

**T.C.  
SAKARYA ÜNİVERSİTESİ  
FEN BİLİMLERİ ENSTİTÜSÜ**

**AFET DURUMLARI İÇİN AKILLI TELEFONLARDA  
HÜCRESEL SİSTEMLERE ALTERNATİF  
HABERLEŞME SİSTEMİ GELİŞTİRİLMESİ**

**YÜKSEK LİSANS TEZİ**

**Yusuf ÇERİ**

**Enstitü Anabilim Dalı : BİLGİSAYAR VE BİLİŞİM  
MÜHENDİSLİĞİ**

**Tez Danışmanı : Yrd. Doç. Dr. Ali GÜLBAĞ**

**Haziran 2014**

T.C.  
SAKARYA ÜNİVERSİTESİ  
FEN BİLİMLERİ ENSTİTÜSÜ

**AFET DURUMLARI İÇİN AKILLI TELEFONLARDA  
HÜCRESEL SİSTEMLERE ALTERNATİF  
HABERLEŞME SİSTEMİ GELİŞTİRİLMESİ**


**YÜKSEK LİSANS TEZİ**


**Yusuf ÇERİ**

Enstitü Anabilim Dalı : **BİLGİSAYAR VE BİLİŞİM  
MÜHENDİSLİĞİ**

Bu tez 18/06/2014 tarihinde aşağıdaki jüri tarafından Oybirliği ile kabul edilmiştir.

  
Doç Dr. İbrahim ÖZÇELİK  
Jüri Başkanı

  
Yrd. Doç. Dr. Ali GÜLBAĞ  
Üye

  
Yrd. Doç. Dr. İrfan YAZICI  
Üye

## **TEŐEKKÜR**

Tez alıőmam boyunca bana destek veren kıymetli hocam Yrd. Do. Dr. Ali GÜLBAĐ'a teőekkür ederim. Yüksek lisans eđitimim sırasında benden desteđini esirgemeyen deđerli iő ve yüksek lisans arkadaőım Orhan ÜÇTEPE'ye teőekkürü bir bor bilirim. Yaptıđım alıőmaya, bilgi birikimleri ile katkıda bulunan Ayhan YENİ'ye, İbrahim HÖKELEK'e teőekkür ederim. Eđitim hayatım boyunca her zaman yanımda olan ve her türlü desteđi sađlayan aileme, tez alıőmama baőladıđımdan beri beni maddi ve manevi olarak destekleyen sevgili eőim Belkıs ÇERİ'ye tüm kalbimle teőekkür ederim.

## İÇİNDEKİLER

TEŞEKKÜR.....	ii
SİMGELER VE KISALTMALAR LİSTESİ.....	v
ŞEKİLLER LİSTESİ.....	vii
TABLolar LİSTESİ.....	x
ÖZET.....	xi
SUMMARY.....	xii
BÖLÜM 1.	
GİRİŞ.....	1
BÖLÜM 2.	
AD-HOC AĞLAR.....	6
2.1. Gezgin Ad-hoc Ağlar (MANET) .....	8
2.2. Yönlendirme Protokolleri .....	8
2.2.1. OLSR (Optimized Link State Routing) protokolü.....	10
2.2.2. AODV (Ad-Hoc On-Demand Distance vector R.) protokolü .....	14
2.2.3. DSR (Dynamic Source Routing) protokolü.....	14
BÖLÜM 3.	
ANDROİD İŞLETİM SİSTEMİ VE AD-HOC MODU.....	16
3.1. Android İşletim Sistemi .....	16
3.2. Android İşletim sistemi Mimarisi .....	18
3.3. Android Haberleşme Alt Yapısı ve Ad-hoc Modu .....	20
BÖLÜM 4.	
AKILLI CİHAZLAR İLE ALTERNATİF HABERLEŞME.....	22
4.1. Android Cihazların Ad-hoc Modu Aktifleştirilmesi .....	22

4.1.1. HTC Desire S ile yapılan çalışma.....	22
4.1.2. ASUS Nexus 7 (flo) ile yapılan çalışma.....	28
4.2. Windows/Linux Bas Konuş Yazılımı.....	29
4.2.1. Yazılım tasarımı .....	31
4.3. Android Bas Konuş Uygulaması .....	34
4.3.1. Yazılım tasarımı .....	37
BÖLÜM 5.	
SINAMA ORTAMI KURULMASI VE TEST SONUÇLARI.....	41
5.1. İki Cihaz Arasında MANET .....	44
5.2. İki Cihaz Etki Alanları Dışında MANET.....	45
5.3. İki Android Cihaz ve Dizüstü Bilgisayar Arasında MANET.....	46
5.4. Ağ Performans Testleri .....	50
BÖLÜM 6.	
SONUÇLAR VE ÖNERİLER.....	53
KAYNAKLAR.....	55
ÖZGEÇMİŞ.....	58

