Teknik Üniversitesi Lisansüstü Eğitim Enstitüsü Bilgisayar Mühendisliği Anabilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Doç. Dr. Mehmet Akif ŞAHMAN	……..
Danışman Dr. Öğr. Üyesi Alper KILIÇ	……..
Üye Prof. Dr. İsmail BABAOĞLU	……..

Yukarıdaki sonucu onaylarım.

Prof. Dr. Mevlüt UYAN
Enstitü Müdürü

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

DECLARATION PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

İmza

Abdurrahman KURTVURAN

Tarih: 5 Temmuz 2023

ÖZET

YÜKSEK LİSANS TEZİ

DAĞITIK SİSTEM YAZILIMLARININ GÜNCELLENMESİNE YÖNELİK YENİ BİR YAKLAŞIM

Abdurrahman KURTVURAN

Konya Teknik Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Alper KILIÇ

2023, 59 Sayfa

Jüri
Dr. Öğr. Üyesi Alper KILIÇ
Doç. Dr. Mehmet Akif ŞAHMAN
Prof. Dr. İsmail BABAOĞLU

Güncelleme uygulamaları, gelişen teknoloji ve yenilenen ürünleri destekleyebilme açısından her geçen gün giderek ihtiyacı artan uygulamalardır. Elektrikli arabalar ve elektrikli şarj istasyonlarıyla beraber günümüzde çok daha önemli bir hale gelen istasyon ve dijital pompa güncellemesi ise bu çalışmanın ana odağını oluşturmaktadır. Dijital pompalar, istasyonların artan ve değişen isterlerine göre güncel kalması ve güncellenebilir olması gereken cihazlardır. Bir istasyon da birden farklı sistemin entegre hareket etmesi gereken bir yapıya sahiptir. İlgili yapıda yıkama makinası, benzin pompası, gaz pompası ve günümüzde gelişen elektrik pompası birleşenleri olmakla birlikte istasyona göre farklılık gösterebilmektedir ve otomatlar da dahil olmak üzere çok farklı alanlara dallanabilmektedir. Bununla beraber ulusal ve uluslararası benzin istasyonları tüm bu cihazları güncellenebilir tutmak ve gerektiğinde yenilenen yapılara göre güncelleyebilmek hatta gerekli görüldüğü takdirde kullanım amacını değiştirmek isteyebilmektedirler. Bu istenen güncellemeler bazen acil ve kritik olduğu gibi bazen de belli bir rutin içerisinde olabilmektedir. Bu çalışmada belirtilen ihtiyacı giderebilmek için hem platform hem de bağımsız bir güncelleme uygulaması örneği tavsiye edilmektedir. Yapılan kullanıcı ve yazılım geliştirici memnuniyeti gözlemleri sonucunda önceki çalışmalara göre bu çalışmamızda önerilen uygulamanın uygulama bağımlılığının daha düşük ve daha kolay yönetilebilir olduğu gözlemlenmiştir.

Anahtar Kelimeler: Çoklu Güncelleme, Elektronik Pompa, İstasyon Yönetimi

ABSTRACT

MS THESIS

A NEW APPROACH TO UPDATING DISTRIBUTED SYSTEM SOFTWARE

Abdurrahman KURTVURAN

Konya Technical University
Institute of Graduate Studies
Department of Computer Engineering

Advisor: Asst.Prof. Dr. Alper KILIÇ

2023, 59 Pages

Jury
Asst.Prof. Dr. Alper KILIÇ
Asst.Prof. Dr. Mehmet Akif ŞAHMAN
Prof. İsmail BABAOĞLU

Update applications are applications that are increasingly needed in terms of supporting developing technology and renewed products. The station and digital pump update, which has become much more important today with electric cars and electric charging stations, is the main focus of this study. Digital pumps are devices that need to be up-to-date and updatable according to the increasing and changing demands of the stations. In addition to this, a station has a structure that requires more than one system to act integrated. Although these are the main headings of the washing machine, petrol pump, gas pump and the electric pump that is developing today, they can differ according to the station and can branch into many different areas, including vending machines. However, national and international gas stations may want to keep all these devices up-to-date and, if necessary, update them according to the renewed structures, and even change the purpose of use if necessary. These desired updates are sometimes urgent and critical, and sometimes they can be in a certain routine. In this study, a platform- and application-independent update application example is recommended to meet this need. As a result of user and software developer satisfaction observations, it has been observed that application addiction is higher and more easily manageable than previous studies compared to similar applications.

Keywords: Electronic Pump, Multi-Update, Station Management

ÖNSÖZ

Tez çalışmam boyunca değerli katkılarını, yönlendirici desteğini ve anlayışını hiçbir zaman esirgemeyen danışmanım Sayın Dr. Öğr. Üyesi Alper KILIÇ'a;

Tez çalışma süreci boyunca bana imkân sağlayan Mepsan Petrol Cihazları A.Ş.'ye ve çok değerli iş arkadaşlarıma;

Desteklerinden dolayı içtenlikle teşekkür eder ve minnetlerimi sunarım.

Abdurrahman KURTVURAN
KONYA-2023

İÇİNDEKİLER

KISALTMALAR

DVM: Destek Vektör Makinesi
EBR: Edition-Based Redefinition
ECU: Elektronik Kontrol Ünitesi
EV: Elektrikli Araç
FTP: File Transfer Protocol
HTTP: Hyper Text Transfer Protocol
IoT: Internet of Things
KAHSA: Kablosuz Ad-Hoc Sensör Ağı
k-NN: k-En Yakın Komşu
LR: Lojistik Regresyon
LMS: Least Mean Squares
SVM: Support Vector Machine
TCP: Transmission Control Protocol

1. GİRİŞ

İlerleyen teknoloji ve artan elektrikli araç kullanımıyla beraber arabalar da tıpkı telefonlar gibi akıllı cihazlar kapsamına girerek sürekli gelişen ve sürekli değişen bir ekosistemin parçası olmuşlardır. Bugün lüks arabalarda 150'nin üzerinde elektronik kontrol birimi bulunmakta ve yeni gelişmelerle bu sayı artmaktadır (Priyatham, 2019). Bu elektronik kontrol birimleri içerisinde dağıtık mimaride 100 milyondan fazla kod satırı bulunmaktadır (Dajsuren & Brand, 2019). Bu yüksek cihaz sayısı ve yazılım karmaşasıyla arabalar akıllı cihaz kapsamına girmiştir. Araçlardaki cihazlar için bağlantı, kontrol, koordinasyon, bilgi ve eğlence gibi ihtiyaçlar doğrultusunda artan yazılım ihtiyacıyla birlikte bu yazılımları güncel tutmak ve bu yazılımlara yeni özellikler eklenmesini sağlamak veya yazılımda tespit edilen sorunları düzeltmek için güncelleme uygulamalarına ihtiyaç duyulmaktadır (Placho, Schmittner, Bonitz, & Wana, 2020). Bu bağlamda değişen ekosistemle birlikte elektrikli şarj istasyonlarına çevrimiçi erişme ve önden randevu oluşturma gibi gereklilikler oluşmuştur. Bu yönelim de beraberinde halen yaygın olarak kullanılan dijital elektronik benzin istasyonlarına ivme kazandırmıştır.
2017 yılında satılan elektrikli araç sayısı, 2016 yılı ile kıyaslandığında %54'lük önemli bir büyüme oranıyla bir milyon adedi aşmıştır. En büyük elektrikli araç pazarı, küresel elektrikli araç satışlarının yarısından fazlasını oluşturan Çin'dir. Elektrikli araçların kümülatif küresel satışları 2017 sonunda üç milyon adedi aşmıştır ve Çin, bu satışların %40'ını gerçekleştirmişken Amerika Birleşik Devletleri de en büyük ikinci elektrikli araç satıcısı olmuştur. Bu istatistikler, elektrikli araç pazarının küresel düzeyde hızla büyüdüğünü göstermektedir (Faizal, Feng, Zureel, Sinidol, Wong, Jian, 2019). Elektrikli araç satış ve kullanımının bu yüksek ivmeli artışı, elektrikli şarj istasyonlarına olan ihtiyacı da arttırmıştır.

Elektrikli araçların gündelik kullanıma girmesi, hem elektrik hem de ulaşım ağlarının dikkatli bir şekilde incelenmesini gerektirdiğinden elektrikli şarj istasyonlarının konumlandırılması gibi farklı çalışma alanlarına imkân oluşturmuştur. Elektrikli araç şarj istasyonlarının konumlandırılması konusunda yapılan bir çalışma için öncelikle belirli bir bölgedeki elektrikli araç sayısı ve talebi belirlenmekte, daha sonra bu bölgedeki olası şarj istasyonu yerleri ve kapasiteleri hesaplanmaktadır. Bu hesaplamalarda bir optimizasyon modeli kullanılmaktadır. İlgili model, belirlenen bölgedeki elektrikli araç sayısı, seyahat mesafeleri, mevcut şarj istasyonlarının kapasiteleri ve istasyonların konumları gibi faktörleri dikkate alarak mevcut şarj istasyonlarının yerlerinin optimizasyonunu, daha

fazla kapasite ve kapsama alanı elde edilmesini sağlamaktadır (Bayram, Bayhan, 2020). Bu tarz çalışmaların da ortaya koyduğu üzere elektrikli araçlar, hem elektrikli araç istasyonları hem de dijital akaryakıt istasyonlarını dönüştürerek akıllı cihazlar ile akıllı istasyonların çevrimiçi uç birimlerle iç içe girmesinin önünü açmış ve girift yapıda esnek ve güncel kalabilmek önemli bir ihtiyaç haline gelmiştir. Bu da istasyonlar, dispanserler ve elektronik pompalar gibi farklı kullanım türevindeki cihazların güncellenebilmesi ihtiyacını doğurmuştur.

1.1. Tezin Amacı ve Önemi

IoT cihazları güncelleme işlemlerinin düzenli olarak yapılması; güncelleme öncesi, güncelleme anı ve sonrasının tutarlı olması gereken cihazlardır. Bu aşamalardan birinde karşılaşılacak olası bir sorunun cihazın rutin çalışmasını sekteye uğratmaması elzemdir. Bu tez çalışmasında; akaryakıt istasyonlarında bulunan internete bağlı donanımları özdeş 1000 cihazın içinde halihazırda çalışabilen veya cihaz içerisinde daha önceden hiç bulunmayan uygulamaların indirilmesi, güncellenmesi ve çalıştırılmasını sağlayan yeni bir yaklaşım önerilmiş olup önceden yapılmış olan çalışmalardan daha hızlı, çok daha esnek ve daha kolay yönetilebilir bir IoT cihazı güncelleme uygulamasını açıklanacaktır. Bu çalışmada popülerliği ve kullanımı her geçen gün artan akıllı cihazları; test amacıyla kullanılan dijital akaryakıt istasyonları ve öncelikli hedefler arasında yer alan elektrikli şarj istasyonları nasıl daha etkili ve hatasız bir şekilde çevrimiçi olarak güncellenebilir sorusu cevaplandırılacaktır.

Dağıtık mimariler için güncelleme önerisi yaklaşımı, daha önce klasik veri tabanı güncellemelerinde karşılaşılan sorunların çözümlerine ve veri tabanlarında karşılaşılan güncelleme öncesi uygulamanın aktif kullanıcıları rahatsız etmemesi, güncelleme öncesi uygulamanın ve güncelleme sonrası uygulamanın aynı anda mevcut olması, güncelleme anında önceki ve sonraki versiyon arasındaki geçiş şartlarını sağlaması gerekliliğine ilişkindir. Choi'nin çalışmasında çevrimiçi uygulama yükseltmelerinde yaşanan sorunları azaltmak amacıyla Oracle veri tabanı yönetim sistemi için sürüm tabanlı güncelleme önerilmiştir. Bahsedilen çalışmada uygulamanın yeni bir sürümü eski sürümden ayrı olarak saklanmakta ve bu sayede aktif çalışan uygulama daha az kesintiye uğramaktadır. Çalışmada uygulamanın yeni sürümünün eski sürümle birlikte mevcut tutulması, yeni sürümün eski sürümden ayrıt edilmesi ve yeni sürümün eski sürümün kullanımı devam ederken çalışmaya alınması gibi adımları nasıl gerçekleştirileceği ayrıntılı olarak

incelenmektedir. Bu yöntem, uygulamanın kullanıcılarına kesintisiz bir hizmet sunulmasını mümkün kılar.

Ayrıca, Choi'nin çalışması, sürüm tabanlı yeniden tanımlama özelliğinin Oracle veri tabanı yönetimi için önemli bir avantaj sağladığını belirtmektedir. Bu özellik, bir uygulamanın yeni sürümünün çıkartılması ve yükseltilmesi sırasında, eski sürümün ve yeni sürümün birlikte çalışabilmesini sağlayarak uygulamanın daha hızlı bir şekilde yükseltilmesini ve daha az etkilenmesini sağlamaktadır.

Sonuç olarak, Choi'nin çalışması, online uygulama yükseltmeleri sırasında yaşanan sorunları azaltmak ve uygulamanın kesintisiz bir şekilde kullanılmasını sağlamak için önemli bir yöntem sunmaktadır (Choi, 2009).

Choi'nin geleneksel veri tabanları için çalıştığı bu geçişin sağlanması için sunucu veri tabanı da önem arz etmektedir. Kablosuz sensör ağlarında olduğu gibi birden çok cihaz veri tabanından aynı anda aynı dosyayı indirme sorgusunda bulunabilir. Sorgunun görevi, doğru düğümleri bularak gerekli bilgileri almak ve verileri kullanıcıya geri yönlendirmektir (Boulisa, Hanb, Sheab, & Srivastavab, 2007).

IoT cihazlarının güncellemeleri sırasında birçok sorun meydana gelebilir. Geleneksel veri tabanı ve uygulama güncellemelerindeki bu konular, günümüzde Nesnelerin İnterneti(Internet of Things-IoT) ve akıllı cihazların güncellenmesindeki sorunlar haline gelmiştir. Bunlardan bazıları şunlardır:

- İnternet bağlantısı: IoT cihazları genellikle internete bağlanarak güncelleme alır ancak internet bağlantısı kesintiye uğradığında güncelleme işlemi de aksayabilir. Bu durumda, güncelleme tamamlanamaz veya hatalar oluşabilir.
- Güvenlik: IoT cihazları güvenlik açıkları nedeniyle hedef alınabilirler. Bu açıkların kapatılması için güncellemeler yayınlanır. Ancak, güncelleme işlemi sırasında da bu cihazlar, saldırganların müdahalesiyle karşı karşıya kalabilir. Bu nedenle, güncelleme işlemi sırasında cihazların güvenliği sağlanmalıdır.
- Donanım ve yazılım uyumluluğu: IoT cihazlarının güncellenmesi donanım ve yazılım uyumluluğu sorunlarına neden olabilir. Güncelleme bazı cihazlarda çalışmayabilir veya cihazların performansını olumsuz etkileyebilir.
- Güncelleme boyutu: IoT cihazlarının belleği genellikle sınırlıdır. Bu nedenle güncelleme boyutu büyükse cihazların belleği dolabilir ve güncelleme tamamlanamayabilir.

- Güncelleme süresi: IoT cihazları genellikle sürekli çalışır. Bu nedenle güncelleme işlemi sırasında cihazların çalışmasını durdurmak zor olabilir. Bu da güncelleme süresini uzatabilir veya güncelleme işleminin tamamlanamamasına neden olabilir.
- Yedekleme ve geri dönüş: IoT cihazlarının güncellenmesi, verilerin yedeklenmesi ve geri dönüş işlemleri gibi konularda sorunlara neden olabilir. Bu nedenle güncelleme işlemi öncesinde yedekleme ve geri dönüş işlemleri iyi planlanmalıdır.

Bu sorunlar IoT cihazlarının güncelleme işlemlerinin planlanması ve uygulanması sırasında dikkate alınması gereken konular arasındadır. Bu çalışmada, belirtilen sorunların meydana gelmesini engellemek için alınabilecek önlemler açıklanacak ve bu önlemlere rağmen meydana gelen sorunlarda sistemin tutarlılığının bozulmasının nasıl engelleneceği tartışılacaktır. Önerilecek güncelleme yaklaşımıyla önceki çalışmalara göre çok daha esnek versiyon değişimi ve yeni uygulama aktarımının nasıl yapıldığı test sonuçlarıyla beraber verilecektir.

1.2. Tezin Organizasyonu

Bu çalışma; Kaynak Araştırması (Bölüm 2), Materyal ve Yöntem (Bölüm 3), Çalışma Prensibi ve Gözlem (Bölüm 4), Sonuçlar ve Öneriler (Bölüm 5) olarak bölümlere ayrılmış ve bu çalışmada son olarak gelecekte yapılacak çalışmalar hakkında bilgi verilmiştir.

2. KAYNAK ARAŞTIRMASI

Bu kaynaklar, dağıtık sistemlerde güncelleme süreçlerinin nasıl yönetileceği konusunda farklı yaklaşımlar ile sistemin performansını ve güvenliğini artırmak için öneriler sunmaktadır.

2.1. IoT'ye Yönelik Çalışmalar

IoT alanında güncellenecek veya sınıflandırılacak cihaza nasıl erişileceğinin tanımlanması önemlidir. Uç noktalar üzerinden cihazlara birbiri vasıtasıyla erişilecek ise Lojistik Regresyon (LR), Destek Vektör Makinesi (DVM), k-En yakın Komşu (k-NN), Gradyan Artırma Karar Ağacı gibi pek çok makine öğrenme algoritması kullanılması yöntemlerden biridir (Huang, Li, Yuan, & Li, 2021). Diğer bir yöntem tüm cihazların tek bir sunucu üzerinden güncellenmesidir ve bizim çalışmamızda da bu yöntem kullanılmıştır.

IoT güncelleme işlemlerinde hesaplama maliyetinden tasarruf etmek için karmaşık değerli algoritmalarda kısmi güncelleme yönteminin kullanılması önem arz eder (Qinga, Nia, Lia, & Chenb, 2021). Bu tarz küçük ölçekli ama zamanlaması kritik ve mali karşılığı olan sistemlerde amaç, uygun parametreleri seçerek sürekli yükseltme için tamamlanma süresini, kullanılabilirlik bozulmasını ve parasal maliyeti en aza indirmektir (Sun, Chen, Li, Zhang, & Atıf, 2019). Çalışmamızda tüm uygulamaları bir paket halinde güncellemek yerine parçalı halde sadece versiyon bilgisi değişenlerinin güncellenmesine yönelik bir yaklaşım belirlenmiştir.

Nesnelerin interneti çeşitli cihazların internete bağlanarak veri alışverişi yapmasını ve bu sayede daha akıllı, daha verimli bir dünya yaratılmasını sağlamaktadır. Aşağıda IoT'nin kullanım amaçları ve örnekleri verilmiştir:

a) Cihazlar Arası İletişim: IoT cihazları internet bağlantısı sayesinde birbirleriyle iletişim kurabilirler. Bu sayede cihazlar, arasında veri alışverişi yaparak birbirleriyle çalışabilirler. Örneğin, akıllı bir buzdolabı marketten ürün siparişi verebilir ya da akıllı bir araba trafiği izleyerek rotasını değiştirebilir.

b) Sensörler ve Veri Toplama: IoT cihazları çeşitli sensörlerle donatılarak çevrelerindeki değişiklikleri algılayabilirler. Bu sensörler sıcaklık, nem, hareket, basınç,

ışık, gaz, toz ve daha birçok ölçümü yapabilirler. Bu veriler daha sonra analiz edilerek farklı alanlarda kullanılabilir. Örneğin; akıllı tarım sistemi, toprak nemini ölçerek sulama sistemini otomatik olarak kontrol edebilir.

c) Veri Analizi ve Yönetimi: IoT cihazlarından toplanan veriler bulut tabanlı sistemlerde depolanarak analiz edilebilir. Örneğin; bir akıllı şehir planlaması uygulaması, trafik yoğunluğu verileri kullanarak trafik akışını düzenleyebilir.

d) Güvenlik: IoT cihazlarının internete bağlı olması onları siber saldırılara açık hale getirir. Bu nedenle güvenlik sağlama yöntemleri dikkatle kullanılmalıdır. Örneğin; akıllı ev sistemi, ev sahibinin kimlik doğrulamasını sağlayarak sadece onun erişimine izin verir.

e) Uygulama Alanları: IoT teknolojisi birçok alanda kullanılabilmektedir. Örneğin; endüstriyel üretimde fabrika makineleri IoT cihazlarıyla donatılarak üretim süreçleri daha verimli hale getirilebilir. Sağlık sektöründe, akıllı cihazlar hastaların sağlık durumunu izleyerek doktorlara veri sağlayabilir. Ev otomasyonunda, akıllı ev sistemleri ev sahiplerinin evlerini uzaktan kontrol etmelerine olanak tanır.

Bu bağlamda bakıldığında IoT teknolojisi günümüzde hızla gelişmekte olup IoT teknolojisinin gelecekte daha birçok alanda kullanılabileceği öngörülmektedir.

Ala Al-Fuqaha'nın arkadaşlarıyla yaptığı IoT teknolojisi, IoT cihazları, IoT protokolleri ve IoT uygulama alanları hakkında ayrıntılı bir inceleme çalışması bu alanda yapılmış kapsamlı derlemelerdendir. Bu çalışmada ilk olarak, IoT teknolojisi ve IoT cihazları incelenmiş IoT cihazlarının özellikleri, sensörleri, aktüatörleri, donanım ve yazılım gereksinimleri, enerji yönetimi, güvenlik ve gizlilik gibi konular ele alınmıştır. Sonraki aşamada IoT protokolleri ve teknolojileri detaylı bir şekilde incelenmiş; kablosuz ağlar, sensör ağları, RFID, NFC, Bluetooth, Wi-Fi, ZigBee, Z-Wave, LoRaWAN, Sigfox, MQTT, CoAP, HTTP, DDS gibi protokollerin avantajları, dezavantajları ve kullanım alanları ele alınmıştır. Son olarak, IoT uygulama alanları incelenmiş; akıllı evler, akıllı şehirler, akıllı sağlık, akıllı tarım, akıllı üretim ve akıllı ulaşım gibi farklı alanlarda IoT teknolojisinin kullanımı ele alınmıştır. Ayrıca, bu çalışmada IoT teknolojisinin sunduğu avantajlar, dezavantajlar, zorluklar ve gelecekteki potansiyel etkileri de tartışılmıştır. Bu çalışmada Ala Al-Fuqaha ve arkadaşları IoT ile büyük veri analitiği ve bulut bilgi işleme gibi alanlar başta olmak üzere diğer gelişmekte olan teknolojiler arasındaki artan ilişkiyi anlatmışlardır (Al-Fuqaha, Guizani, Mohammadi, Aledhari, Ayyash, 2015).

2.1.1. IoT Güncellemelerinin Gelişim Süreci

IoT güncellemelerinin gelişim süreci, genel olarak şu adımlardan oluşur:

İlk Sürüm ve Güncellemelerin Belirlenmesi: IoT cihazının ilk sürümü geliştirildikten sonra kullanıcı geri bildirimleri, pazar talepleri ve teknolojik gelişmeler göz önünde bulundurularak güncellemelerin belirlenmesi süreci başlar. Bu süreçte, kullanıcıların talepleri ve ihtiyaçları, cihazın eksiklikleri ve potansiyel güvenlik açıkları dikkate alınır.

Yazılım Güncellemeleri: IoT cihazları genellikle yazılım tabanlıdır ve güncellemeler yazılım yoluyla sağlanır. Bu aşamada, geliştiriciler yeni özellikler, hata düzeltmeleri veya performans iyileştirmeleri gibi güncellemeleri hazırlar ve test eder. Ardından güncellemeler kullanıcılara yayılmak üzere cihazlara dağıtılır.

Firmware Güncellemeleri: IoT cihazlarının bazıları donanım üzerinde çalışan firmware adı verilen bir yazılım kullanır. Firmware güncellemeleri, cihazın donanım bileşenlerinin çalışmasını etkileyen değişiklikleri içerir. Bu güncellemeler genellikle üreticinin sunucuları üzerinden cihazlara yayılır veya kullanıcılar tarafından elle yüklenir.

Güvenlik Güncellemeleri: IoT cihazlarının güvenliği büyük bir öneme sahiptir. Güvenlik açıkları keşfedildiğinde, üretici güncellemeleri yayınlar ve bu açıkları kapatır. Güncellemeler, kullanıcı verilerinin korunmasını sağlamak, saldırılara karşı savunma sağlamak veya güvenlik politikalarını güncellemek amacıyla dağıtılabilir.

Uzaktan Yönetim ve Dağıtım: IoT cihazlarının birçoğu uzaktan yönetilebilir ve güncellenebilir. Üretici, güncellemeleri otomatik olarak cihazlara dağıtabilir veya kullanıcılara manuel olarak yüklemeleri için bildirimler gönderebilir. Uzaktan yönetim ve dağıtım, kullanıcılar için kolaylık sağlar ve güncellemelerin yayılmasını hızlandırır.

Kullanıcı Geri Bildirimi ve İyileştirme: Güncellemelerin dağıtımı ve kullanımı sonrasında kullanıcı geri bildirimleri toplanır. Bu geri bildirimler, yeni güncellemelerin planlanması ve gelecekteki sürümlerin geliştirilmesi için kullanılır. Kullanıcı geri bildirimleri, hataların düzeltilmesi, yeni özelliklerin eklenmesi veya kullanıcı deneyiminin geliştirilmesi gibi iyileştirmelerin belirlenmesine yardımcı olur. Bu geri bildirimler, üreticiye yol gösterici bilgiler sunar ve IoT cihazlarının gelecekteki güncellemeleri için temel oluşturur.

Güncellemelerin gelişim süreci, bu adımların tekrarlanmasını içermektedir. Bir güncelleme yayınlandıktan sonra yeni geri bildirimler ve gereksinimler ortaya çıkabilmekte ve bu da yeni bir güncelleme sürecini tetikleyebilmektedir. Bu şekilde, IoT

cihazları sürekli olarak güncellenmekte, kullanıcı ihtiyaçlarına ve teknolojik gelişmelere uyum sağlamaktadır.

Özetlemek gerekirse, IoT güncellemelerinin gelişim süreci; güncellemelerin belirlenmesi, yazılım ve firmware güncellemeleri, güvenlik güncellemeleri, uzaktan yönetim ve dağıtım, kullanıcı geri bildirimi ve iyileştirme adımlarını içermektedir. Bu süreç, IoT cihazlarının performansını, güvenliğini ve kullanıcı deneyimini sürekli olarak geliştirmeyi amaçlamaktadır.

Sun arkadaşlarıyla beraber yaptığı çalışmada dağıtık yazılım sistemlerindeki güncelleme süreçlerini optimize etmek için bir çözüm sunmaktadır. Çalışmada dağıtık yazılım sistemlerinin güncelleme süreçlerindeki performans, güvenilirlik ve maliyet unsurlarını ele almak üzere üç farklı hedef fonksiyonu tanımlanmaktadır. Bu hedef fonksiyonları, yazılım güncellemelerinin otomatikleştirilmesi için optimizasyon problemi oluşturmaktadır. Çalışma; çözümün ölçeklenebilirliği, etkinliği ve gerçek zamanlı güncelleme özellikleri üzerinde durarak önerilen çözümün diğer yöntemlere göre avantajlarını tartışmaktadır. Önerilen çözümün özelliklerinin yanı sıra, bu çözümün uygulanmasında kullanılan teknolojiler de çalışmada açıklamaktadır. Bu teknolojiler arasında, bulut bilişim hizmetleri, dağıtık yazılım sistemleri, yazılım güncelleme yönetimi, otomatikleştirme araçları ve optimizasyon algoritmaları yer almaktadır. Ek olarak bu çalışma, önerilen çözümün gerçekleştirilmesinde kullanılan bir dizi deneysel sonucu da sunmaktadır. Bu sonuçlar, önerilen çözümün diğer yöntemlere göre daha verimli performans gösterdiğini ve daha etkili bir yazılım güncelleme süreci sağladığını göstermektedir. Sonuç olarak, bulut bilişim ortamlarında yazılım güncellemelerinin otomatikleştirilmesi ve DevOps prensiplerinin uygulanması için önemli bir adım olarak gördükleri çalışmalarında Sun ve arkadaşları “küçük ölçekli ama zamanlama kritik ve mali karşılığı olan sistemlerde amaç, uygun parametreleri seçerek sürekli yükseltme için tamamlanma süresini, kullanılabilirlik bozulmasını ve parasal maliyeti en aza indirmektir” çıkarımında bulunmuşlardır (Sun, Chen, Li, Zhang, & Atıf, 2019).

Çok katmanlı kurumsal sistemler, genellikle, dağıtılmış sistem boyunca yuvarlanan bir dalga halinde her bir düğümü bir seferde yükselten ve yeniden başlatan sürekli yükseltmeler kullanmaktadır. Dumitraş'ın çalışması, online yazılım güncellemelerinin çoklu yönetim alanları üzerindeki etkilerini incelemektedir. Çalışmada farklı yönetim alanlarının yazılım sistemlerinin farklı özellikleri, politikaları ve güncelleme süreçleri olduğu göz önünde bulundurularak online güncellemelerin, yönetim alanları arasındaki ilişkileri nasıl etkilediği araştırılmıştır. Çalışma gerçek dünya

senaryoları ve simülasyonlar kullanılarak yapılmıştır. Bu senaryolardan bazıları bir şirketin bir uygulamasının güncellenmesi, bir hizmet sağlayıcının hizmet sunucusunun güncellenmesi veya bir kurumun kendi özel ağındaki bir yazılımın güncellenmesi gibi çeşitli senaryolardır. Çalışmanın sonuçları, online güncellemelerin etkilerinin yönetim alanlarının özelliklerine, güncelleme süreçlerine ve politikalarına bağlı olarak farklılık gösterdiğini ortaya koymuştur (Dumitraş, Narasimhan, & Tilevich, 2010).

Güncelleme sisteminin gerekli uygulama bünyesine entegre bir yapı olmasından ziyade ayrı bir uygulama olarak tasarlanması sayesinde altyapı, sistem kaynak koduna erişim gerektirmez ve şeffaftır: düğüm yazılımı, farklı sürümlerin varlığından bile habersizdir. Ajmani'nin çalışması dağıtık sistemlerin otomatik yazılım güncellemelerinin nasıl yapılacağına dair bir yöntem sunmaktadır. Bu çalışma, dağıtık sistemlerdeki yazılım güncellemelerinin yönetimini kolaylaştırmak ve hataları önlemek için bir dizi teknik ve araçlar önermekte, dağıtık sistemlerdeki yazılım güncellemelerinin yönetiminde ortaya çıkan sorunları ve bu sorunların üstesinden gelmek için kullanılabilecek teknikleri ele almakta, yazılım güncellemelerinin otomatik olarak yapılması için bir algoritma önerilmektedir. Bu algoritma, dağıtık sistemlerin özelliklerini ve yazılım güncellemelerinin risk seviyelerini dikkate almakta, yazılım güncellemelerinin otomatik olarak yapılması için kullanılabilecek farklı araçlar hakkında bilgi vermekte ve dağıtık sistemlerde yazılım güncellemelerinin yönetimi için bir dizi uygulama sunmaktadır (Ajmani, 2004).

2.1.2. IoT Güncelleme Önerileri

Dağıtık sistemler çalışma prensipleri gereğince birçok alt destek donanımlarına ihtiyaç duymakta ve bu da donanım fazlalığının oluşmasına neden olmaktadır. 1980 başlarına doğru bu donanım fazlalığını avantaja çevirmek için farklı çalışmalar yapılmıştır. Örnek olarak Boloom'un çalışması, dağıtık programlama sistemlerinde dinamik modül değiştirme yöntemlerini ele almaktadır ve bunu yapanbieln bir çerçeve oluşturmak için tasarlanmıştır. Bu çerçeve yazılım güncellemeleri, yamalar ve hatalı modüllerin yenisiyle değiştirilmesi gibi işlemleri yapmak için kullanılabilmektedir. Çalışma iki temel bileşenden oluşmaktadır: modül yükleyicisi ve modül yöneticisi. Modül yükleyicisi dağıtık programlama sistemine yeni bir modül eklemek için kullanılmaktayken, modül yöneticisi ise var olan bir modülün güncellenmesi veya değiştirilmesi için kullanılmaktadır. Bu iki bileşen birbirleriyle etkileşim halindedir ve dağıtık programlama sistemi üzerinde dinamik modül değiştirme işlemini

gerçekleştirmek için birlikte çalışmaktadırlar. Çalışmada özellikle modül değiştirme işleminin mümkün olduğunca kesintisiz gerçekleşmesi üzerinde durulmuştur. Bu amaçla modül yükleyicisi ve modül yöneticisi, modülün önce yüklenip yüklenmediğini, daha önceki bir modülün yerine geçip geçmeyeceğini ve kullanım sırasında herhangi bir kesintiye neden olup olmayacağını kontrol etmektedir. Böylece, modül değiştirme işlemi mümkün olduğunca sorunsuz bir şekilde gerçekleştirilmektedir. Sonuç olarak çalışmada dağıtık programlama sistemlerinde yazılım güncellemeleri, yamalar ve hatalı modüllerin yenisiyle değiştirilmesi gibi işlemleri yapabilen bir çerçeve sunulmaktadır. Bu sayede, dağıtık sistemlerin daha güvenli ve daha güncel bir şekilde çalışması sağlanabilir (Bloom, 2001).

Bloom ve Day'in çalışması, dağıtık sistemlerde esneklik sağlamak için otomatik yeniden yapılandırma tekniklerine odaklanmaktadır. Bu teknikler sistemdeki bir bileşenin başarısız olduğu veya yenilendiği durumlarda, sistemdeki diğer bileşenlerin uyumlu bir şekilde yeniden yapılandırılmasını sağlamaktadır. Çalışma Argus adlı bir dağıtık programlama sistemine odaklanmaktadır. Argus birçok bilgisayarın birbirine bağlanarak dağıtık bir sistem oluşturmasına olanak tanıyan bir programlama dilidir. Argus'un yeniden yapılandırılabilir bir sisteme sahip olmasını sağlamak için bazı teknikler önerilmiştir. Önerilen teknikler arasında Argus sistem bileşenlerinin birbirleriyle iletişim kurabilmesi için gerekli olan adreslerin otomatik olarak yeniden yapılandırılması ve Argus sistem bileşenlerinin başarısızlık durumunda otomatik olarak yeniden başlatılması yer almaktadır. Bu teknikleri Argus üzerinde uygulayarak, Argus sistemlerinin daha esnek ve güvenilir hale getirilmesine katkı sağlayabileceğini göstermiştir. Bu çalışma, dağıtık sistemlerin yeniden yapılandırılmasına odaklanan diğer çalışmalar için de önemli bir referans kaynağıdır (Bloom & Day, 1992).

Segal ve Frieder'in çalışması yazılım sistemlerindeki değişikliklerin canlı ortamda yapılmasına olanak tanıyan bir teknik olan dinamik program güncelleme (DPU) konusuna odaklanmaktadır. Çalışma yazılım sistemlerinin canlı ortamda sürekli olarak çalışması gerektiği için geleneksel yazılım güncelleme yöntemlerinin yetersiz kaldığı durumlarda DPU'nun sunduğu avantajlara odaklanmaktadır. DPU yöntemi yazılımın kullanıcılarına kesintisiz bir hizmet sunmakta, sistemleri kapatmadan veya kullanıcıları kesintiye uğratmadan yazılımı güncelleme veya hata düzeltme imkanı sağlamaktadır . Çalışmada yazılım geliştirme ve sürdürme sürecinde önemli bir etkiye sahip olan DPU konusunda önemli bir araştırma yapılmıştır (Segal & Frieder, 1989).