## SİMGELER VE KISALTMALAR LİSTESİ

ADB	: Android Debug Bridge
ADT	: Android Debug Tool
Android NDK	: Android Native Development Kit
Android SDK	: Android Software Development Kit
AODV	: Ad-Hoc on Demand Distance Vector
AOKP	: Android Open Kang Project
AOSP	: Android Open Source Project
API	: Application Programming Interface
CWM	: Clockworkmod Recovery
DSDV	: Destination-Sequenced Distance Vector
DSR	: Dynamic Source Routing
DTN	: Delay Tolerant Network
DVM	: Dalvik Virtual Machine
GPL	: General Public License
GPS	: Global Position System
GSM	: Global System For Mobile Communications
IDC	: International Data Corporation
IETF	: Internet Engineering Task Force
IP	: Internet Protocol
JDK	: Java Development Kit
JNI	: Java Native Interface
JRE	: Java Runtime Environment
LTE	: Long-Term Evolution
MANET	: Mobile Ad-Hoc Network
MPR	: Multipoint Relay
OHA	: Open Handset Alliance

OLSR	: Optimized Link Source Routing Protocol
ORWAR	: Opportunistic DTN Routing With Window–Aware Adaptive Replication
RFC	: Request Force Comments
ROM	: Read Only Memory
SMS	: Short Message Service
SPAN	: Smartphone Ad-Hoc Networks
SSID	: Service Set Identification
STAR	: Source Tree Adaptive Routing
TBRPF	: Topology Broadcast Based on Reverse Path Forwarding
TC	: Topology Control
TORA	: Temporally ordered Routing Algorithm
UDP	: User Datagram Protocol
UMTS	: Universal Mobile Telecommunications System
USB	: Universal Serial Bus
VANET	: Vehicular Ad-Hoc Network
Wi-Fi	: Wireless Fidelity
Wi-Max	: Worldwide Interoperability for Microwave Access

## ŞEKİLLER LİSTESİ

Şekil 2.1. Yapılandırılmış kablosuz ağlar .....	6
Şekil 2.2. Ad-hoc ağlar.....	7
Şekil 2.3. Yönlendirme protokolleri çeşitleri .....	9
Şekil 2.4. OLSR yönlendirme protokolü MPR seçimi.....	11
Şekil 2.5. OLSR mesaj paketi yapısı.....	13
Şekil 2.6. AODV protokolü kaynağın yaptığı yayın.....	14
Şekil 2.7. AODV protokolü hedeften dönen cevap.....	14
Şekil 2.8. DSR protokolü kaynağın ROUTE REQUEST mesajı yayını.....	15
Şekil 2.9. DSR protokolü hedeften gelen ROUTE REPLY mesajı .....	15
Şekil 3.1. IDC verilerine göre 2013 yılı mobil cihazlarda kullanılan işletim sistemi oranları.....	16
Şekil 3.2. Android işletim sistemi yığın yapısı .....	18
Şekil 3.3. Android uygulamasının derlenme süreci .....	18
Şekil 4.1. HTC Desire S .....	22
Şekil 4.2. ASUS Nexus 7 (flo).....	22
Şekil 4.3. HTC Desire S S-OFF yapılması aşamaları .....	23
Şekil 4.4. HTC Desire S Önyükleyici ekranının başlatılması.....	24
Şekil 4.5. HTC Desire S hızlı açılış ayarı .....	24
Şekil 4.6. HTC Desire S bootloader lock ekranı .....	24
Şekil 4.7. Fastboot seçilmesi .....	24
Şekil 4.8. Telefonun bilgisayara bağlanması .....	25
Şekil 4.9. Fastboot komutunun çalıştırılması .....	25
Şekil 4.10. Cihaz tanımlayıcı kodu .....	25
Şekil 4.11. Fasboot komutu ile unlock yapılması .....	26
Şekil 4.12. Unlock onay ekranı .....	26
Şekil 4.13. HTC Desire S önyükleyici ekranı .....	27
Şekil 4.14. HTC Desire S yüklenen CWM .....	28



Şekil 4.15. Nexus Tool Kit ekran görünümü .....	29
Şekil 4.16. Win/Linux bas konuş yazılımı ana ekranı .....	30
Şekil 4.17. Normal durum ekranı .....	31
Şekil 4.18. Dinleme ekranı .....	31
Şekil 4.19. Konuşma ekranı .....	31
Şekil 4.20. Win/Linux bas konuş yazılımı kanal durum makinesi .....	32
Şekil 4.21. Windows-Linux bas konuş yazılım tasarımı blok diagramı .....	33
Şekil 4.22. Android bas konuş uygulamasının işletim sistemindeki görünümü .....	35
Şekil 4.23. Android bas konuş uyg ana ekranı .....	36
Şekil 4.24. Android bas konuş uyg menüsü .....	36
Şekil 4.25. Android bas konuş uyg ağ topoloji bilgisi ekranı .....	36
Şekil 4.26. Normal durum ekranı .....	37
Şekil 4.27. Dinleme durum ekranı .....	37
Şekil 4.28. Konuşma durumu ekranı .....	37
Şekil 4.29. Android bas konuş uygulaması servis durum makinesi .....	38
Şekil 4.30. Android bas konuş uygulaması yazılım tasarımı blok diagramı .....	39
Şekil 5.1. OLSR Switch yazılımı konfigürasyon ekranı .....	42
Şekil 5.2. OLSR Switch yazılımı çalıştırılması - Bağlı uç nokta IP bilgileri .....	42
Şekil 5.3. OLSR Switch yazılımı çalıştırılması - Yönlendirici uç nokta IP bilgileri .....	43
Şekil 5.4. Test - İki cihaz arasında MANET .....	44
Şekil 5.5. Cihaz 1 bağlılık tablosu .....	44
Şekil 5.6. Cihaz 2 bağlılık tablosu .....	44
Şekil 5.7. Test - İki cihaz etki alanı dışında .....	45
Şekil 5.8. Etki alanı dışında Cihaz 1 bağlılık tablosu .....	45
Şekil 5.9. Etki Alanı dışında Cihaz 2 bağlılık tablosu .....	45
Şekil 5.10. Test - Cihaz 1, Cihaz 2 ve Cihaz 3 ile MANET ağı kurulması .....	46
Şekil 5.11. Cihaz 1 bağlılık durum bilgileri .....	46
Şekil 5.12. Cihaz 2 bağlılık durum bilgileri .....	46
Şekil 5.13. Cihaz 3 yönlendirme IP bilgileri .....	47
Şekil 5.14. Yönlendirilen paketin Wireshark görüntüsü-1 .....	48
Şekil 5.15. Yönlendirilen paketin Wireshark görüntüsü-2 .....	48
Şekil 5.16. Test - Cihaz 1 yönlendirici, MANET ağı .....	49
Şekil 5.17. Cihaz 1 Yön.-Ch1 Görüntüsü .....	49

Şekil 5.18. Cihaz 1 Yönlendirici - Cihaz 2 Görüntüsü .....	49
Şekil 5.19. Cihaz 3'e gelen paketlerin Wireshark programındaki görüntüsü .....	50
Şekil 5.20. Cihaz 1 yönlendirici durumunda test MANET ağı .....	50
Şekil 5.21. Test uygulaması ekran görünümü .....	51
Şekil 5.22. MANET ağı paket gecikme süresinin belirlenmesi .....	52

## TABLolar LİSTESİ

Tablo 1.1. Yapılan çalışmalar kapsamında geliştirilen uygulamaların özellikleri.....	4
Tablo 3.1. Android işletim sistemi versiyonları.....	17
Tablo 3.2. Android versiyonları ve kullandıkları Linux çekirdeği versiyonu.....	19
Tablo 5.1. Cihaz IP bilgileri.....	44
Tablo 5.2. Pingb yazılımı ile veri aktarım hızı ölçüm sonuçları.....	51

## ÖZET

Anahtar kelimeler: Android, MANET, OLSR, MANET Manager

Günümüzde iletişim için yaygın olarak hücreli haberleşme sistemleri kullanılmaktadır. Afet anında arızalanma veya aşırı yüklenme gibi sebeplerle haberleşme sistemleri çökebilme ve iletişim sağlanamadığı zamanlar olabilmektedir. Afet anlarında kullanılacak alternatif haberleşme alt yapısının varlığı bu gibi durumlarda önem kazanmaktadır. Bu çalışma doğal afetlerde (deprem, fırtına gibi) hücreli şebekelerin aşırı yüklenme, arızalanma gibi sebeplerle devre dışı kaldıkları durumlarda, kurtarma ekipleri ve afetzedeler arasında iletişimi sağlayacak alternatif haberleşme alt yapısını kurmayı hedeflemektedir. Bu haberleşme alt yapısı, akıllı cihazların WiFi arayüzü kullanılarak örgü ağı kurulması ile oluşturulmuştur. Akıllı cihaz olarak, maliyeti düşük ve yaygın olarak kullanılan Android işletim sistemli cep telefonları tercih edilmiştir. Android işletim sistemine sahip cihaz üzerinde çalışan “baskonuş” özelliğine sahip sesli konuşmayı sağlayan uygulama geliştirilmiştir. Örgü ağının oluşturulması için MANET Manager açık kaynak kodlu mobil ad-hoc ağı çatısı kullanılmıştır. Android cihazlar ve bilgisayar üzerinde OLSR (Optimized Link State Routing) yönlendirme protokolünü kullanan yazılım ile örgü ağı oluşturularak veri aktarım testleri gerçekleştirilmiştir.