Ophir ve Segal'in çalışması, yazılım programlarının dinamik olarak güncellenmesi konusuna odaklanmaktadır. Bu çalışmada yazılım programlarının hatalarını düzeltmek veya yeni özellikler eklemek için güncellemeler yapılması gerektiği gerçeğinden yola çıkılmaktadır. Ancak bu güncellemelerin gerçekleştirilmesi sırasında programların durdurulması veya kullanılamaz hale gelmesi gibi sorunlar ortaya çıkabilir. Bu çalışmada programların dinamik olarak güncellenmesi için bir yöntem önerilmektedir. Bu yöntem programın güncellenmiş sürümünün bir kopyasının oluşturulması ve güncelleme işleminin bu kopyada gerçekleştirilmesi üzerine kuruludur. Eski sürüm hala kullanımda olduğu için kullanıcıların programının durması veya programın hatalarından dolayı kullanılamaz hale gelmesi riski ortadan kalkmaktadır. Güncellenmiş sürümdeki hataların olası etkileri de bu şekilde önlenmiş olmaktadır. Çalışmada bu yöntemin gerçekleştirilmesi için bir prototip yazılım geliştirilmiştir. Bu prototip bir uygulama yazılımı için güncelleme işleminin nasıl gerçekleştirilebileceğini göstermektedir. Yapılan deneylerde, bu yöntemin program güncelleme işleminin başarılı bir şekilde gerçekleştirilmesine olanak tanıdığı ve programın durdurulmasına veya kullanılamaz hale gelmesine neden olmadığı gösterilmiştir. Çalışma yazılım programlarının güncellenmesi sırasında ortaya çıkabilecek sorunların üstesinden gelmek için yeni bir yöntem sunmaktadır. Bu yöntem programların dinamik olarak güncellenmesini mümkün kılarak, kullanıcılarının programları durdurulmadan veya programların hatalarından dolayı kullanılamaz hale gelmesini önleyerek, programların daha güvenli ve daha etkili bir şekilde güncellenmesine olanak sağlamaktadır (Ophir & Segal, 1991).

Magee ve arkadaşlarının çalışması, bir dağıtık sistem tasarlama ve geliştirme yöntemi olan Conic'in ayrıntılı bir tanıtımını sunmaktadır. Conic yöntemi sistem tasarımı için modüler bir yaklaşım sunmakta ve sistem modülleri arasındaki bağlantıları tanımlamak için bir dizi formalizm kullanmaktadır. Çalışmada Conic'in özellikle büyük ölçekli sistemlerin tasarımında ve geliştirilmesinde faydalı olduğu üzerinde durulmuş ve Conic ile dağıtık sistemlerin tasarımı ve geliştirilmesi için kullanılabilecek temel adımlar ayrıntılı bir şekilde açıklanmıştır. Bu adımlar arasında sistem gereksinimlerinin belirlenmesi, sistem mimarisinin tasarlanması, modüllerin belirlenmesi, sistem modüllerinin tasarlanması, modüller arası iletişimin tanımlanması ve sistem testlerinin gerçekleştirilmesi yer almaktadır.

Megee ve arkadaşları çalışmasında Conic'in kullanımıyla ilgili bir dizi örnek vermektedir. Bu örnekler dağıtık sistemlerin tasarımı ve geliştirilmesinde Conic'in nasıl kullanılabileceğini göstermektedir. Örnekler arasında bir uçuş rezervasyon sistemi, bir

şube ofisi ağı, bir emlak listeleme sitesi ve bir veri merkezi yer almaktadır. Sonuç olarak Magee ve arkadaşlarının çalışması dağıtık sistemlerin tasarımı ve geliştirilmesi için Conic yaklaşımının kullanımını ayrıntılı bir şekilde açıklamıştır. Bu çalışma Conic'in modüler ve formal yaklaşımı sayesinde büyük ölçekli sistemlerin tasarımında faydalı olduğunu göstermektedir (Magee, Kramer, & Sloman, 1989).

Hyeong ve arkadaşları bir oyun kimin için güncellenir sorusuna cevap aramış, aynı zamanda çalışan bir dağıtık mimarinin tüm düğümlerinin ihtiyaca göre güncellenmesi ve güncellenme anının cihazın çalışma anının dışında olması gerektiği değerlendirmesini yapmıştır (Hyeaong, Choi, Lee, & Pyo, 2020).

Priyatham, çalışmasında, elektrikli araçların giderek artan talebi ve bunların gerektirdiği otomotiv elektronik kontrol üniteleri (ECU) konularını ele almaktadır. Çalışmada geleneksel içten yanmalı motorlu araçlara kıyasla elektrikli araçların daha az hareketli parçaya ve daha fazla otomotiv elektronik kontrol ünitesine ihtiyaç duyduğu belirtilmektedir.

Elektrikli araçlarda batarya yönetimi, elektrik motoru kontrolü, şarj cihazı yönetimi ve diğer birçok işlev için bir dizi kontrol ünitesi gerekmektedir. Bu da elektrikli araçların üretim maliyetlerini artırmakta ve otomotiv sektöründe yeni fırsatlar yaratmaktadır. Priyatham'ın çalışmasında elektrikli araçlardaki ECU'ların güvenilirliği, yazılım güncellemeleri, veri paylaşımı ve siber güvenlik gibi konuları da ele alınmaktadır. Bu faktörler, gelecekteki elektrikli araçların geliştirilmesi ve kullanımının başarısı için önemli bir rol oynamaktadır. Sonuç olarak çalışma elektrikli araçların giderek artan talebi ve bunların otomotiv sektörüne yaptığı etkiyi ele almakta ve bu konuda önemli bir bilgi kaynağı sağlamaktadır (Priyatham, 2019).

2.2. Benzin İstasyonları

Benzin istasyonlarında kullanılan benzin dispenserleri, yakıtın aracın deposuna aktarılmasını sağlayan cihazlardır. Benzin dispenserleri, ayrı bir gövdeye sahip bir pompa, bir ölçer ve bir vanadan oluşmaktadır.

Bir müşteri pompayı çalıştırdığında, pompa harekete geçmekte ve benzin depodan dispenser üzerine doğru akmaktadır. Akış hızı, akış hızı ölçer tarafından ölçülmekte ve pompayı durdurmak için bir kontrol mekanizması tarafından izlenmektedir. Yakıtın istenen miktarı verildiğinde, vanalar kapanmakta ve müşteri ödeme yapmak üzere kasaya yönlendirilmektedir.

Benzin dispenserleri, hem manuel hem de otomatik olarak çalışabilen farklı türlerde mevcuttur. Bazı dispenserler aynı zamanda alternatif yakıtların dağıtımı için de tasarlanmıştır. Dispenserlerin bakımı ve kalibrasyonu düzenli olarak yapılıp ve sürekli kontrol altında tutulmaktadır.

2.3. Dijital Yakıt Tabancası

Dijital yakıt tabancası benzin istasyonlarında kullanılan bir tür otomatik tabancadır. Geleneksel mekanik tabancaların yerini almıştır ve daha doğru, güvenli bir yakıt dağıtımı sağlamaktadır.

Dijital yakıt tabancaları, elektronik kontrol ünitelerini kullanarak yakıtın akışını izlemekte ve yönetmektedir. Bu sayede yakıtın doğru miktarının verilmesini ve hatalı yakıt dağıtımının önlenmesini sağlamaktadır. Ayrıca dijital yakıt tabancaları, sürücünün istediği miktarı seçebileceği dokunmatik ekran gibi kullanıcı dostu özellikler sunmaktadır.

Dijital yakıt tabancalarının avantajları arasında daha hızlı ve verimli yakıt dağıtımı, daha doğru yakıt ölçümü ve hatalı yakıt dağıtımının önlenmesi sayılabilir. Ayrıca, yakıt tabancasının dijital olması, işletme sahiplerinin işletmelerini daha verimli bir şekilde yönetmelerine yardımcı olmaktadır. Örnek olarak, Gilbarco Veeder-Root, Tokheim, Mepsan Petrol Cihazları ve Wayne Fueling Systems gibi şirketler, dijital yakıt tabancası üretmektedir.

Salvi ve arkadaşlarının çalışması, elektrikli araçların elektronik kontrol üniteleri için yeni bir tasarım mimarisi sunmaktadır. Bu çalışma otomotiv endüstrisindeki elektrikli araçların artan popülaritesi nedeniyle önem kazanmaktadır. Bu çalışmada, FPGA ve DSP gibi yeni teknolojiler kullanarak elektrikli araçların performansını artırmak için geleneksel ECU'lardan daha hızlı ve daha güçlü bir sistem önerilmektedir. FPGA ve DSP; elektrikli araçların batarya yönetimi, motor kontrolü, şarj yönetimi ve diğer işlevleri gibi önemli görevlerde kullanılabilir. Ayrıca bu çalışma enerji verimliliği ve tasarrufu için daha fazla optimize edilmiş bir ECU tasarımı önermektedir. FPGA ve DSP teknolojilerinin birleştirilmesi, daha fazla verimlilik sağlamakta ve batarya ömrünü uzatmaktadır. Çalışma da elektrikli araçların daha hızlı ve güçlü ECU'lara sahip olması, daha iyi performans ve daha fazla enerji verimliliği sağlayacağı sonucuna varılmıştır (Salvi, Mitra, Devlekar, Choudhary, Patil & Bhosale, 2022).

2.4. Otonom Araçlar

Otonom araçlar sürücü müdahalesi olmadan kendi kendine seyahat edebilen araçlardır. Bu araçlar, bir dizi teknoloji kullanarak çevrelerini algılayabilmekte, karar verebilmekte ve yolu takip edebilmektedir. Otonom araçlar; yaşlılık, yaralanma, engellilik veya alkollü sürüş gibi sürücülerin çeşitli nedenlerden dolayı araç kullanamayacakları durumlarda büyük bir potansiyel sunmaktadır. Bununla birlikte otonom araçların yaygınlaşması bir dizi teknolojik, yasal ve etik sorunları da beraberinde getirmiştir.

Otonom araçlar çeşitli seviyelerde otonom sürüş yeteneğine sahiptir. SAE International tarafından belirlenmiş beş seviye sınırlandırması aşağıdaki gibidir:

- Seviye 0: Araç tamamen insan kontrolünde çalışır.
- Seviye 1: Araç, sürücü kontrolünde ancak bazı yardımcı teknolojilerle desteklenir. (örneğin, kör nokta uyarıları)
- Seviye 2: Araç, önceden belirlenmiş koşullarda kısmen otonom sürüş yapabilir. Sürücü, aracın kontrolünü her zaman elinde tutar.
- Seviye 3: Araç, belli bir koşulda tam otonom sürüş yapabilir. Ancak, sürücü, sistemin uygun koşullar altında kontrolü ele alabileceği şekilde her zaman hazırda beklemelidir.
- Seviye 4: Araç, belirli koşullarda tamamen otonom olarak sürüş yapabilir. Sürücü müdahalesi gerektirmeyen koşullarda çalışabilir.
- Seviye 5: Araç, tüm koşullarda tamamen otonom olarak sürüş yapabilir. İnsan müdahalesi gerektirmez.

Otonom araçların yaygınlaşması trafik güvenliği, trafik sıkışıklığı, çevre kirliliği, ulaşım maliyetleri ve ulaşılabilirlik açısından önemli potansiyeller sunmaktadır ancak otonom araçların yasal, etik, güvenlik ve mahremiyet sorunları da hala tartışma konusu olmaya devam etmektedir.

Shengbo Eben Li ve arkadaşlarının çalışması platoon (konvoy) kontrolleri için ağ tabanlı bir kontrol perspektifini ele alarak, bileşen modellemesi ve kontrolör sentezi konularını ele almaktadır. Platoon kontrolü bir grup aracın birbirine yakın bir şekilde seyahat ettiği durumlarda araçların hızını ve konumunu optimize etmek için kullanılan bir kontrol stratejisidir. Bu yakın seyahat, araçların yakıt tasarrufu yapmasını ve trafik akışının daha verimli hale gelmesini sağlamaktadır. Çalışmada platoon bileşenleri olan araçlar, mesafe ölçümleri ve iletişim bileşenleri ile matematiksel olarak modellenmekte, bu bileşenlerden platoon modeli oluşturulmaktadır. Daha sonra platoon kontrolörü

tasarımı ele alınmaktadır. Birinci adımda platoon hızının ve konumunun optimizasyonu için kontrol hedefi belirlenmektedir. İkinci adımda referans sinyal üreten lider araç ve diğer araçların takip ettiği strateji geliştirilmektedir. Üçüncü adımda, lider araçtan gelen referans sinyaline göre, her araç için bir kontrol yolu tasarlanmaktadır. Bir platoon kontrol sistemi otomatik olarak seyahat eden bir grup kamyon veya otomobil gibi bir araç grubunda kullanılabilir. Bu araçlar platoon bileşenleri tarafından sağlanan mesafe ölçümleri ve iletişim bileşenleri aracılığıyla birbirleriyle bağlantı kurabilmektedir. Bu bağlantı lider araçtan gelen referans sinyaline göre her aracın hızını ve konumunu kontrol edebilmektedir. Bu sayede araçlar yakıt tasarrufu yaparak daha verimli bir trafik akışı sağlayabilmektedir (Li, Zheng, Li, Wang & Zhang, 2017).

2.5. Ağlar

Güncelleme işlemi için kullanılabilecek ağlar ve bu ağlar kullanılarak üzerine yeni projeler geliştirilebilecek sistemler şu şekildedir:

2.5.1. Hücresel Ağlar

Hücresel ağlar, kablosuz iletişim teknolojisi kullanarak mobil cihazlar arasında veri iletimini sağlayan bir telekomünikasyon ağı türüdür. Bu ağlar özellikle cep telefonları, tabletler, dizüstü bilgisayarlar ve akıllı saatler gibi mobil cihazlarla bağlantı kurmak için kullanılmaktadır.

Hücresel ağlar, dünya genelinde birçok telekomünikasyon şirketi tarafından yönetilmekte olup bu ağların iletişim altyapısı GSM, CDMA, UMTS, LTE ve 5G gibi standartlarla belirlenmiştir. Her hücresel ağ bir dizi hücreden oluşmakta ve her hücre, antenler ve baz istasyonları kullanılarak iletişim sağlamaktadır.

Bu ağlar sesli arama, metin mesajı, veri indirme/yükleme, video akışı ve mobil uygulamalar gibi birçok farklı işlem için kullanılabilmektedir. Hücresel ağlar, mobil cihazlar arasındaki iletişimi sağlamak için önemli bir rol oynamakta olup günümüzde birçok insanın günlük yaşamında vazgeçilmez bir araç haline gelmiştir. Bununla beraber bu geniş kullanım çerçevesinde hücresel ağ optimizasyonu gibi konular ortaya çıkmış ve konuya ilişkin Mishra çalışmasında farklı yöntemleri ele almıştır. Bu yöntemler arasında, istatistiksel analiz, ölçümlere dayalı optimizasyon, modelleme ve simülasyon gibi teknikler yer almaktadır (Mishra, 2007).

2.5.2. Kablosuz Ağlar

Kablosuz ağlar, veri iletimi için kablolar yerine elektromanyetik dalgaları kullanarak cihazlar arasında iletişim kurmak için kullanılan bir iletişim teknolojisidir. Kablosuz ağlar Wi-Fi, Bluetooth ve mobil veri ağları gibi farklı türlerde olabilmektedir. Kablosuz ağlar birçok avantaj sunmakta olup öncelikle kablolara ihtiyaç duymadıkları için esnek ve taşınabilirlerdir ki bu da ilgili cihazların birbirine yakın olması gerekmeden veri aktarımı yapabilecekleri anlamına gelmektedir. Ayrıca bu ağlar geniş alanlara yayılabileceği için geniş kapsama alanlarına sahip olabilmektedir.

Bununla birlikte kablosuz ağların bazı dezavantajları da mevcuttur. Kablosuz ağların güvenliği kablolara göre daha düşük olabilmektedir ev bu nedenle hassas verilerin iletilmesi sırasında dikkatli olunmalıdır. Ayrıca kablosuz ağlar, sinyal kalitesi ve hızı gibi faktörlere bağlı olarak kablolara göre daha yavaş ve dengesiz olabilir.

Kablosuz ağların yaygın kullanımı bugünün dünyasında birçok farklı sektörde önemli bir rol oynamaktadır. Bu sektörlere örnek olarak evler, iş yerleri, okullar, hastaneler ve havaalanları verilebilir.

IoT ve kablosuz ağ etkileşimi de her gün artmakta olup bununla ilgili olarak Gupta ve arkadaşları IoT teknolojisinin kullanıldığı bir ev otomasyon sistemi tasarlamıştır. Sistem evdeki elektrikli cihazların izlenmesini ve kontrol edilmesini sağlamaktadır. Sensörler ve akıllı cihazlar aracılığıyla evdeki cihazların durumu izlenebilmekte ve uzaktan kontrol edilebilmektedir. Çalışmanın amacı evlerdeki enerji tüketimini izlemek ve optimize etmek için IoT teknolojisi kullanarak enerji tasarrufu sağlamaktır. Ayrıca sistem kullanıcılara evdeki cihazların durumu hakkında geri bildirim sağlayarak bu cihazların daha etkili kullanımını sağlamaktadır (Gupta & Johari, 2019).

2.5.3. Güvenilir Olmayan Ağlar

Güvenilir olmayan ağlar kullanıcılara ve verilere zarar verebilecek ağlardır. Bu tür ağlar genellikle ağ yöneticileri tarafından hem kontrol edilmemekte hem de yönetilmemektedir ve kötü niyetli kişiler tarafından kullanılmaktadırlar. Güvenilir olmayan ağlara örnekler aşağıda verilmiştir:

a) Açık kablosuz ağlar: Halka açık olan ve herhangi bir kimlik doğrulaması yapılmadan herkesin bağlanabileceği kablosuz ağlardır. Bu tür ağlar kullanıcıların verilerini izinsiz olarak izleyen veya çalan kötü niyetli kişiler tarafından kullanılabilmektedir.

b) İnternet kafeler: Özellikle seyahat edenler veya evde interneti olmayan kişiler arasında yaygındır ancak bu tür yerlerdeki bilgisayarlar güvenli olmayabilirler ve kötü amaçlı yazılımlarla enfekte olabilirler.

c) Pazar yerleri: İnternet üzerindeki pazar yerleri alışveriş yapmak için yaygın olarak kullanılmaktadır ancak güvenilir olmayan pazar yerleri kullanıcıların kişisel ve finansal bilgilerini çalabilmekte veya kullanıcıları dolandırabilmektedir.

d) Tor ağı: İnternete gizlice erişmek için kullanılan bir ağdır. Ancak bu ağ; dolandırıcılık, kimlik hırsızlığı ve diğer yasa dışı faaliyetler gibi kötü amaçlara da zemin hazırlayabilmektedir.