# **ESTABLISHING AN ALTERNATIVE COMMUNICATION NETWORK IN DISASTER SITUATIONS USING SMART DEVICES**

## **SUMMARY**

Key Words: Android, MANET, OLSR, MANET Manager

Nowadays cellular communication systems are widely used for communication. Communication systems can be collapsed for reasons such as breakdown or overloading during disaster and communication isn't provided in this time period. It is important that the existence of alternative communications infrastructure that can be used in case of disaster situations. This study aims to set up an alternative communication network which enables rescue crew to communicate with each other and disaster victims in case of a cellular network breakdown or overload in a natural disaster such as earthquake or storm. The communication network is created via establishing mesh network by using Wi-fi interface of smart devices. Commonly used smart phones with Android operating systems are preferred as smart devices. A “push to talk” application is developed on Android. Open source software MANET Manager framework is used for creating mesh network. A series of tests on Android devices and computers are applied by creating mesh network with a software using OLSR (Optimized Link State Routing) routing protocol. As a result of the tests, communication among devices is succeeded.

## **BÖLÜM 1. GİRİŞ**

Günümüzde en yaygın olarak kullanılan haberleşme sistemi baz istasyonları ile kurulan hücresele ağlardır. Yapısı belirli olan bu tür ağlarda bir operatör kontrolünde baz istasyonları ve yönlendirici cihazlar sabit olarak yerleştirilir. Literatürde bu tür altyapı gerektiren ağlar “Infrastructure Based Networks” olarak isimlendirilir. Bu iletişim alt yapısının kurulum maliyeti yüksektir. Bu istasyonlar ile mobil terminaller arasında ses ve veri iletişimi gerçekleştirilir. Bir cep telefonu ile yer ve zaman kısıtlaması olmadan istenilen kişiyle haberleşme sağlanır. Akıllı Cep telefonlarının kullandığı erişim ağı ve sensör birimleri çeşitliliğinin artmasıyla yeni nesil mobil servisler desteklenebilmektedir [1].

Günümüzde Android ve iOS işletim sistemlerine sahip akıllı telefonlar yaygın olarak kullanılmaktadır. Akıllı telefonların yanında tabletlerin kullanımı hızla artmıştır. IDC 2013 Kasım ayı verilerine göre, Android işletim sistemine sahip cihazların kullanımı %81 oranla ilk sırada yer almakta bunu iOS işletim sistemine sahip cihazlar %12,9 oranıyla takip etmektedir [2]. Bu cihazlar ile GSM şebekeleri üzerinden 3G, Wi-Fi (802.11b, 802.11g, 802.11n) ve bluetooth gibi kablosuz bağlantılar gerçekleştirilebilmektedir. Üreticilerin sağlamış olduğu esnek uygulama geliştirme alt yapıları ile cihazlardaki bu iletişim protokollerini kullanmak oldukça kolaydır.

Mobil cihazların bu gelişmiş özelliklerine karşın haberleşmenin yapılamadığı zaman dilimleri de olabilir [1]. Hücresele şebekelerin aşırı yüklenme, arızalanma gibi sebeplerle devre dışı kaldıkları durumlarda, kurtarma ekipleri ve afetzedeler arasında iletişimi sağlayacak alternatif haberleşme alt yapısı kurulması güncel araştırma konularından biridir. Afet durumlarında haberleşmenin etkin bir şekilde devam etmesi hayati önem taşıdığı bilinmektedir. Felaket anında iletişim teknolojileri yardımıyla haberleşme ağının kurulması, yardıma ihtiyacı olan vatandaşlara ulaşılabilmesi ve yardım götürülebilmesi, risk yönetimi kapsamında devlet

politikaları içerisinde yer alır [3]. Kesintisiz iletişimin sağlanabilmesi için gelişen teknoloji ile birlikte yeni cihazlar ve varolan iletişim ağlarına alternatifler geliştirilmeye çalışılmaktadır [4]. Bu gibi durumlarda mobil cihazların Wi-Fi arayüzleri kullanılarak alternatif haberleşme sistemleri kurulabilir. Akıllı cihazların maliyetlerinin düşmesine paralel olarak yaygınlaşması ve bu cihazların sağladığı iletişim alt yapısı düşünüldüğünde, Android işletim sistemine sahip mobil cihazların bu alanda kullanılması gündeme gelmiştir [2].

Zamanında müdahalenin önem kazandığı afet anında, haberleşme alt yapısının hızlıca kurulup kurtarma birimleri arasında kesintisiz iletişimin sağlanması gerekir. Belli erişim noktaları üzerinden haberleşme sistemi kurmak yerine cihazların direkt birbirleriyle iletişimde olduğu MANET (mobile ad-hoc network) kurulabilir. Bu ağlar ile telefonlar birbirlerine herhangi kurulu alt yapı olmadan kablosuz olarak bağlanabilirler ve maliyeti düşük veri alış verişi sağlanabilir [5]. Ağa bağlı her bir cihaz yani uç noktalar hareket halinde olabileceğinden ve bağlantı durumları her an değişebileceğinden yapısız ağ olarak (infrastructureless network) adlandırılır. Her bir uç nokta kendisi ile ilgili olmayan ağ trafiğini kendi üzerinden diğer uç noktaya iletir yani yönlendirici gibi davranır.

Mobil ad-hoc ağlar, uzun süredir devam eden bir araştırma konusu olmasına rağmen, akıllı mobil telefonlarının bu ağlarda kullanımı ile ilgili çalışmalar oldukça nadirdir [5]. Bu konuda daha önce yapılan çalışmalar arasında BEDnet gösterilebilir. BEDnet, bluetooth teknolojisini kullanarak J2ME telefonlarda MANET kurmayı hedefleyen açık kaynak kodlu çatıdır. "Scatternet formation algorithm" ve DSDV yönlendirme protokolünü kullanmıştır. Kısa mesafeli olan bluetooth haberleşmesini geniş alana yaymayı hedeflemiştir. J2ME'nin kısıtlarından dolayı düşük hızlarda kalmıştır. Verimin artırılması için BEDnet projesinde elde edilen tecrübe Android üzerinde çalışan Beddernet geliştirilmesi için kullanılmıştır. Android üzerinde bulunan RFCOMM bluetooth API'sini ve DSDV yönlendirme protokolünü kullanan Beddernet ile veri alış veriş hızında daha iyi sonuçlar elde edilmiştir [6][7]. Bluetooth teknolojisi kısa mesafede haberleşmeyi sağladığından daha çok kısa mesafede cihazlar arasında haberleşmeyi hedeflemiştir. Örneğin ofis ortamındaki

yazıcıya MANET ortamında erişim, dosya veya metin haberleşmesi, oyun ağı kurulması Beddernet ile yapılabilecekler arasındadır.

Davide Anzaldi yaptığı yüksek lisans tezinde, DTN (Delay Tolerant Network) ağlarda ORWAR (Opportunistic DTN Routing with Window - Aware Adaptive Replication) protokolünün Android işletim sistemi üzerinde çalışması için çalışma yapmıştır. Geliştirilen bu kütüphane Android cihazların GPS alt yapısını kullanarak çalışır. Yaptığı çalışmada ayrıca kütüphaneyi kullanan mesajlaşma uygulaması geliştirmiştir. Sadece dış ortamlarda verimli çalışmaktadır. Daha sonra geliştirilebilecek uygulamalar için çatı sunmamaktadır [8][5].

Rabie jradi ve arkadaşı tarafından Android işletim sistemi üzerinde çalışacak AODV yönlendirme protokolü kütüphanesi geliştirilmiştir. Kütüphane, daha sonra geliştirilebilecek uygulama tarafından kullanılabilir özelliğine sahiptir. Projede bu kütüphaneyi kullanan basit mesajlaşma uygulaması geliştirmişlerdir [9][5].

Halen devam etmekte olan projeler "Serval Projesi" ve "SPAN" (smartphone ad-hoc networks) isimli açık kaynak kodlu projeler akıllı cihazlarda MANET kurmayı hedeflemektedir. Bu projeler, mobil telefonlar kullanarak, varolan hücresele şebekelere alternatif haberleşme sistemleri kurmayı hedeflemektedir. Shuttleworth Foundation tarafından desteklenen Serval açık kaynak kodlu projesi, bazı istasyonlarının olmadığı yerlerde veya iletişim alt yapısının çeşitli sebeplerle çöktüğü durumlarda mobil telefonlar kullanılarak haberleşmenin hızlı ve ucuz olarak sağlanmasını hedeflemektedir [10]. Serval projesinde haberleşmenin, her bir kullanıcının günlük hayatta sahip olduğu GSM hattı numarası kullanılarak yapılması öngörülmüştür. Bir telefonun Wi-Fi bağlantı mesafesi açık alanda 100 metreden fazla değildir [11]. Serval projesi geliştireceği donanımlar ve yazılımlar ile daha geniş alanda mesajlaşma, dosya transferi, sesli haberleşme sağlamayı planlamaktadır [10]. Bir diğer devam projesi olan SPAN projesi ise, OLSR yönlendirme protokolünü Android cihazlarda çalıştırarak ad-hoc ağlar kurmayı amaçlamaktadır. SPAN projesi bünyesinde geliştirilmekte olan Android MANET Manager çatısı, geliştirilebilecek uygulamalar tarafından kullanılabilir.