Güvenilir olmayan ağlar kullanıcılara ve verilere zarar verme riski taşıdıkları için mümkün olduğunca kullanılmamalıdır. Kullanıcıların güvenliği ve verilerin korunması için güvenilir ağlara bağlanmak ve her zaman güvenlik yazılımlarını güncellemek önemlidir.

F. Tang ve arkadaşları yeni nesil network ve bunun güvenliği konusunda 6G ağları ve metaverse (sanal evren) kavramlarını bir araya getirerek bu iki alanın birleşmesinin nasıl gerçekleşebileceğine dair bir yol haritası sunmaktadır. Çalışmada 6G teknolojisinin metaverse uygulamalarında kullanılmasının getireceği avantajlar ve mevcut sorunlar ele alınmaktadır. 6G teknolojisi ile metaverse uygulamalarının birleştirilmesi sonucunda yüksek hızlı veri transferi, düşük gecikme süresi, yüksek güvenlik ve ölçeklenebilirlik sağlanabileceği belirtilmektedir. Çalışmada 6G ağları ile metaverse arasındaki etkileşimin nasıl olacağına ve bu etkileşimin hangi alanlarda kullanılabileceğine dair örnekler verilmektedir (Tang, Chen, Zhao & Kato, 2022).

2.5.4. Araç Ağları

Araç ağları, otomotiv endüstrisindeki elektronik donanımların birbirleriyle ve dış dünyayla iletişim kurabildiği ağlardır. Bu ağlar otomobildeki sensörlerden araç içi eğlence sistemlerine, akıllı navigasyon sistemlerine ve hatta akıllı sürüş yardımcılarına kadar birçok bileşenin birbiriyle haberleşmesini sağlamaktadır.

Araç ağları, araçlarda bulunan elektronik donanım bileşenlerinin birbirleriyle bağlanması ve iletişim kurması için tasarlanmıştır. Bu bileşenler arasında motor kontrol üniteleri, araç içi eğlence sistemleri, klima kontrol üniteleri, fren sistemi sensörleri ve güvenlik sistemleri gibi çeşitli donanımlar yer almaktadır. Otomobilin içindeki donanımların birbirleriyle haberleşmesini sağlamanın yanı sıra, otomobilin dışındaki sensörlerle de iletişim kurabilmektedir. Örneğin bir otomobilin park sensörleri aracın park edildiği

alanın yakınlarında bulunan nesnelere yakınlığını ölçerek sürücüye uyarı mesajları göndermektedir. Benzer şekilde otomatik acil fren sistemleri araçtaki sensörlerin yol koşullarını ölçerek aracın durmasını sağlamaktadır.

Araç ağları; yakıt verimliliğini artırmak ve araçlarda daha iyi kontrol sağlamak gibi birçok fayda sağlamakta; araçlar arasında iletişim kurabilen akıllı araç teknolojilerinin geliştirilmesinde de trafik sıkışıklığını azaltmak, kazaları önlemek, ve sürüş güvenliğini artırmak gibi önemli bir rolü bulunmaktadır. Örneğin; Tesla'nın araçları internete bağlanarak yazılım güncellemeleri alabilmekte olup araç içi eğlence sistemleri, güvenlik sistemleri ve sürüş yardımcıları gibi birçok bileşeni birbirine bağlayan bir araç ağına sahiptir. Benzer şekilde General Motors, Ford ve diğer otomobil üreticileri de araç ağlarına dayalı akıllı araç teknolojileri geliştirmektedirler.

Thanh Nam Pham ve arkadaşları çalışmalarında akıllı park sistemi tasarımını ve uygulamasını ele almaktadır. Bu sistem IoT teknolojisi ve bulut bilişim teknolojisi kullanılarak tasarlanmıştır. Akıllı park sistemi park alanlarında araç sayısının sürekli olarak takip edilmesini ve park edilebilecek boş yerlerin belirlenmesini sağlamaktadır. Bunun için park alanlarına yerleştirilen sensörler aracılığıyla araçların giriş ve çıkışları izlenmekte ve böylece park alanındaki mevcut araç sayısı belirlenmektedir. Bu bilgiler park edilebilecek boş yerlerin belirlenmesinde kullanılmakta ve sürücülere bu bilgi sunulmaktadır. Bu da sürücülerin park etmede zaman tasarrufu yapmalarını sağlamaktadır.

Sistemde kullanılan IoT cihazları; sensörler, veri toplama cihazları ve akıllı aydınlatma sistemlerini içermektedir. Bu cihazlar birbirleriyle iletişim kurarak ve verileri bulut bilişim teknolojisi kullanarak depolayarak sürücülerin park etme işlemini daha kolay ve verimli hale getirmektedir.

Bulut bilişim teknolojisinde verilerin depolanması ve işlenmesi için yüksek kapasiteli sunucular kullanılmaktadır. Bu sayede, büyük ölçekli park alanlarının yönetimi için uygun bir çözüm sunulmuş olur. Sistemde kullanılan bulut bilişim teknolojisi sayesinde, tüm veriler merkezi bir veri tabanında toplanmakta; bu veriler sürücülere ve park yöneticilerine hızlı ve güvenilir bir şekilde sunulmaktadır.

Çalışmada; sistem bileşenleri, sistem mimarisi ve uygulama senaryoları hakkında detaylı bilgiler yer almakta olup sistem bileşenleri arasında sensörler, veri toplama cihazları, bulut sunucuları ve mobil uygulamalar yer almaktadır. Sistem mimarisi, cihazlar arasındaki iletişim ve veri akışı hakkında bilgi verirken, uygulama senaryoları sistemin gerçek hayatta nasıl kullanılabileceği hakkında örnekler sunmaktadır.

Sonuç olarak çalışma sürücülerin park etme işlemini daha kolay ve verimli hale getirmek için kullanılabilir bir çözüm sunmaktadır. Akıllı park sistemleri, IoT ve bulut bilişim teknolojileri kullanılarak geliştirildiği için gelecekte park yönetimi konusunda önemli bir rol oynayacakları düşünülmektedir (Pham, Tsai, Nguyen, Dow, Deng, 2015). Pham ve arkadaşlarının bu çalışmasında da görüldüğü gibi araç ağları çok sıradan gündelik işlerimizde bile bize fayda sağlayabilen teknolojiler haline gelmiştir.

2.6. Akıllı Şehirler

Akıllı şehirler teknolojik altyapıyı kullanarak kentleri daha verimli, sürdürülebilir ve yaşanabilir hale getirmeyi amaçlamaktadır. Bu nedenle akıllı şehirlerde kullanılan teknolojiler ve uygulamalar şehirlerin mevcut altyapısına entegre edilmektedir. Örneğin; trafik yönetimi için kullanılan sensörler yol durumunu takip ederek trafik sıkışıklığını önlemek veya azaltmak için alternatif yollar önermektedir. Benzer şekilde enerji yönetimi için kullanılan sensörler, enerji tüketimini takip ederek enerji tasarrufu yapılacak alanları tespit etmektedir. Akıllı şehirlerde kullanılan başka bir teknoloji ise akıllı aydınlatmadır. Bu sistemler sensörlerle kontrol edilmekte ve insanların varlığına göre açılıp kapanmaktadır. Bu sayede enerji tasarrufu sağlanırken, şehirler daha güvenli hale getirilebilir. Akıllı şehirlerde çevre koruma alanında kullanılan bir diğer teknolojide sensörler su ve hava kalitesi gibi faktörleri izlerken, şehir yöneticileri çevre kirliliği ile ilgili sorunları belirleyip hızlı bir şekilde müdahale edebilirler.

Sonuç olarak akıllı şehirler sadece şehirlerde yaşayanların hayatını daha iyi hale getirmekle kalmaz, aynı zamanda kent yönetimini daha verimli ve sürdürülebilir hale getirerek geleceğin şehirleri için bir model oluşturur.

Dünya genelinde birçok ülke akıllı şehirler oluşturmak için çeşitli projeler yürütmektedir. Bu şehirlerden bazıları Barselona, New York City, Tokyo, Singapur, Dubai olarak sayılabilir. Buralarda yer alan sensörler ve akıllı cihazlar sayesinde enerji verimliliği artırmak, trafik akışını yönetmek ve su kaynaklarını verimli bir şekilde kullanmak gibi akıllı şehir teknolojileri ile daha sürdürülebilir bir kent oluşturma planlaması yapılmaktadır.

Yang ve arkadaşlarının çalışmasında akıllı şehirlerdeki akıllı sistemler için akıllı sistemlerin yüksek veri oranlarına ihtiyaç duydukları durumlarda hata vektörü büyüklüğüne dayalı bir adaptif modülasyon stratejisi önerilmektedir. Bu yöntem veri oranını arttırırken aynı zamanda bit hata oranını da düşürerek daha güvenli ve etkili iletişim sağlamaktadır. Bu çalışma özellikle akıllı şehirlerdeki iletişim sistemleri ve enerji

yönetimi sistemleri gibi yüksek veri trafiği gerektiren akıllı sistemler için bir çözüm sunmaktadır (Yang, Huang, Bhardwaj, Hussain, El-Latif & Yu, 2023).

2.7. Akıllı Ulaşım

Akıllı ulaşım trafik akışını optimize etmek, trafik sıkışıklığından kaynaklanan gecikmeleri azaltmak ve çevresel sürdürülebilirliği artırmak için akıllı teknolojilerin kullanıldığı bir ulaşım sistemidir. Bu teknolojiler arasında sensörler, veri analitiği, yapay zeka ve otomasyon gibi birçok farklı altyapı yer almaktadır.

Akıllı ulaşım birçok farklı alanda kullanılmaktadır. Bunlardan önemli olanları aşağıda listelenmiştir:

- Akıllı trafik yönetimi: Trafik yoğunluğunu, trafik sıkışıklığını ve trafik kazalarını azaltmak için sensörler, kamera ve radar sistemleri gibi teknolojiler kullanarak trafik akışı yönetilmektedir.

- Akıllı otopark yönetimi: Otopark yerlerinin izlenmesi ve yönetimi için sensörler ve akıllı park sistemleri kullanılmaktadır.

- Akıllı ulaşım sistemleri: Yolcuların toplu taşıma araçlarını takip etmelerine, yolculuklarını planlamalarına ve ödemelerini yapmalarına olanak tanıyan akıllı sistemlerdir.

- Akıllı araçlar: Sürücü destek sistemleri, otonom araçlar ve elektrikli araçlar gibi akıllı araç teknolojilerini içermektedir.

Araştırmalar akıllı ulaşım teknolojilerinin kullanımının trafik sıkışıklığı, enerji tüketimi ve karbon ayak izi gibi konularda önemli faydalar sağladığını göstermektedir. Akıllı trafik yönetimi teknolojileri trafik sıkışıklığından kaynaklanan zaman kaybını %10-20 oranında azaltabilirken enerji tüketimini %5-10 oranında düşürebilmektedir. Akıllı otopark sistemleri de araç park yeri bulma süresini %40 oranında azaltabilmektedir. Akıllı ulaşım teknolojilerinin yaygınlaşması geleneksel ulaşım sistemlerine göre daha az maliyet gerektirmektedir.

Derawi ve arkadaşları çalışmalarında; akıllı ulaşım sistemlerinin ve nesnelerin internetinin kullanımıyla daha güvenli yollar oluşturulabileceğine yönelik araştırmalar yapmışlardır. Bu çalışma da akıllı ulaşım sistemlerinin özellikleri, bileşenleri ve IoT teknolojisi ile nasıl birleştirilebileceği incelenmektedir. Bu birleşim sayesinde trafik sıkışıklığı, araç kazaları ve güvenlik tehditleri azaltılabilmektedir. Söz konusu çalışmada yer alan birçok örnek uygulama akıllı ulaşım sistemlerinin gerçek hayatta nasıl kullanıldığını göstermektedir. Bu uygulamalar arasında trafik yoğunluğu ve kazaların

izlenmesi, acil durum yönetimi, araç takip sistemleri, sürücü davranışlarının analizi ve otonom araçlar bulunmaktadır.

Çalışma akıllı ulaşım sistemlerinin önemini ve potansiyel faydalarını kanıtlamakta ve bu sistemlerin geliştirilmesinde IoT teknolojisinin kritik rolünü vurgulamaktadır (Derawi, Dalveren, Cheikh, 2020).

Sonuç olarak akıllı ulaşım, ulaşım sistemlerinde sürdürülebilirlik, etkinlik ve güvenlik sağlamak için önemli bir araçtır.

3. MATERYAL VE YÖNTEM

3.1. Dağıtık Sistem

Dağıtık sistem birden fazla bilgisayarın birbirleriyle iletişim halinde olduğu ve birbirleriyle senkronize bir şekilde çalışarak tek bir sistem gibi davrandığı yapıdır. Bu sistemler genellikle ağ üzerinden iletişim kurarak veri ve işlemleri farklı bilgisayarlar arasında paylaşmaktadır (Şekil 1).

Dağıtık sistemler çeşitli uygulamalarda kullanılmakta olup bazı örnekler aşağıda listelenmiştir:

- Bulut bilişim: Bulut bilişim; sunucular, veri merkezleri ve ağlar gibi birçok farklı bileşenden oluşmaktadır. Bu bileşenler birbirleriyle iletişim halinde olup farklı uygulamaların çalıştırılması için bir altyapı sağlamaktadır.

- Akıllı telefon uygulamaları: Akıllı telefon uygulamaları birçok farklı bileşenin bir araya gelmesiyle oluşan dağıtık bir sistemdir. Uygulamalar; telefonun işlemcisinde, sunucularda veya bulut hizmetlerinde çalışabilmektedir.

- Veri tabanı sistemleri: Veri tabanı sistemleri; farklı bilgisayarlar arasında paylaşılan verileri yönetmek için dağıtık bir sistem kullanmaktadır. Veriler farklı sunucularda veya veri tabanlarında depolanabilir ve birbirleriyle senkronize bir şekilde çalışabilirler.

- Dosya paylaşımı: Dosya paylaşımı, birçok farklı kullanıcının dosyalarını birbirleriyle paylaştığı bir dağıtık sistemdir. Bu sistemler ağ üzerinden veya bulut depolama hizmetleri aracılığıyla gerçekleştirilebilmektedir.

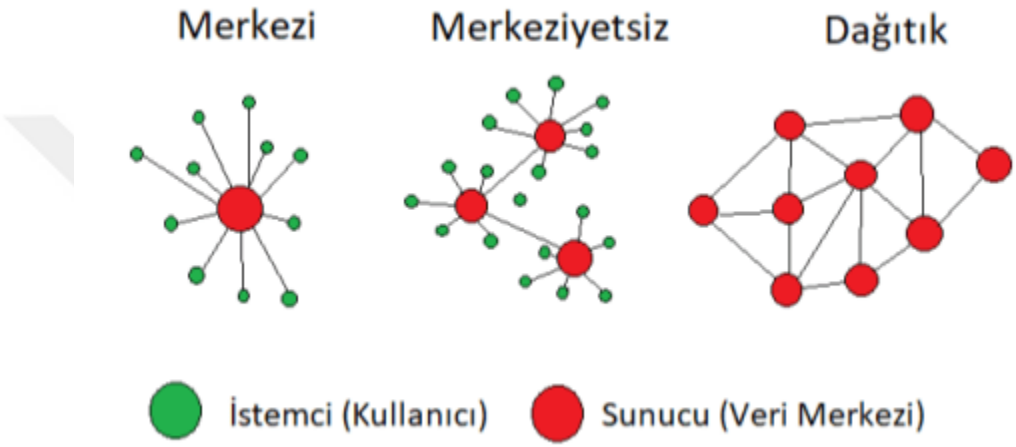
- Sensör ağları: Sensör ağları birçok farklı sensörün veri toplama ve paylaşımı için bir araya geldiği dağıtık sistemlerdir. Sensörler farklı lokasyonlarda veya nesnelerde yer alabilirler ve birbirleriyle iletişim halindedirler.

Bu örnekler dağıtık sistemlerin çeşitli uygulamalarda kullanılabileceğini göstermekte olup yüksek performans, ölçeklenebilirlik ve güvenilirlik gerektiren uygulamalarda önemli bir rol oynamaktadır.

Schmitt ve arkadaşlarının çalışması dağıtık işletim sistemleri ve çok işlemcili sistemler arasındaki etkileşimi ele almaktadır. Çalışmada dağıtık işletim sistemleri ile işlemciler arasındaki ilişkiler incelenmekte ve bu ilişkileri güçlendirmek için bir çözüm önerilmektedir. Bu öneri tek işlemcili bir dağıtık işletim sistemi üzerine işlemci desteği eklenmesini sağlayan bir mimari değişikliği içermektedir. Önerilen mimari, işlemciye özel bellek bölgesi ve bu bölgede tutulan verilere erişim için özel bir protokol kullanarak işlemciler arasında veri paylaşımını mümkün kılmaktadır. Bu protokol paralel işlem

yaparken hatalı durumları tespit etmek ve geri alma işlemlerini yürütmek için bir dizi karşılıklı etkileşim işlemine dayanmaktadır.

Bu çalışma, dağıtık işletim sistemleri ile çok işlemcili sistemler arasındaki etkileşimi geliştirme konusunda önemli bir adım olarak görülmektedir. Çalışmanın sonuçları, daha sonraki dağıtık işletim sistemleri ve paralel programlama çalışmalarında kullanılmış ve dağıtık sistemlerin paralelleştirilmesi konusunda birçok araştırmaya ön kaynak olmuştur (Schmitt, Schmidt, Kaemmer, Gerhold, Schulthess, 2010).



Şekil 1. Sunucu Yönetim Mimarileri

3.1.1. Dağıtık Sistem Gereksinimleri

Dağıtık mimarinin yapılandırılması ve çalışabilmesi için çeşitli bileşenlerin birbiriyle uyumlu ve senkron şekilde çalışılabilmesi gerekmemektedir. Dağıtık sistemlerin gereksinimleri, bu sistemlerin işlevselliği, performansı, güvenilirliği ve ölçeklenebilirliği gibi çeşitli yönlerini kapsamaktadır. Aşağıda dağıtık sistem yapısının gereksinimlerine örnekler verilmiştir:

- İşlevsellik: Dağıtık sistemler birçok farklı kullanıcı ve uygulama için işlevsel olmalıdır. Sistem kullanıcıların ihtiyaçlarını karşılamalı ve hizmetlerini sağlamalıdır.