Doğal afetlerde (deprem, fırtına gibi) hücresel şebekelerin aşırı yüklenme, arızalanma gibi sebeplerle devre dışı kaldıkları bilinmektedir. Tez kapsamında yapılan çalışmada Android işletim sistemli cihazlar ile MANET kurularak kurtarma birimleri arasında alternatif haberleşmeyi sağlayacak bir sistemin geliştirilmesi amaçlanmaktadır. Piyasada yaygın kullanılması ve diğer akıllı cihazlara göre ucuz olması sebebiyle Android işletim sistemli cihazlar tercih edilmiştir. Standart Android cihazlar ad-hoc modunu desteklemediğinden yapılan değişiklikler ile ad-hoc ağı kurabilecek özellik kazandırılmıştır. Hücresel haberleşmenin çalışamaz hale geldiği afet ve kriz durumlarında koordinasyon birimlerinin haberleşmeleri için örnek bir bas-konuş Android uygulaması geliştirilmiştir. MANET kurulumu için SPAN projesi kapsamında geliştirilen, OLSR (optimized link source routing) yönlendirme protokolü kullanan “MANET Manager” açık kaynak kodlu çatısı kullanılmıştır. Ayrıca yapılan çalışmada yönlendirme testlerini daha fazla sayıda uç nokta ile gerçekleştirmek için OLSR yönlendirme protokolünü çalıştıran Windows ve Ubuntu işletim sistemli dizüstü bilgisayarlar ağa dahil edilmiştir. Yapılan testlerde verimin anlaşılması için Android bas konuş uygulaması ile uyumlu, Windows ve Ubuntu işletim sistemlerinde çalışan bas konuş uygulaması geliştirilmiştir.

Buraya kadar bahsedilen projeler kapsamında geliştirilen uygulamaların özellikleri Tablo 1.1’de gösterilmiştir. Sadece SPAN projesi kapsamında geliştirilen ManetPTT ve Serval projesinde sesli haberleşme mevcut olmasına karşın diğerlerinde sadece mesajlaşma uygulaması geliştirilmiştir.

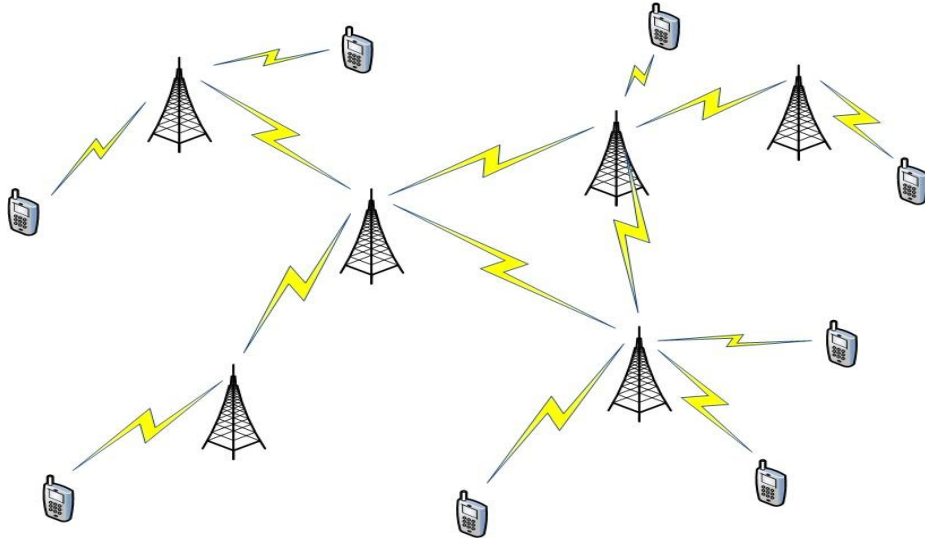
Tablo 1.1. Yapılan çalışmalar kapsamında geliştirilen uygulamaların özellikleri

Uygulama	İletişim Alt Yapısı	Yönlendirme Protokolü	Çalışma Ortamı
<b>SPAN</b>	Android Manet PTT; Ses ve mesaj uygulama Android Manet Visualizer; Haritada uç noktaların konumunu gösterir	Wi-Fi	Farklı protokoller kullanılabilir(OLSR, DSR ) Android
<b>Serval Project</b>	Serval Mesh; Ses ve mesaj uygulaması	Wi-Fi	Özelleştirilmiş BATMAN (Serval Mesh Routing) Android
<b>BEDnet</b>	Mesajlaşma	Bluetooth	DSDV J2ME
<b>Beddernet</b>	Mesajlaşma	Bluetooth	DSDV Android
<b>ORWAR</b>	Mesajlaşma	WiFi	ORWAR Android
<b>AODV Android</b>	Mesajlaşma	WiFi	AODV Android

Bu tez altı bölümden oluşmaktadır. Birinci bölüm giriş bölümüdür. Çalışmanın amacı ve daha önce konuyla ilgili yapılan çalışmalardan bahsedilmiştir. İkinci bölümde ad-hoc ağlar hakkında bilgi verilmiştir. Mobil ad-hoc ağları çeşitleri ve bu ağlarda kullanılan yönlendirme protokolleri anlatılmıştır. Üçüncü bölümde Android işletim sistemi özellikleri ve Android cihazların ad-hoc desteği üzerinde durulmuştur. Dördüncü bölümde tez kapsamında yapılan çalışma anlatılmıştır. Android cihazların ad-hoc modu aktif hale getirilmesi için yapılması gerekenler, geliştirilen Android bas konuş uygulaması, Windows ve Ubuntu işletim sistemli bilgisayarlar için geliştirilen bas konuş yazılımı detayları ile açıklanmıştır. Beşinci bölümde Android cihazlar ve dizüstü bilgisayarlar ile MANET kurulması ve geliştirilen yazılımlar ile yapılan testler anlatılmıştır. Altıncı bölümde sonuçlar üzerinde durulmuştur.

## BÖLÜM 2. AD-HOC AĞLAR

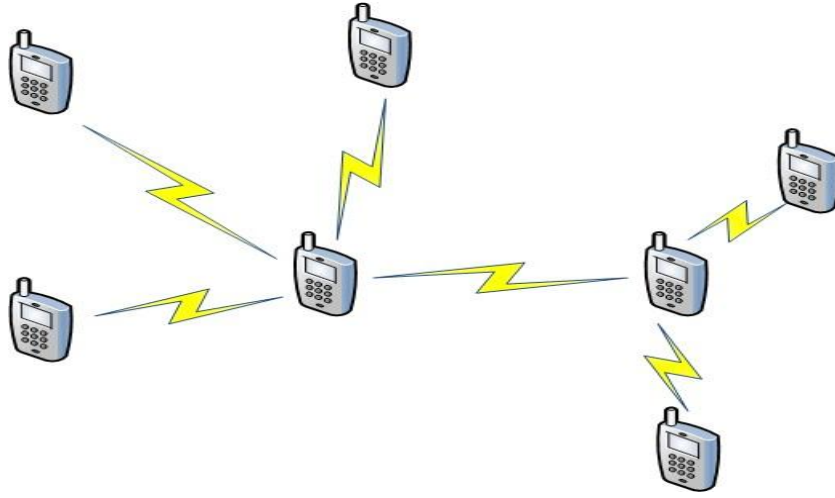
Kablosuz ağlar yapılandırılmış (infrastructure) ve tasarsız (ad-hoc) ağlar olarak iki modda oluşturulabilir. Yapılandırılmış kablosuz ağlara örnek olarak gösterilen "Group Standard for Mobile communications" (GSM), UMTS, LTE, WiMAX gibi popüler çözümlerde gezgin uç noktaları sabit erişim noktaları ile iletişim kurarak haberleşme sağlar. Şekil 2.1'de yapılandırılmış kablosuz ağların genel görünümü gösterilmiştir.



Şekil 2.1. Yapılandırılmış kablosuz ağlar

Ad-hoc ağlarda ise uç noktalar, sabit kurulu alt yapı gerektirmeden ve herhangi bir dağıtıcı olmadan haberleşebilirler. Uç noktalar direkt erişim mesafesinde değilse, ağdaki diğer uç noktalar üzerinden haberleşme kururlar [12]. Cihazların kısa sürede minimum konfigürasyon ile oluşturdukları kablosuz ağlara ad-hoc ağ denir. Bu ağlar rasgele ve anlık olarak oluşturulur. Örnek bir ad-hoc ağı Şekil 2.2'de gösterilmiştir. görüldüğü gibi sabit dağıtıcı yoktur ve uç noktalar birbirleri üzerinden veri alışverişini sağlayabilmektedirler.