- Performans: Dağıtık sistemler yüksek performans seviyeleri sağlamalıdır. Bu da yüksek işlem hızları, düşük gecikme süreleri ve hızlı yanıt süreleri anlamına gelmektedir.

- Güvenilirlik: Dağıtık sistemlerde veri kaybı, arızalar ve güvenlik açıkları gibi sorunlarla başa çıkmak için önlemler alınmalıdır.

- Ölçeklenebilirlik: Dağıtık sistemler değişen kullanıcı sayısı ve iş yüküne göre ölçeklenebilir olmalıdır. Bu, daha fazla kaynak ve kapasite gerektiğinde sistemlerin büyütülebilmesi anlamına gelmektedir.

- Uyumluluk: Dağıtık sistemler farklı platformlar, işletim sistemleri ve protokoller arasında uyumlu olmalıdır. Bu, sistemlerin kullanımını kolaylaştırmalı ve daha fazla işletim sistemi veya platformu desteklemelidir.

- Yönetilebilirlik: Dağıtık sistemler kolayca yönetilebilir olmalıdır. Bu; sistemlerin izlenmesini, yönetilmesini ve sorunları çözmeyi kolaylaştırmaktadır.

- Erişilebilirlik: Dağıtık sistemler kullanıcıların erişimini kolaylaştırmalıdır. Bu; da erişim hızları, kullanıcı arayüzleri ve güncelleme yönetimi gibi faktörleri kapsamaktadır.

- Esneklik: Dağıtık sistemler esnek olmalıdır. Bu, farklı hizmetlerin ve uygulamaların entegrasyonunu kolaylaştırmakta ve sistemlerin uyumlu olmasını sağlamaktadır.

- Ölümsüzlük: Dağıtık sistemler çeşitli parçalarında meydana gelebilecek arızalara karşı dayanıklı olmalıdır. Bu, sistemlerin yüksek kullanılabilirlik seviyeleri sağlamasını ve sürekli olarak kullanılabilir olmasını sağlamaktadır.

- Verimlilik: Dağıtık sistemlerin işlemlerini ve hizmetlerini mümkün olan en verimli şekilde gerçekleştirmeleri gerekmektedir. Bu, kaynakları verimli bir şekilde kullanarak enerji tasarrufu ve düşük işletme maliyetleri sağlamaktadır.

- Güvenlik: Dağıtık sistemler; kullanıcı bilgilerini koruması, kimlik doğrulama, şifreleme ve diğer güvenlik önlemlerinin uygulanmasını gerektirmektedir.

- Sorun Çözme Yeteneği: Dağıtık sistemlerin sorunlarla başa çıkabilecek teknik yeterliliklere sahip olması gerekmektedir. Bu, sorunları hızlı bir şekilde tanımlayabilme ve çözüm yolları üretme becerisini içermektedir.

- İzlenebilirlik: Dağıtık sistemlerde izlenebilirlik; hizmetlerin performansını ölçme, kullanıcı deneyimini izleme ve sorunları tespit etme becerilerini kapsamaktadır.

- Yedeklilik: Dağıtık sistemler çalışmada oluşabilecek arızalara karşı yedekli bir yapıya sahip olmalıdır. Bu, kesintisiz bir hizmet sağlar ve sistemlerin yüksek kullanılabilirlik seviyeleri sağlamasını mümkün kılar.

- Uzaktan Erişim: Dağıtık sistemlerde uzaktan kullanıcıların sistemlere erişebilmesi ve işlemleri gerçekleştirebilmesi gereklidir.

- Kaynak Yönetimi: Dağıtık sistemler; kaynakların kullanımı izleme, yönetme ve optimize etme becerisini içermelidir.

- Veri Yönetimi: Veri yönetimi; dağıtık sistemlerin veri kaynaklarını izleme, yönetme ve işleme becerisini kapsamaktadır.

- Uygulama Geliştirme: Dağıtık sistemler, geliştiricilerin sistemlerini kolayca entegre edebilmesini ve uygulamaları oluşturabilmesini sağlamalıdır.

- Bulut Entegrasyonu: Dağıtık sistemler, kullanıcıların verilerine bulut hizmetleri aracılığıyla erişebilmesini ve verileri depolayabilmesini sağlamalıdır.

- Protokol Uyumluluğu: Dağıtık sistemler farklı modüllerin ve uygulamaların birbirleriyle uyumlu olmasını sağlamalıdır.

Bu gereksinimler, dağıtık sistemlerin başarılı bir şekilde işlemesi ve kullanıcıların ihtiyaçlarını karşılaması için önemlidir. İşlevselliği, performansı, güvenilirliği, ölçeklenebilirliği, uyumluluğu, yönetilebilirliği ve erişilebilirliği destekleyen bir dağıtık sistem; işletmeler ve kullanıcılar için kolay yönetilebilir faydalar sağlayabilir.

3.1.2. Dağıtık Sistemin Kullanım Amaçları ve Alanları

Dağıtık sistemler, büyük ölçekli uygulamaların zaman ve bellek gereksinimi konusunda karşılaştıkları engelleri aşmak için ortaya atılan bir mimari türüdür. HDD gibi donanım kaynakları ve dosyalar, veri tabanları gibi yazılım kaynakları paylaşımlı kullanılmaktadır ve bileşenler senkron olarak çalışmaktadır.

Dağıtık sistemler, birden fazla bilgisayar ve ağ cihazını birbirine bağlayarak verilerin ve işlemlerin farklı noktalardan işlenebilmesini ve depolanabilmesini sağlayan sistemlerdir. Bu nedenle dağıtık sistemlerin kullanım amaçları ve alanları oldukça geniştir. Aşağıda bazı örnekleri verilmiştir:

- Büyük veri işleme: Dağıtık sistemler büyük veri işleme işlevselliği sağlamakta ve büyük veri setlerinin işlenmesini kolaylaştırmaktadır.

- Bulut bilişim: Bulut bilişim dağıtık sistemlerin bir uygulama şeklidir. Bu bilgisayar kaynaklarının internet üzerinden paylaşılmasını ve kullanılmasını sağlamaktadır.

- Yüksek performanslı hesaplama: Dağıtık sistemler yüksek performanslı hesaplamalar için idealdir. Bu, bilimsel hesaplama, mühendislik hesaplama, finansal hesaplama ve benzeri alanlarda kullanılabilmektedir.

- Web hizmetleri: Dağıtık sistemler web hizmetleri sağlamak için de kullanılabilir. Bu, web tabanlı uygulamaların web servislerinin ve diğer internet tabanlı hizmetlerin geliştirilmesini kolaylaştırmaktadır.

- Paralel işleme: Dağıtık sistemler paralel işleme işlevselliği sağlamaktadır. Bu, birden fazla işlemci veya işlemci çekirdeği kullanarak bir işlemin daha hızlı ve verimli bir şekilde işlenmesini sağlamaktadır.

- Dağıtık veri tabanları: Dağıtık sistemler; veri tabanı işlevselliği sağlayarak verilerin farklı bilgisayarlar arasında dağıtılmasını ve işlenmesini kolaylaştırmaktadır.

- Akıllı ev sistemleri: Akıllı ev sistemleri, farklı ev aletlerinin birbirine bağlanmasını ve kontrol edilmesini sağlayan sistemlerdir. Bu sistemler genellikle dağıtık sistemlerin kullanımını içermektedir.

- İnternet of Things (IoT): IoT cihazları birbirleriyle ve internetle bağlantılı cihazlardır. Bu cihazlar genellikle dağıtık sistemlerin kullanımını içermektedir.

- Telekomünikasyon: Telekomünikasyon şirketleri, dağıtık sistemleri ağ yönetimi ve veri işleme için kullanabilmektedir.

- Finansal hizmetler: Finansal hizmetler; dağıtık sistemleri para transferleri, hisse senedi işlemleri ve diğer finansal işlemler için kullanabilmektedir.

Dağıtık sistemler günümüzde birçok alanda kullanılmaktadır ve kullanım alanları giderek artmaktadır. Bunun nedenlerinden biri, dağıtık sistemlerin ölçeklenebilirliği ve esnekliği sağlamasıdır. Birden fazla cihazın birbirine bağlanmasıyla oluşturulan dağıtık sistemler özellikle büyük veri işleme, paralel işleme, yüksek performanslı hesaplama, web hizmetleri, IoT ve akıllı ev sistemleri, finansal hizmetler ve telekomünikasyon alanlarında yaygın olarak kullanılmaktadır.

Büyük veri işleme bugünün dünyasında önemli bir rol oynamaktadır ve dağıtık sistemler büyük veri setleri ile çalışmak için ideal bir araçlardır. Büyük veri setleri tek bir bilgisayarın işleyebileceği kapasitenin ötesinde olduğu için dağıtık sistemler bu verileri paralel olarak işleyebilir ve böylece işlem süresini kısaltabilir.

Paralel işleme dağıtık sistemlerin bir başka önemli kullanım alanıdır. Paralel işleme, birden fazla işlemci veya işlemci çekirdeği kullanarak bir işlemin daha hızlı ve verimli bir şekilde işlenmesini sağlamaktadır. Bu nedenle bilimsel hesaplama, mühendislik hesaplamaları ve finansal hesaplama gibi alanlarda dağıtık sistemler sıkça kullanılmaktadır.

Yüksek performanslı hesaplama, benzer şekilde dağıtık sistemlerin bir diğer önemli kullanım alanıdır. Bu sistemler genellikle büyük ölçekli hesaplama problemlerinin çözümü için kullanılmaktadır. Bu problemler arasında nükleer enerji simülasyonları, hava tahmini, uzay araştırmaları ve benzerleri yer almaktadır.

Web hizmetleri dağıtık sistemlerin bir başka önemli kullanım alanıdır. Bu sistemler, web tabanlı uygulamaların geliştirilmesi ve web servislerinin sağlanması için kullanılmaktadır. Örneğin; bir web uygulaması kullanıcılara farklı yerlerdeki sunucular arasında veri işleme işlevselliği sağlamaktadır.

IoT ve akıllı ev sistemleri dağıtık sistemlerin kullanımını içeren bir başka alandır. Bu sistemler, birbirine bağlı cihazların verilerini işlemek ve kontrol etmek için dağıtık sistemleri kullanmaktadır. Bu sayede akıllı ev sistemleri bir evdeki cihazların tümünü birbirine bağlayabilmekte ve evin otomatik kontrolünü sağlayabilmektedir.

Bunlar ve bunlar gibi daha birçok alanda kullanılmakla beraber yeni ortaya çıkan birçok alanla da dağıtık sistemler kullanım alanını genişletmektedir.

3.1.3. Dağıtık Sistemin Karakteristik Özellikleri

Dağıtık sistemlerin temel karakteristik özellikleri aşağıda listelenmiştir:

- Ölçeklenebilirlik: Dağıtık sistemler, düğümlerin eklenmesi veya çıkarılmasıyla kolayca ölçeklendirilebilmektedir.

- Esneklik: Dağıtık sistemler, değişen iş yükleri ve sistem gereksinimlerine hızlı bir şekilde yanıt verebilmektedir.

- Yüksek performans: Dağıtık sistemler, birden çok cihazın bir araya gelmesiyle yüksek performans ve işlem gücü sağlamaktadır.

- Yüksek kullanılabilirlik: Dağıtık sistemler, tek bir noktanın arızalanması durumunda dahi çalışmaya devam edebilmektedir.

- Verimlilik: Dağıtık sistemler, iş yükünü birden çok cihaza dağıtarak cihazların kaynaklarının daha verimli bir şekilde kullanılmasını sağlamaktadır.

- İletişim: Dağıtık sistemler, farklı cihazlar arasında iletişim sağlamak için çeşitli yöntemler kullanmaktadır.

- Güvenlik: Dağıtık sistemler, verilerin güvenliği ve gizliliği için farklı güvenlik mekanizmaları kullanmaktadır.

- Yönetim: Dağıtık sistemler, farklı düğümleri yönetmek ve kontrol etmek için özel yönetim araçlarına ihtiyaç duymaktadır.

- Heterojenlik: Dağıtık sistemler, farklı cihazlar farklı işletim sistemleri veya donanımlara sahip olabileceği için bu farklılıkları yönetebilecek esneklikte olmalıdır.

- Dağıtık veri depolama: Dağıtık sistemler, verilerin farklı cihazlar arasında depolanması ve işlenmesi için özel veri depolama yöntemleri kullanmaktadır.

3.1.4. Dağıtık Sistemin Avantajları ve Dezavantajları

Dağıtık sistemlerin avantajları şunlardır:

- Ölçeklenebilirlik: Daha fazla cihaz ekleyerek sistemin ölçeğini arttırabilmektedir.

- Yüksek performans: Dağıtık sistemler, birden fazla cihazın işlem gücünü birleştirerek yüksek performans sağlamaktadır.

- Yüksek kullanılabilirlik: Dağıtık sistemler, tek bir cihazın arızalanması durumunda bile çalışmaya devam etmektedir.

- Esneklik: Dağıtık sistemler, iş yükleri ve sistem gereksinimlerindeki değişikliklere hızlıca uyum sağlamaktadır.

- Verimlilik: Dağıtık sistemler, cihazların kaynaklarını daha verimli kullanarak enerji tasarrufu yapmaktadır.

- Güvenlik: Dağıtık sistemler, verilerin güvenliği ve gizliliği için çeşitli güvenlik önlemleri almaktadır.

- İletişim: Dağıtık sistemler, farklı cihazlar arasındaki iletişimi kolaylaştırmaktadır.

- Yönetim: Dağıtık sistemler, düğümleri yönetmek ve kontrol etmek için özel yönetim araçları sunmaktadır.

Dağıtık sistemlerin dezavantajları şunlardır:

- Karmaşıklık: Dağıtık sistemlerin sistem yapısı karmaşık olduğundan, yönetimi ve sorun giderme işlemleri zor olabilmektedir.

- İletişim maliyeti: Dağıtık sistemlerin farklı cihazlar arasındaki iletişim maliyeti yüksek olabilmekte ve sistem performansını etkileyebilmektedir.

- Güvenlik sorunları: Dağıtık sistemlerin kullanımında veri güvenliği ve gizliliği konusunda ciddi sorunlar ortaya çıkabilmektedir.

- Veri bütünlüğü: Dağıtık sistemlerde farklı cihazlar arasında veri bütünlüğü sorunları ortaya çıkabilmektedir.

- Sorunların tespiti: Dağıtık sistemlerde sorunların tespiti ve çözümü zor olabilmektedir.

3.1.5. Dağıtık Sistem Örnekleri

Dağıtık sistemler farklı donanım ve yazılım bileşenlerinden oluşan ve ağ üzerinden birbirleriyle iletişim halinde olan sistemlerdir. Bu sistemler farklı uygulama alanlarında kullanılmaktadır. Bu uygulama alanlarının bazı örnekleri şunlardır:

-Bulut Bilişim: Bulut bilişim internet üzerinde dağıtık bir şekilde sunulan hizmetleri ifade etmektedir. Örneğin; bir şirketin müşteri ilişkileri yönetimi yazılımını bulut bilişim aracılığıyla kullanması, bu yazılımın bir dağıtık sistem üzerinde çalıştığı anlamına gelmektedir.

- Peer-to-Peer Dosya Paylaşımı: Bu sistemler dosya paylaşımı için kullanılmaktadır. Kullanıcılar birbirlerinin dosyalarını paylaşmakta ve ağda depolanan dosyaların kopyaları barındırılmaktadır. Bu sistemlerin örnekleri arasında BitTorrent ve eMule sayılabilir.

- Dağıtık Veri Tabanı Yönetimi: Bu sistemler farklı lokasyonlarda bulunan veri tabanlarının yönetimi için kullanılmaktadır. Veriler, ağ üzerinden birbirleriyle senkronize edilmekte ve yönetmektedir. Örneğin ATM'ler ve bankalar arasındaki veri tabanı işlemleri dağıtık veri tabanı yönetimi kapsamında gerçekleştirilmektedir.

- Akıllı Telefonlar: Akıllı telefonlar birden fazla bileşenin bir araya gelmesiyle oluşan dağıtık bir sistemdir. İşlemci, ekran, kamera, hoparlör, mikrofon gibi bileşenler birbirleriyle uyumlu bir şekilde çalışarak kullanıcıya hizmet vermektedir.

- Bulut Oyun: Oyunların internet üzerinde sunulduğu bir sistemdir. Oyunlar sunucular tarafından yönetilmekte ve oyuncular internet bağlantısı aracılığıyla oyunları oynamaktadır. Örneğin Google Stadia ve PlayStation Now bulut oyunlarıdır.

3.1.6. Dağıtık Sistemlerde Güncelleme

Dağıtık sistemlerde güncelleme işlemi oldukça önemlidir çünkü sistemdeki herhangi bir bileşenin güncellenmesi diğer bileşenleri de etkileyebilir. Bu nedenle güncelleme işlemi özenle planlanmalı ve yürütülmelidir.

Güncelleme işlemi genellikle şu aşamaları içerir:

a) Güncelleme Stratejisi Belirleme: Güncelleme işlemi öncesinde hangi bileşenlerin güncelleneceği ve ne zaman güncelleneceği gibi konular belirlenmelidir. Güncelleme işlemi sırasında kullanıcıların sistemden faydalanmasını sürdürebilmesi için alternatif yöntemler de belirlenebilmektedir.

b) Güncelleme Paketinin Hazırlanması: Güncelleme paketi değiştirilen kodlar, güncellenen veriler ve diğer ilgili dosyaları içermektedir. Bu paket, sistemdeki diğer bileşenlerle uyumlu olacak şekilde hazırlanmalıdır.

c) Güncelleme Yayınlama: Güncelleme paketi sistemdeki bileşenlere yayınlanmalıdır. Yayınlanan güncelleme paketinin, diğer bileşenlerle uyumlu bir şekilde çalışabilmesi sistemdeki tüm bileşenlerin güncellenmesi gerektirebilir.

d) Güncelleme İşleminin Gerçekleştirilmesi: Güncelleme işlemi belirlenen stratejiye uygun olarak gerçekleştirilmelidir. Güncelleme işlemi sırasında, sistemde herhangi bir kesinti yaşanmaması için alternatif yöntemler kullanılabilir.

e) Test ve Onaylama: Güncelleme işlemi tamamlandıktan sonra sistem test edilmeli ve kullanıcılar tarafından onaylanmalıdır. Bu aşama güncelleme işleminin başarılı olup olmadığını kontrol etmek için önemlidir.

Güncelleme işlemi dağıtık sistemlerde oldukça önemlidir ve dikkatli bir şekilde planlanmalı ve yürütülmelidir. Güncelleme işleminin başarılı bir şekilde tamamlanması; sistemdeki güvenlik açıklarının kapatılması ve yeni özelliklerin eklenmesi gibi birçok avantaj sağlayabilir.

3.2. HTTP (Hyper Text Transfer Protocol)

HTTP, Hyper Text Transfer Protocol'ün (Hiper Metin Aktarım Protokolü) kısaltmasıdır. Genellikle web sitelerinde kullanılan iletişim protokolü olup internet üzerinden veri alışverişi yapmak için kullanılmaktadır. HTTP istemci-sunucu mimarisi ile çalışmaktadır. İstemci web tarayıcısı gibi bir uygulama olabilirken sunucu ise web sitesini barındıran bir bilgisayar olabilir.

HTTP'nin çalışma prensibi şöyledir:

i) İstemci sunucuya bir kaynak (örneğin bir web sayfası) isteği gönderir.

ii) Sunucu isteği alır ve kaynağı oluşturan verileri yanıt olarak geri gönderir.

iii) İstemci aldığı yanıtı işler ve kaynağı görüntüler veya başka bir işlem yapar.

HTTP varsayılan olarak 80 numaralı bağlantı noktasını kullanmaktadır. İstemci ve sunucu arasındaki iletişim, istek ve yanıt mesajları şeklinde gerçekleşmektedir. İstek mesajı, istemcinin sunucuya gönderdiği talebi, yanıt mesajı ise sunucunun isteği karşılamak için gönderdiği cevabı içermektedir. Bu mesajlar; HTML, CSS, JavaScript, resimler veya diğer veri türlerini içerebilmektedir.

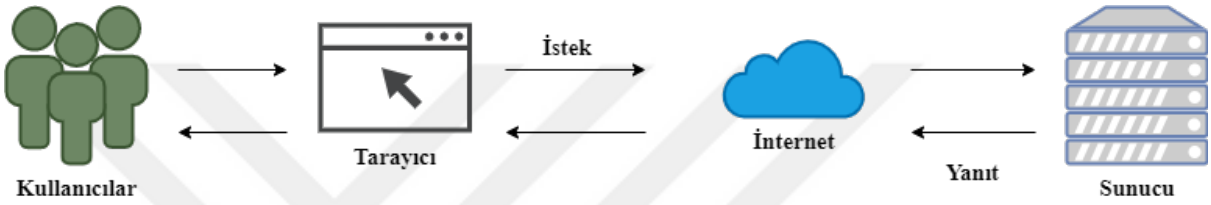
HTTP ayrıca birçok farklı yöntem (GET, POST, PUT, DELETE vb.) ve durum kodu (200 OK, 404 Not Found, 500 Internal Server Error vb.) içermektedir. GET yöntemi, sunucudan bir kaynağı almak için kullanılırken POST yöntemi, sunucuya yeni veriler göndermek için kullanılmaktadır.

HTTP web sitesi trafiği için yaygın olarak kullanılan bir protokoldür ve web tarayıcıları, sunucular ve diğer uygulamalar tarafından desteklenmektedir.

Günümüzde sıklıkla kullanılan internet tarayıcıları arama sırasında internet sitelerinin alan adından önce http etiketini eklemektedir. Bunun sebebi internet

kullanıcılarının pek çok farklı web sitesi içinde indirme yapmak istediğinde internet tarayıcısının(browser) HTTP yolu ile sunucudan bir istekte(request) bulunmasıdır. Sunucuların buradaki görevi herhangi bir internet sitesine giriş yapıldığında, bu isteği http bünyesinde verilen bir komut olarak anlamak ve cevap(response) göndermektir. (Şekil 2) Yani söz konusu komuta göre indirme işlemi yapılmaktadır.

Günümüzde internet ortamında baktığımızda sunucular arasındaki bilgi aktarımı ile bu bilgilerin kontrollerini sağlayan birçok protokol mevcuttur. HTTP veri iletişiminin temelidir. Sebebi ise http uzantısı olmadığı taktirde bir internet ortamından veri almanın mümkün olmamasıdır.



Şekil 2. HTTP çalışma prensibi

3.3. Ağ Topolojisi

Ağ topolojisi, bir bilgisayar ağındaki cihazların fiziksel veya mantıksal olarak nasıl bağlandığını ve düzenlendiğini tanımlayan yapıdır. Topoloji bir ağın veri iletimi ve yönetimi için kullanılan altyapıyı belirlemektedir.

Bir ağın topolojisi, ağdaki cihazlar arasındaki bağlantıyı ve veri iletişimini nasıl yöneteceğini belirlemektedir. Ağ topolojisi; ağda kullanılan donanım, yazılım ve protokoller gibi faktörlerle birlikte düşünülmelidir.

En yaygın ağ topolojilerinden bazıları şunlardır:

- Yıldız Topolojisi: Tüm cihazların bir merkezi yönlendiricinin etrafında bağlı olduğu topolojidir. Merkezi yönlendirici, cihazlar arasındaki veri iletimini yöneterek ağın sorunsuz çalışmasını sağlamaktadır.

- Ağaç Topolojisi: Merkezi bir yönlendiriciye bağlı olarak hiyerarşik bir yapıda düzenlenen bir ağ topolojisidir. Ağın her düzeyinde birbirine bağlı alt ağlar bulunmaktadır.

- Halka Topolojisi: Tüm cihazların birbirine bağlandığı bir halka yapısında düzenlenen topolojidir. Bu topolojide her cihazın veriyi göndermek ve almak için birbirine bağlı komşu cihazlara ihtiyacı vardır.

- Mesh Topolojisi: Tüm cihazların birbirine bağlı olduğu bir topolojidir. Bu topolojide, her cihazın birden fazla doğrudan bağlantısı vardır ve veri iletimi birden fazla yoldan gerçekleştirilebilmektedir.

Ağ topolojileri; ağın boyutu, kullanım amacı, performans gereksinimleri ve maliyet faktörleri gibi çeşitli faktörlere bağlı olarak seçilmekte ve uygulanmaktadır. Ayrıca modern ağlarda genellikle karmaşık topolojiler kullanılmakta ve birden fazla topolojinin birleştirilmesiyle daha esnek ve ölçeklenebilir bir yapı elde edilmektedir.

3.4. Kriptolojik Özet Fonksiyonu (SHA256)

Çalışmamızda veri doğruluğunu kesinleştirmek için SHA 256 veri özeti fonksiyonu kullanılmıştır.

Michael Christopher Xenya ve Kester Quist-Aphetsi'in çalışması, dijital delil incelemelerinde kullanılan SHA-256 algoritmasını kullanarak hesap denetimi yapmak için kriptografik yöntem sunmaktadır.

Bu çalışmada hesap denetimi süreci, her hesap için bir SHA-256 özet değeri oluşturarak başlamakta, bu özet değerleri daha sonra bir bilgisayar veri tabanına kaydedilmektedir. Hesap denetim işlemi gerçekleştirildiğinde hesap bilgilerinin SHA-256 özet değeri hesaplanmakta ve veri tabanındaki özet değeri ile karşılaştırılmaktadır. Eşleşme durumunda hesap bilgileri geçerli olarak kabul edilmektedir.

Bu yöntem hesap denetimi sürecinde sahtecilik ve veri manipülasyonu gibi olası güvenlik tehditlerine karşı koruma sağlamaktadır. Bu yöntem, hesap denetimi sürecinin tamamlanması için gereken zamanı azaltmakta ve denetim işleminin doğruluğunu artırmaktadır.

Bu çalışma, dijital delil incelemelerinde hesap denetimi sürecinde kullanılan mevcut yöntemlerin güçlü yönlerini ele alarak daha güvenli ve doğru bir yöntem sunmaktadır. SHA-256 algoritması gibi kriptografik yöntemler, dijital delil incelemeleri gibi güvenlik açısından hassas işlemlerde kullanılan verilerin korunması için önemli bir rol oynamaktadır (Xenya, Quist-Aphetsi, 2019).

SHA-256, 256 bit uzunluğunda kriptografik özet fonksiyonudur, SHA-2 ailesinin bir parçasıdır ve Amerikan Ulusal Güvenlik Ajansı (NSA) tarafından geliştirilmiştir. SHA-256, verilen bir mesajdan 256 bit uzunluğunda bir özet değeri üretmektedir.

Kriptografik özet fonksiyonları verilen herhangi bir boyutta, veriyi sabit boyutta bir özet değerine dönüştürmektedir. Bu özet değeri, verinin kendisiyle ilgili bilgileri koruyan ancak tamamen farklı bir değerdir. Kriptografik özet fonksiyonları, verinin

bütünlüğünü doğrulama, kimlik doğrulama ve mesaj doğrulama gibi işlemlerde yaygın olarak kullanılmaktadır.

SHA-256 bir dizi matematiksel işlem kullanarak verilen mesajın özet değerini hesaplamaktadır. Bu işlemler arasında döngüler, mantıksal işlemler, bit kaydırmaları ve özel olarak tasarlanmış özetleme fonksiyonları bulunmaktadır. SHA-256'nın bir diğer özelliği herhangi bir mesaj için benzersiz bir özet değeri üretebilmesidir. Yani iki farklı mesajın SHA-256 özetleri aynı olamaz.

SHA-256 birçok uygulamada kullanılan güvenli bir özetleme algoritmasıdır. Bu uygulamalar arasında parola doğrulama, veri bütünlüğü doğrulama, dijital imza ve blok zinciri teknolojisi yer alır.

3.5. Güncelleme Uygulamalarının Kullanım Amacı ve Alanları

Oluşturulan yazılımların güncellenme ihtiyacı, tüm uygulamalar için gereksinim duyulan bir alan olmakla beraber yazılımların çevrimiçi kullanılması ve aynı uygulamayı kullanan kullanıcı sayısının birden fazla olmasıyla daha da karmaşık hale gelmiştir. Gelişmiş güncelleme tarzlarının ilk örnekleri ise veri tabanı sistemlerinin gelişmesi ve çoklu erişime imkân vermesiyle, daha yazılım planlanırken güncellemenin nasıl olması gerektiğinin planlanmasını gerektirmektedir. Veri tabanlarında toplu erişim imkânı, mutex ve semaphore kullanımıyla önemli ölçüde önlenmiştir. Fakat bu çalışmanın da odak noktasını teşkil eden dağıtık sistemler ve her cihazın kendi içerisinde çalışan benzer uygulamaya sahip olması ve bu uygulamanın çevrimiçi olarak güncellenmesi sorunu, daha güncel bir problem olarak karşımıza çıkmaktadır.

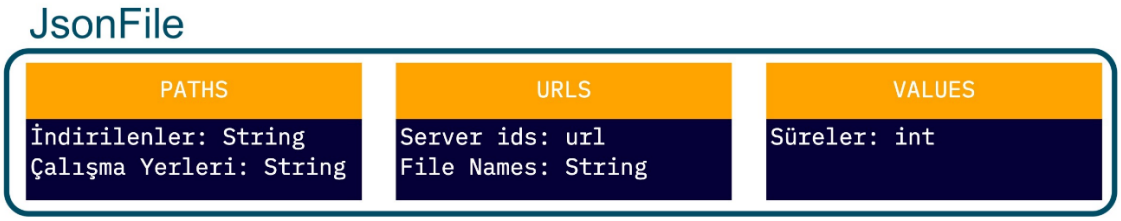
Elektrikli arabaların akıllı cihazlar kapsamında değerlendirilmesi, daha da artan denetim ve koordinasyon sorununu gündeme taşımıştır. Bununla beraber şarj istasyonlarının sayısının az olması ve batarya sisteminin benzin ve diğer yakıt türlerine göre daha yavaş doluyor olması, istasyonların da çevrimiçi olarak takip edilip güncellenebilir olması ihtiyacını doğurmuştur.

Bu çalışmada da bir dispanser ve elektrikli pompanın uzaktan internet tabanlı olarak güncellenmesine odaklanılmıştır. Tek sunucuya bağlı birçok dağıtık cihazın eş veya yakın zamanlı olarak güncellenmesi ve bu sırada çalışmakta olan uygulamaların hassasiyetine zarar vermeyecek, paralel olarak çalışan güncelleyici önerilmektedir.

3.6. Uygulamanın Çalışma Prensibi

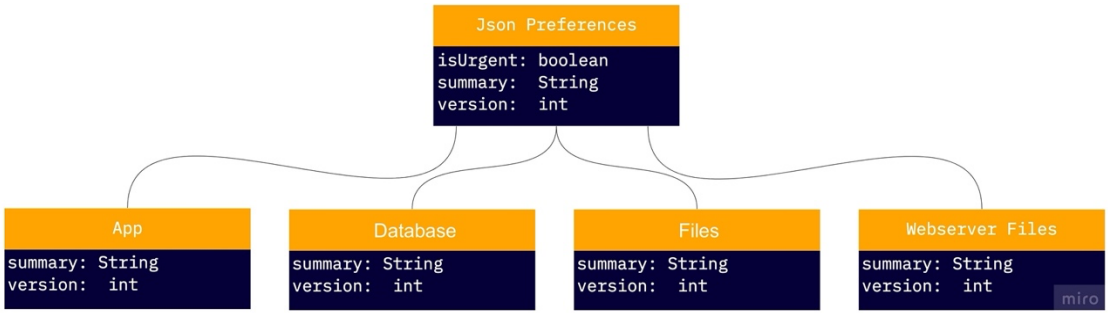
3.6.1. İndirme Şablonu

İçerisinde üç ayrı paketin olduğu; Şekil-3'de gösterilen dosyadır.



Şekil 3. İndirme Şablonu

İndirme şablonu, Urls'in içerisinde, server Id'si ve indirilecek dosya adlarının bulunduğu pakettir. Paths içerisinde, indirilen dosyaların nereye indirileceği ve nereye açılacağı bilgileri tutulmaktadır. Values içerisinde, dosya boyutlarına göre bekleme sürelerinin değişkenliği göz önünde bulundurularak ayarlanmış tetikleme süreleri bilgisi bulunmaktadır.



Şekil 4. Ayarlar Dosyası Şablonu

Ana Json dosyasının içerisinde Şekil-4'de gösterildiği gibi cihazın amacına göre verileri değişkenlik gösterebilen fakat her cihazda aynı şablonda bulunan App, Database, Files, WebServerFiles adıyla versiyon ve özet verisinin bulunduğu indirme kontrol dosyası bulunmaktadır. Ek olarak JsonFile adıyla uygulamanın genel sürümün aciliyetini ve özet sabitini gösteren bir diğer paket bulunmaktadır. Bu iki paket de güncelleme uygulamasının çalışması sırasında hangi uygulama veya dosyaların güncellenebileceği bilgisini tutarak, kontroller sırasında severdeki dosyalarla uyum düzeninin takip edildiği ana konfigürasyon dosyalarıdır. Ayrı bir yapıda saklanmasındaki ana neden, uygulamanın içerisine hardcode olarak yazıldığında esneklik yönünden kısıtlamalar meydana getirmesi ve güncelleme uygulamasının daha sonraki geliştirmelerine engel olabilecek olmasıdır.

Bu değişkenler, uygulama içerisinde değişken isimleriyle çağırılarak program versiyonu ve indirme doğruluğu denetimlerini sağlamaktadır.

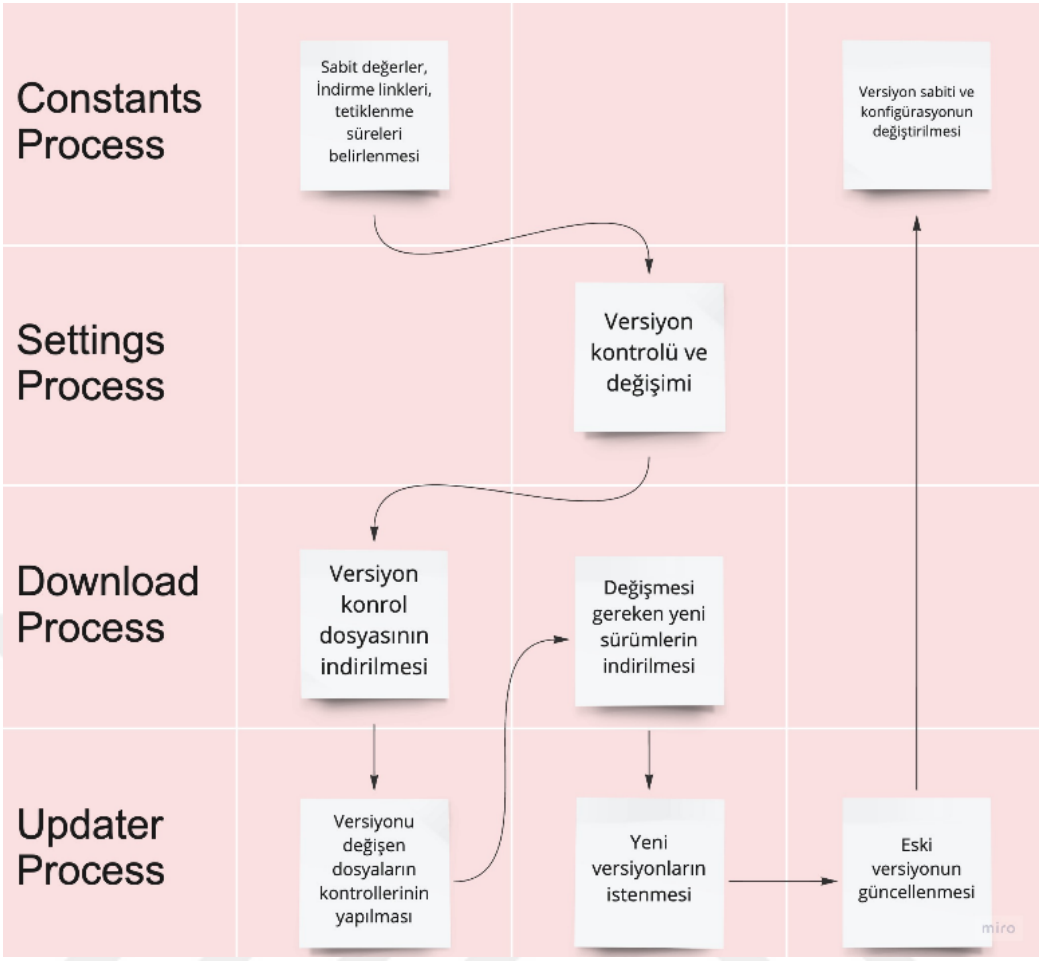
3.6.2. Sınıflandırma ve İşleyiş ayarları

Bu çalışmadaki güncelleme uygulaması; ayarlar, kısıtlamalar, indirme işlemleri, ana işleyiş ve kontrolün sağlandığı dört ana sınıftan oluşmaktadır.

Ayarlar sınıfında, sunucudaki sürüm ve cihaz ile alakalı 4 farklı uygulamanın ve dosyalarının güncelliği periyodik aralıklarla kontrol edilmektedir (Şekil 5). Bu kontrol edilen sistemler; ana json dosyasının uyumluluğu, cihazın görev uygulamasının uyumluluğu, veri tabanın uyumluluğu, web dosyalarının uyumluluğudur. Kontrol dosyası indirildikten sonra bu sınıfta güncellenecek uygulamaların yeni halleri uygun yere indirilmekte ve güncellenmeyecek olanlar elemine edilmektedir.

Kısıtlamalar sınıfında; hangi dosyanın indirilmesi için ne kadar bekleneceği, dosyaların nereye indirileceği, güncellenecek uygulamaların nereden indirileceği ve bu kontrol döngüsünün ne kadar aralıklarla tekrarlanacağı belirlenmektedir.

İndirme işlemleri sınıfında; indirilmesi gereken dosyaların sunucu erişimleri, indirme hızı ve indirilme yüzdesi kuyruk yapısına uygun olarak karmaşıklığa imkân vermeden, gerekli ek kontroller yapılarak sağlanmaktadır. İndirilen dosyaların başarılı veya başarısız indirilmesine göre kuyruğu yeniden işletmesi ve bu sırada oluşabilecek gereksiz indirmelerin önüne geçmek ve hatalı indirilen dosyanın tek başına tetiklenmesini sağlamak bu sınıfın denetimindedir.



Şekil-5

Uygulamanın çalışması Şekil-5'de gösterildiği gibi ilk olarak sabitlenmektedir, ayarlar ve kontrol dosyası indirme tetiklemeleri için gerekli düzenlemeler yapılmaktadır. Daha sonra cihazın durumuna göre indirme yapılacak dosyalarda hata yaşanmaması için gerekli klasörlerin varlık denetimleri yapılmaktadır. Akabinde kontrol dosyasının düzenli tetiklenme sinyali başlatılmaktadır. İlk aşamada, güncelleme dosyaları hali hazırda indirilmiş olarak var mı onun kontrolü yapılmakta ve varsa direk güncelleme işlemine geçilmektedir; yoksa sunucudaki sürüm kontrol dosyası indirilmektedir. Bu çalışmanın kattığı yeniliklerden birisi bu dosyanın indirilme tetiklenmesinin her tetiklenmede birbirinden bağımsız ve farklı tetiklenme aralığına sahip olmasıdır. Çünkü aynı sisteme bağlı binlerce veya daha fazla dağıtık cihaz olduğunda, hepsi aynı anda sunucudan dosya talebinde bulunduğunda sunucu cevap veremeyebilir ve indirmeleri sonsuz bir döngüye sokabilir. Bunu engellemek için sunucunun gücü ve işlem hızı daha da arttırılabilmektedir fakat o zaman da maliyette artış yaşanacaktır. Bunun dışında tüm indirilen dosyaların doğruluk kontrolünün yapılabilmesinde 256 bytelık özet verilerinden faydalanılmaktadır.

3.6.3 Log yapısı

Aşağıda güncel tarihli olarak aktifte koşan bir istasyondaki tetiklenme ve işleme alınma logları gösterilmiştir.

31.03.2023 14:20 Updater Requested Url:
http://255.255.255.0:8080/tcpcontroller/UpdaterPreference.json
31.03.2023 14:20 Updater Network Accessible: http://
255.255.255.0:8080/tcpcontroller/UpdaterPreference.json
31.03.2023 14:20 Updater Download Succeeded
31.03.2023 14:20 Updater UpdaterPreferences Corrected
31.03.2023 14:20 Updater Versions Not Changed

31.03.2023 14:50 Updater Requested Url: http://
255.255.255.0:8080/tcpcontroller/UpdaterPreference.json
31.03.2023 14:50 Updater Network Accessible: http://
255.255.255.0:8080/tcpcontroller/UpdaterPreference.json
31.03.2023 14:50 Updater Download Succeeded
31.03.2023 14:50 Updater UpdaterPreferences Corrected
31.03.2023 14:50 Updater Versions Not Changed

31.03.2023 15:20 Updater Requested Url: http://
255.255.255.0:8080/tcpcontroller/UpdaterPreference.json
31.03.2023 15:20 Updater Network Accessible: http://
255.255.255.0:8080/tcpcontroller/UpdaterPreference.json
31.03.2023 15:20 Updater Download Succeeded
31.03.2023 15:20 Updater UpdaterPreferences Corrected
31.03.2023 15:20 Updater Versions Not Changed

31.03.2023 15:50 Updater Requested Url: http://
255.255.255.0:8080/tcpcontroller/UpdaterPreference.json
31.03.2023 15:50 Updater Network Accessible: http://
255.255.255.0:8080/tcpcontroller/UpdaterPreference.json
31.03.2023 15:50 Updater Download Succeeded
31.03.2023 15:50 Updater UpdaterPreferences Corrected
31.03.2023 15:50 Updater Versions Not Changed

Yukarıda da gösterildiği gibi internete erişimle ilgili herhangi bir sorun yaşamayan ECU'larda güncelleme işlemi çok kısa saniyeler içerisinde gerçekleşebilmektedir. Herhangi bir erişim veya indirme hatası oluşması durumunda ise deneme sayısı sıklaştırılıp, bağlantı sistemleri tekrar kontrol edilerek uygulamanın güncelleştirilmesi kaçınılmaz hale getirilmeye çalışılmıştır.

3.7. Algoritmanın Karmaşıklık Analizi

Yapılan analizler sonucunda algoritmanın en iyi duruma göre zaman karmaşıklık değeri big O notation'una göre O(1)'dir, o'da sadece kontrol dosyası güncellendiği durumdur.

Ortalama duruma göre zaman karmaşıklık değeri big O notation'una göre O(n) şeklinde hesaplanmıştır.

En kötü duruma göre zaman karmaşıklık değeri big O notation'una göre O(n^2) şeklinde hesaplanmıştır.

4. ÇALIŞMA PRENSİBİ VE GÖZLEM

4.1. Çalışmanın Kattığı Yenilikler

Bu çalışmada daha önceki güncelleme çalışmalarından farklı olarak birden fazla yeni yöntem geliştirilmiş, denenmiş ve kullanılmıştır. İlk olarak benzer işlevlere sahip tek sunucudan yönetilebilen dağıtık cihazlarının hepsinin belirlenen sınır aralıklarında fakat zamanı netleştirilmeden, sürekli rassal olarak değişen zamanlarda güncelleme isteklerinin yapılmasıyla sunucuya tek zamanda yüklenme sorununa alternatif bir çözüm oluşturulmuştur. Bir diğeri cihaza, işletim sistemine veya içerisindeki uygulamaya bağlı olmayan bir güncelleme uygulaması yardımıyla diğer uygulamalardan tam bağımsız ek bir güncelleyici olmasıdır. Son olarak da birbirinden bağımsız sonsuz farklı uygulamayı güncelleyebilir olmasıdır.

4.2. Uygulamanın Çalışması

Sistem Linux işletim sistemine sahip cihazlar üzerinde test edilmiştir. İlk olarak cihazın açılış anından itibaren otomatik çalışmaya başlamakta, daha sonra 1710000 ile 420000 mili saniyelik aralıklarla rassal olarak sunucudaki kontrol dosyasını indirilmektedir. İndirilen dosyada güncellenecek dosyaların her biri için ayrı ayrı 120000 mili saniyelik indirme bekleme süresi tanımlanmakdır. Eğer belirlenen sürede dosya indirilemez ise kuyruktaki diğer uygulamaların indirmeleri gerçekleştirilmektedir. Tüm kuyruk bitirildikten sonra indirilemeyen dosyalar için yeniden beklemeye geçilmekte ve tekrar sunucuya istek atılmaktadır. Her istek arası minimum 2 saniye bekleme süresi vardır. Bu sırada gelen veri ile beklenen veri arasında karmaşa yaşanmaması için sistem indirme linkleri üzerinden haritalama yaparak indirme dizisini ayrıştırılmış şekilde takip etmektedir. Tüm güncellenmesi gereken uygulama dosyaları indirildikten sonra sistem uyumunun sağlanması için ana kontrol dosyası olan kontrol dosyası güncellenmektedir (Şekil 6).

Ardından gelen bir sonraki sistem tetiklemesinde artık indirilmiş olan dosyalar düzgün şekilde istenen yerlere açılmalıdır. Bunun için güncelleme aşamasına geçilmektedir. Eğer istenen güncelleme acil ise sistem hiç beklemeden güncellenmeye alınmakta eğer aciliyet yok ise benzin dispanserlerinin en az kullanıldığı saatler olan 01:00 ile 03:00 aralığında yine rassal olarak belirlenen bir zaman dilimi içerisinde sistem

kendini güncellemeye geçmektedir. Bu süre içerisinde tüm indirme ve tetiklenme işlemleri durdurulmakta ve sadece güncelleme için beklenmektedir. Uygun vakte gelindiğinde yeni versiyona sahip uygulamalar tar sıkıştırma sistemine göre açılmaktadır. Bunun nedeni tar sisteminin içeriği değişmiş olan dosyaları kontrol ederek eğer bir değişim var ise yenisiyle değiştirmesi yok ise dosyaları olduğu gibi bırakmasıdır. Çalıştırılabilir uygulamalarda, dosyaların güncellenmesinde sıkıştırma veya açma işlemi yapılmadan dosya doğrudan eski dosyanın üzerine yazılmaktadır. İndirilen dosyalar gerekli yerlere yerleştirildikten sonra son olarak cihaz yeniden başlatılmakta ve sistem yeni haliyle çalışır şekle gelmektedir.

Şekil 6. Çalışma Prensibi

4.3. Test Ortamı

Çalışma 1000 adet özdeş ECU'lar üzerinde test edilmiştir. Bu cihazların donanım özellikleri aşağıdaki gibidir:

- Graperain firmasının Samsung S5P4418 Single Board Computer cihazlarıdır.
- Çekirdek: 4 Çekirdekli ARM Cortex-A9
- Hız: 1.4GHz * 4
- Bellek: 1GB DDR3
- Hafıza: 4GB EMMC
- 24bit RGB Arayüz;
- 8bit LVDS Arayüz;

- VGA Arayüz Desteği
- MIPI DSI Arayüz
- 2 Kanallı USB HOST Arayüz
- USB OTG Arayüz
- 2 Kanallı TTL Elektriksel UART Arayüz
- 2 Kanallı TF Kart Arayüz
- 2 Kanallı LED
- Reset Butonu
- Yazılımsal güç aç/kapa Anahtar
- Harici Hoparlör
- MIC Girişi
- Kulaklık Çıkış Arayüzü
- HDMI Arayüzü
- USB arabirimi WIFI/BT ikisi bir arada modül
- SPI, I2C, UART vb.
- 2D, 3D yüksek performanslı grafik hızlandırma
- Gigabit Ethernet RTL8211E
- BT656/BT601/MIPI kamera arayüzü
- GPS
- GPRS
- USB 3G modülü

Denenen bu cihazlar içerisinde 750 tanesi Suudi Arabistan'da, 250 tanesi Türkiye'dedir. Bulundukları istasyonların lokasyonları değişkenlik göstermekle beraber genel olarak şehir merkezleri, şehirler arası yollar, köyler, taşradaki ekin arazi yakınlarıdır.

5. SONUÇLAR VE ÖNERİLER

Dağıtık sistemlerde güncelleme uygulamalarına yeni bir yapı sunulmuştur. Özellikle dijitalleşen benzin istasyonları ve akıllı şarj istasyonlarının hem dijital pompa hem de benzin dispanserleri için uzaktan ağ üzerinden güncellenmesine yönelik bir çalışma sunulmuştur. Yapılan çalışma ve testler sonucunda güncelleme uygulamasının aynı anda birden fazla farklı uygulamayı başarı ile güncelleyebildiği, esnek ve geliştirilebilir bağımsız bir dağıtık güncelleyici sunduğu görülmüştür.

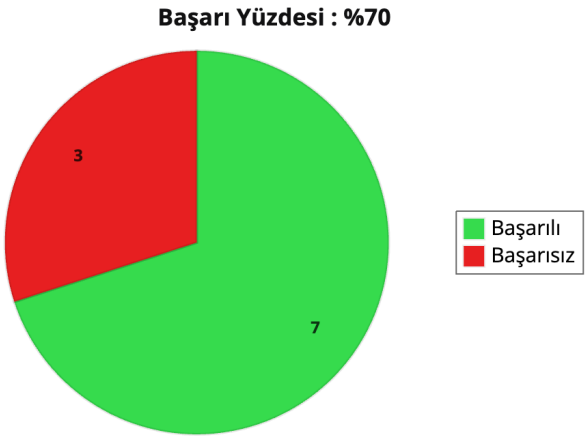
Sistemde, içerisinde birbirinden bağımsız uygulamaların güncellenmesinde oluşabilecek muhtemel uyum problemlerine anında yeni sürüm güncellemesi veya eski sürüme geri dönülmesi gibi bir imkân sunulmasıyla yeniden düzenlenebilir bir uygulama sunulmuştur. Güncelleme uygulamasının bağımsız bir uygulama olarak farklı uygulamalar ile uyumlu çalışabilmesi, sisteme müdahale edebilme esnekliğini güncel olan uygulamalara göre çok daha başarılı bir şekilde gerçekleştirebilmiştir.

Gelecekte uygulamanın internet hızının çok düşük olduğu yerlerde, en yüksek sinyal alınan cihazı güncelleyerek aynı istasyondaki diğer cihazların da onun üzerinden güncellenebileceği üzerinde durularak, bu noktada yeni çalışmalar gerçekleştirilmesi öngörülmüştür.

Çalışma kritik uygulamalar için ise TCP soket üzerinden gerekli uygulamayla konuşarak en uygun zamanda güncellenebilir bir yapı ile farklı versiyonların da oluşturulabileceği düşünülmektedir. Daha sonraki çalışmalarda bu alanda ayrı bir faaliyet gerçekleştirilecektir.

5.1. Maksimum Hata Yüzdesi

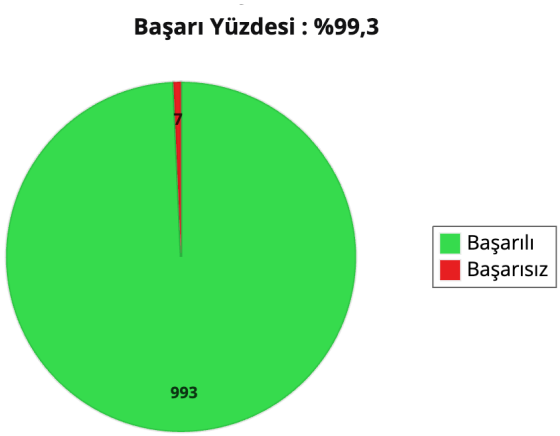
Arabistan ve Türkiye istasyonlarında bulunan 1000 cihazlık havuzdan, 20 kere rastgele seçilen 10'lu cihaz kümesi içerisinden en başarısız olanı, 10 cihazın 7 sinin başarılı bir şekilde güncellenen 3 tanesinin ise başarısız sonuçlanan cihazları oluşturduğu %70 başarı oranlı dilim olmuştur (Şekil 7).



Şekil 7. Maksimum Hata Yüzdesi

5.2. Genel Başarı Yüzdesi

Arabistan ve Türkiye istasyonlarında bulunan 1000 cihaz içinde farklı sebeplerle güncellenemeyen 7 adet cihaz olmuştur. Bu da güncelleme başarısını %99,3 olarak ortaya koymuştur (Şekil 8).



Şekil 8. Genel Başarı Yüzdesi

5.3. Çalışmadan Çıkarılan Sonuçlar ve Öneriler

Çalışma kapsamında yapılan gözlem ve test işlemleri sonucunda alınan sonuç verileri için test edilen 1000 özdeş cihaz için (Şekil 9):

Suudi Arabistan'da olan 750 cihazdan;

28'i yıkama makinası

612 tanesi dijital pompa

35 tanesi hava makinası

75 tanesi dispanser cihazıdır.

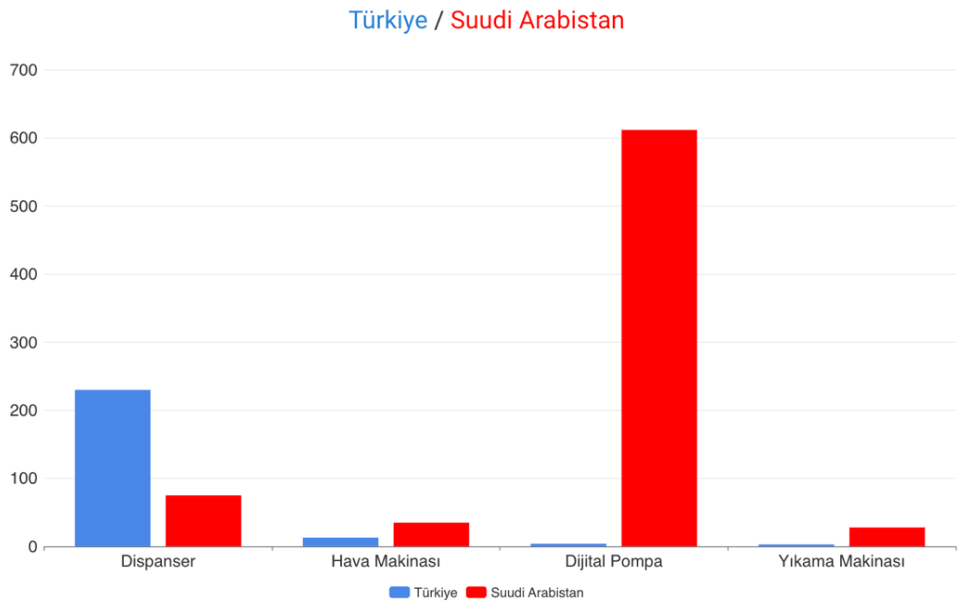
Türkiye'de bulunan 250 cihazdan;

230 tanesi dispanser

13 tanesi hava makinası

4 tanesi dijital pompa

3 tanesi yıkama makinasıdır.



Şekil 9. Ülkelere Göre Cihaz Dağılımları

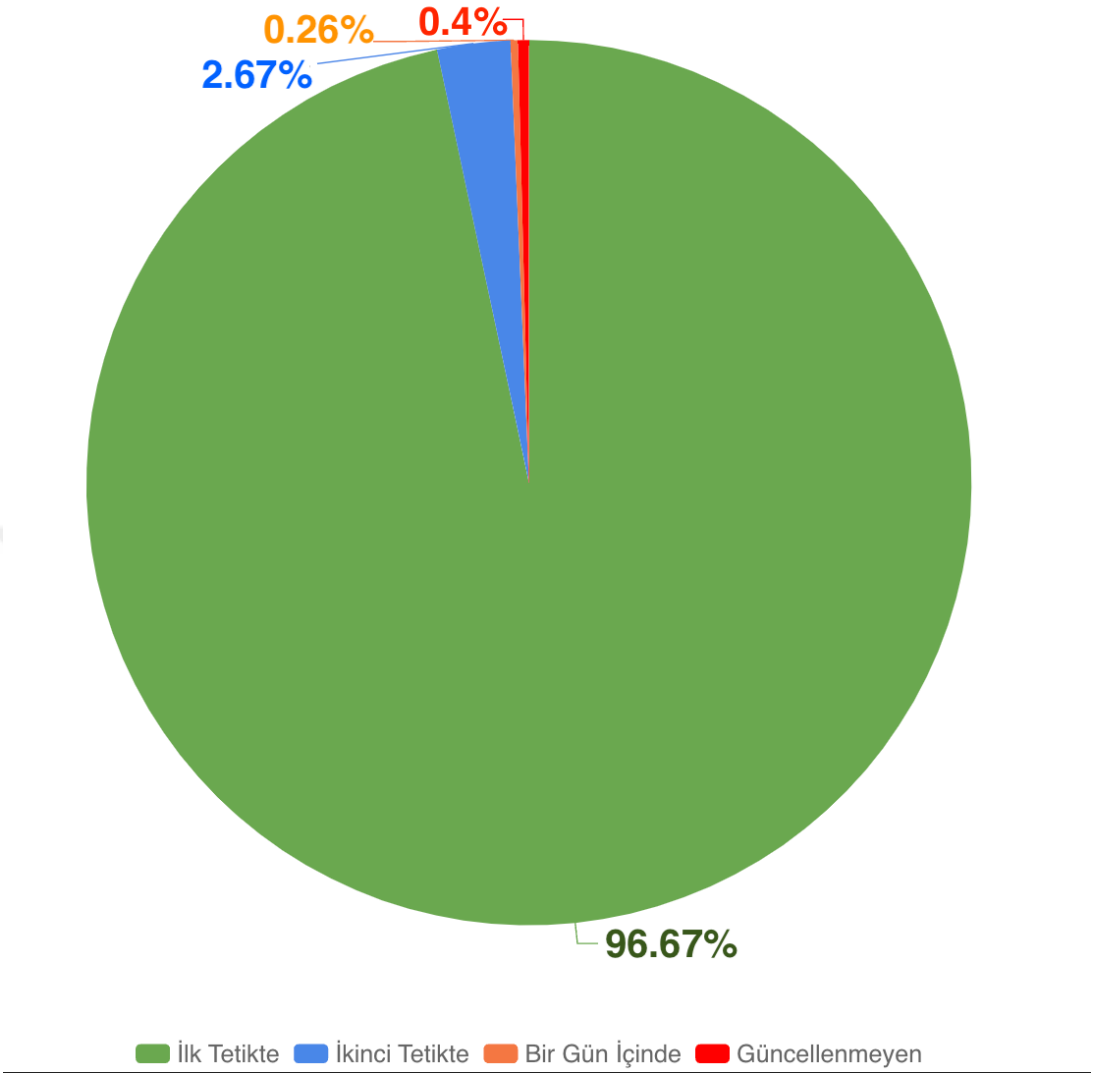
Arabistan'daki 750 cihazdan (Şekil 10)

725'i ilk tetiklenme de güncellenmiştir.

20 tanesi 2. tetiklenmede güncellenmiştir.

2 tanesi 3 ve üzeri tetiklenmede, en geç 1 gün içerisinde güncellenmiştir.

3 tanesi güncellenememiştir.



Şekil 10. Suudi Arabistan Detaylı Güncelleme Başarısı

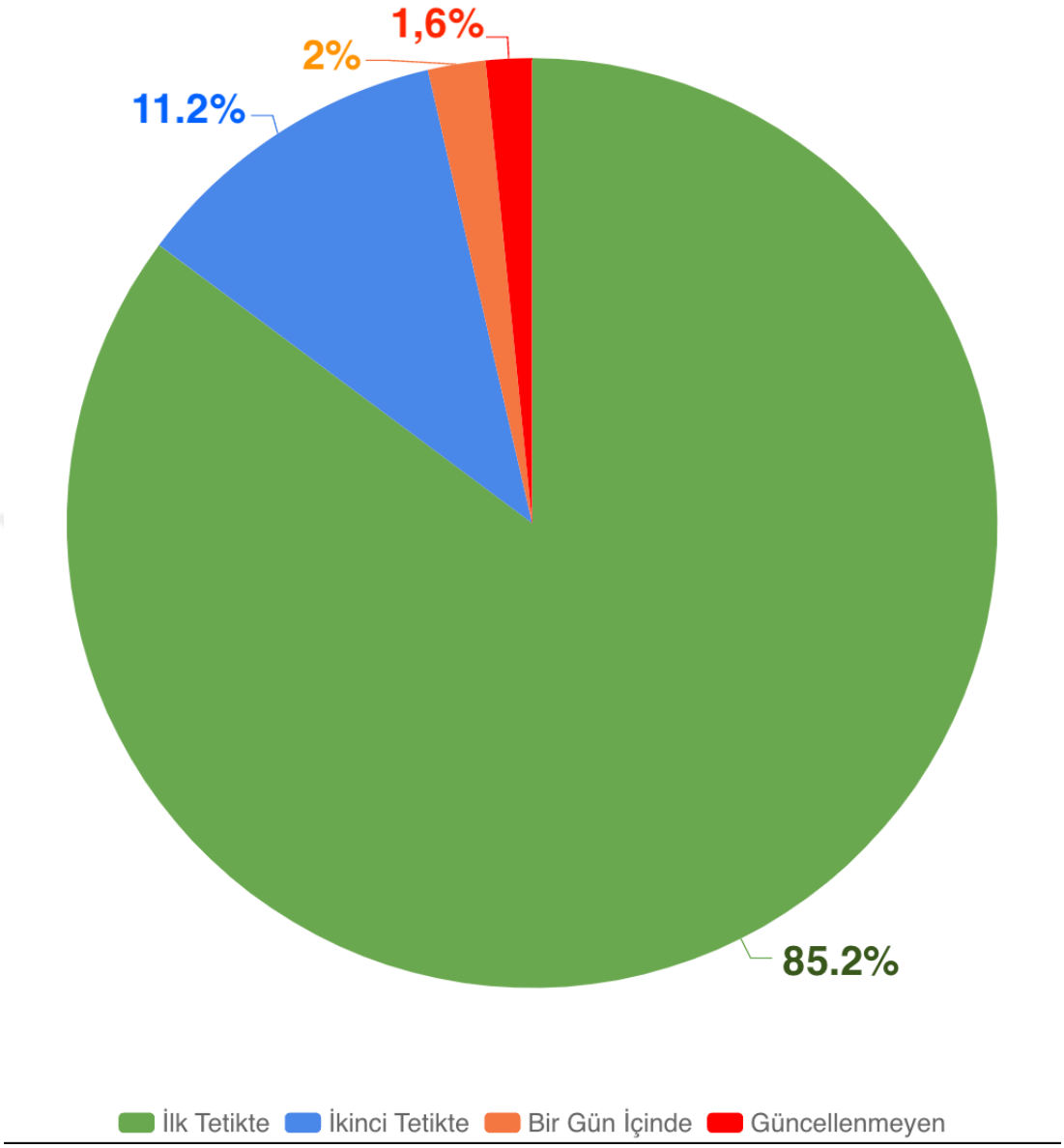
Türkiye'deki 250 cihazdan (Şekil 11)

213 tanesi ilk tetiklenmede güncellenmiştir.

28 tanesi ikinci tetiklenmede güncellenmiştir.

5 tanesi 3 ve üzeri tetiklenmede, en geç 1 gün içerisinde güncellenmiştir.

4 tanesi güncellenememiştir.



Şekil 11. Türkiye Detaylı Güncelleme Başarısı

Arabistan'daki güncellenemeyen 3 cihazdan 2'sinin internet hızlarının 2 kbps hiç bir zaman geçmediği ve bu cihazlarda çok sık kopmalar yaşandığı gözlemlenmiştir.

Türkiye'deki güncellenemeyen 4 cihazdan 2'sinin internet hızlarının 2 kbps hiç bir zaman geçmediği ve bu cihazlarda çok sık kopmalar yaşandığı gözlemlenmiştir.

Arabistan'da 1, Türkiye'de 2 cihazla, hiç bir şekilde bağlantı kurulamamış olup yetkili teknik personel tarafından sahaya gidilerek kontrol yapması istenmiştir.

6. TARTIŞMA

İlk temel bileşenlerinin veri tabanı sistemleri üzerine geliştirildiği çevrimiçi güncelleme önerileri, bugün her elektronik bileşen için ihtiyaç haline gelmiştir. Büyük veri ve veri ambarı kavramlarıyla dağıtık manada güncelleme önerileri üzerine çok güvenli ve başarılı çalışmalar gerçekleştirilmiş. Yapılan yeni çalışmalarla bu öneriler daha da ileri taşınmaktadır. Muadil ihtiyaçların IoT cihazlarında ihtiyaç haline gelmesi, özellikle akıllı telefon işletim sistemlerinde hızlı bir gelişme göstermiştir. Günümüzde tüm dijital ürünlerin akıllanmaya gitmesiyle gömülü sistem cihazlarında da bu ihtiyaç oluşmuştur.

Bu ihtiyacı karşılamak için daha önceden yapılan birçok çalışmada, uygulamanın kendi içerisinden kendisini hatasız ve hızlı bir şekilde güncellemesine odaklanmıştır. Bu da ticari firmaların genel bir literatür takip etmek yerine kendi içlerinde uygulama özelinde güncelleme şekli veya yerinden giderek güncellenmesi şeklinde süregelmiştir.

Bu çalışma özellikle elektrikli araçlarla artan çevrimiçi güncelleme ve eş zamanlı olarak açık internet dünyası içerisinde son durumlarını herkesle paylaşması gereken şarj istasyonlarını ve beraberinde yeniden ivme kazanan akaryakıt istasyonlarını hedef alarak yeni bir öneri sunmuştur. Bu çalışma; daha önceki yapılan çalışmalarla karşılaştırıldığında esneklik, ölçeklenebilirlik, kolay uyumluluk ve başarı yüzdesi yüksek sonuçlar vermesi sebebiyle ticari ve birbirinden tamamen bağımsız sektörlerin ihtiyacını karşılayabilir hale getirilmiştir. Yapılan testler ve kullanıcıların yorumları analiz edildiğinde, başarı oranı yüksek, cihaz ve içinde çalışan uygulamalardan tamamen bağımsız olması yönüyle memnun eden sonuçlar verdiği gözlemlenmiştir.

KAYNAKLAR

Ajmani, S. (2004). *Automatic Software Upgrades for Distributed Systems.* Boston: Massachusetts Institute of Technology.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M., (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials,* 2347-2376

Bayram, Ş., & Bayhan, S. (2020). Location Analysis of Electric Vehicle Charging Stations for Maximum Capacity and Coverage. *14th International Conference on Compatibility, Power Electronics and Power Engineering (CPE-POWERENG),* Setubal, Portugal.

Bloom, T. (2001). Dynamic Module Replacement in a Distributed Programming System. *Boston: Massachusetts Institute of Technology*.

Bloom, T., & Day, M. (1992). Reconfiguration in Argus. *IEE* (s. 176-187). London: Proc. *IEE Int'l Workshop on Configurable Distributed Systems*.

Boulisa, A., Hanb, C.-C., Sheab, R., & Srivastavab, M. B. (2007, Ağustos). SensorWare: Programming sensor networks beyond code update and querying. *Pervasive and Mobile Computing*, s. 386-412.

Choi, A. (2009). Online Application Upgrade Using Edition-Based Redefinition. *Proceedings of the 2nd ACM Workshop on Hot Topics in Software Upgrades* (s. 1–5). Orlando: HotSWUp.

Dajsuren, Y., & Brand, M. v. (2019). Automotive Systems and Software Engineering: State of the Art and Future Trends. *Springer.*

Derawi, M., Dalveren, Y. & Cheikh, F. A. (2020). *IEEE 6th World Forum on Internet of Things (WF-IoT)*.

Douglas, S. (1997). Adaptive filters employing partial updates. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 209-216.

Dumitraş, T., Narasimhan, P., & Tilevich, E. (2010). To Upgrade or Not to Upgrade: Impact of Online Upgrades across Multiple Administrative Domains. *Proceedings of the ACM international conference on Object oriented programming systems languages and applications* (s. 865–876). Reno/Tahoe: OOPSLA.

Faizal, M., Feng, S. Y., Zureel, M. F., Sinidol, B. E., Wong, D., & Jian, G.K. (2019). A Review on Challenges and Opportunities of Electric Vehicles (EVS). *Journal of Mechanical Engineering Research and Developments*.

Godavarti, M., & Hero, A. O. (2005). Partial Update LMS Algorithms. *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, 2382–2399.

Gupta, A. K., & Johari, R. (2019). IOT based Electrical Device Surveillance and Control System. *4th International Conference on Internet of Things: Smart Innovation and Usages*.

Huang, J., Li, T., Yuan, Y., & Li, S. (2021). M-GBDT2NN: A more generalized framework of GBDT2NN for online update. *Ad Hoc Networks*.

Hyeaong, J. H., Choi, K. J., Lee, J. Y., & Pyo, T.-H. (2020). For whom does a game update? Players' status-contingent gameplay on online games before and after an update. *Decision Support Systems*.

Li, S.E., Zheng, Y., Li, K., Wang, L., &Zhang, H. (2017). Platoon Control of Connected Vehicles from a Networked Control Perspective: Literature Review, Component Modeling, and Controller Synthesis. *Transactions on Vehicular Technology*, 1-1.

Magee, J., Kramer, J., & Sloman, M. (1989). Constructing Distributed Systems in Conic. *IEEE Transactions on Software Engineering*, 663 - 675.

Mayyas, K. (2005). Performance Analysis of the Deficient Length LMS Adaptive Algorithm. *IEEE Transactions on Signal Processing*, 2727–2734.

Mishra, Ajay R. (2007). Advanced Cellular Network Planning and Optimisation. *Nokia Networks.*

Ophir, F., & Segal, M. E. (1991). On dynamically updating a computer program: From concept to prototype. *Journal of Systems and Software*, 111-128.

Schmitt, T., Schmidt, P., Kaemmer, N., Gerhold S., & Schulthess, P. (2010). Adding Multiprocessor Support to an Uniprocessor Distributed Operating System with Transactional Distributed Memory. *2010 Second International Conference on Computer Engineering and Applications*.

Pham, T. N., Tsai, M., Nguyen, D. B., Dow, C., & Deng, D. (2015). A Cloud-Based Smart-Parking System Based on Internet-of-Things Technologies. *IEEE Access*, Vol. 3, 1581-1591.

Placho, T., Schmittner, C., Bonitz, A., & Wana, O. (2020, 8 27). Management of automotive software updates. *Microprocessors and Microsystems*.

Priyatham, J. (2019). *ADAS & Electric Vehicles Accelerate Demand of Automotive Electronic Control Units.* IHS Markit.

Qinga, Z., Nia, J., Lia, Z., & Chenb, J. (2021). Selective partial-update augmented complex-valued LMS algorithm and its performance analysis. *Signal Processing*.

Salvi, S., Mitra, A., Devlekar, S., Choudhary, A., Patil, S., & Bhosale, S. (2022). Novel ECU Design Architecture using DSP and FPGA for Electric Vehicle. *IEEE 10th Power India International Conference (PIICON)*.

Segal, M. E., & Frieder, O. (1989). Dynamic Program Updating: A Software Maintenance Technique for Minimizing Software Downtime. *Software Maintenance:Research And Practice*, 59-79.

Segel, M. E., & Frieder, O. (1989). Dynamically Updating Distributed Software: Supporting Change in Uncertain and Mistrustful Environments. *Proceedings. Conference on Software Maintenance* (s. 254-261). Los Alamitos: EEE CS Press.

Sun, D., Chen, S., Li, G., Zhang, Y., & Atıf, M. (2019). Multi-objective Optimisation of Online Distributed Software Update for DevOps in Clouds. *ACM Transactions on Internet Technology*, 1-20.

Tang, F., Chen, X., Zhao, M. & Kato, N. (2022). The Roadmap of Communication and Networking in 6G for the Metaverse. *IEEE Wireless Communications*, 1-15.

Xenya, M. C., & Quist-Aphetsi, K. (2019). A Cryptographic Technique for Authentication and Validation of Forensic Account Audit Using SHA256. *2019 International Conference on Cyber Security and Internet of Things (ICSIoT)*.

Yang, F., Huang, J., Bhardwaj, A. El-Latif, A. A. Yu, K. & Hussain, A. (2023). Adaptive Modulation based on Nondata-Aided Error Vector Magnitude for Smart Systems in Smart Cities. *IEEE Internet of Things Journal*, 1-1.