Ad-hoc ağlar sabit bir alt yapı kullanmadan kendiliğinden yapılanarak geniş bir alana kolayca yayılırlar. Ad-hoc ağlardaki düğümler sabit olabildikleri gibi, çok yüksek hızla da hareket edebilirler [13]. Sabit altyapılı kablosuz ağlar ile kıyaslandığında, kullanıcılara daha fazla esneklik, hareketlilik sağlar ve ağ yönetimini kolaylaştırır. Ad-hoc ağın tasarımı sabit alt yapıya göre farklıdır, telsiz ortamda veri gönderimi, güç tüketimi ve ağ ölçeklenebilirliği (scalability) karşılaşılan problemlerdir [13].



Şekil 2.2. Ad-hoc ağlar

Ad-hoc ağlarla ilgili temel özellikler şu şekilde sıralanabilir [13]:

1. Ağın denetimi, yönetimi, topolojinin belirlenmesi için merkezi bir otorite yoktur.
2. Tüm ağ elemanları hareket halinde olabilir.
3. Band genişliği ve enerji ad-hoc ağlarda kısıtlıdır.
4. Uç düğümlerden herbiri gerektiğinde yol atama gibi fonksiyonları da yerine getirirler.

Ad-hoc ağlar farklı amaç için kullanılabilir [13]:

1. Sel, deprem gibi doğal afetler nedeniyle mevcut iletişim altyapısının çalışmadığı zamanlarda

2. Kurtarma görevlerinde, sabit altyapılı ağların kapsama alanlarının dışında (denizde, dağda)
3. Askeri (taktik) iletişimde, daha hızlı kurulabilen bir iletişim altyapısı sağlamak için
4. Sergilerde, konferanslarda iletişimi sağlamak için

Ad-hoc ağların farklı tipleri vardır;

1. Gezin Ad-hoc Ağlar (MANET); İki tip vardır,
  - a. VANET (Vehicular ad-hoc Network), genellikle askeri alanda kullanılır, hareket halinde olan araçlar ile oluşturulur.
  - b. Akıllı araç ad-hoc ağları (Intelligent VANETs), araçlar arası ve araçlar ile sabit istasyonlar arası haberleşme kurar.
2. Kablosuz duyarga ağları (Wireless sensor networks), basınç, nem, sıcaklık gibi farklı duyarga tiplerinden veri toplamak için kullanılan ağ çeşididir.
3. Kablosuz örgü ağları (Wireless mesh networks), geniş bölgelerde iletişimin kablosuz olarak yapılması için kurulan ad-hoc ağlardır. 802.11s ile standartlaştırılmaktadır.

### **2.1. Gezin Ad-hoc Ağlar (MANET)**

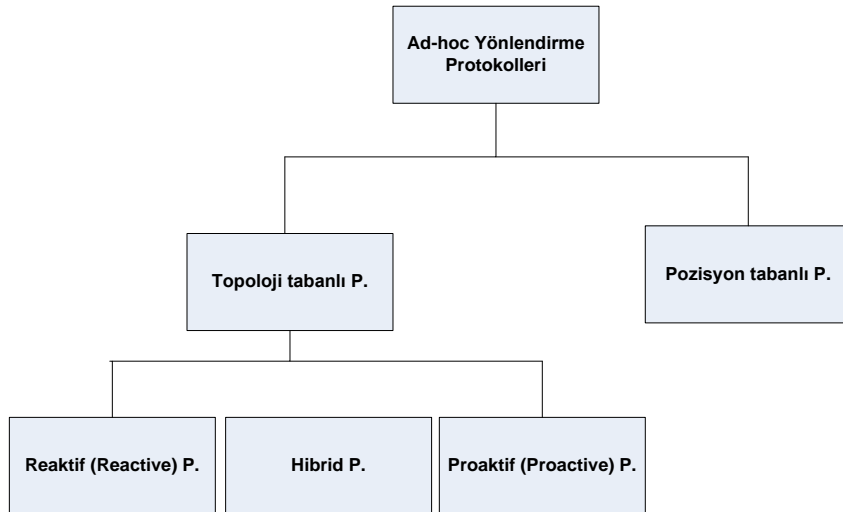
Ad-hoc ağların uygulamaları olarak ortaya çıkarlar. MANET uygulamaları, güç kaynakları sınırlı küçük ağlardan, büyük çapta gezgin dinamik ağlara çeşitlilik arz eder [14]. MANET kurulumu hem ucuz hem hızlıdır. Multi-Hop iletişimin gerçekleştirilebilmesi için tüm uç noktalar birbirleri için yönlendirici gibi davranmalıdır. Uç noktalar arasındaki yollar yönlendirme protokolü ile kurulur ve devamlılığı sağlanır.

### **2.2. Yönlendirme Protokolleri**

Ad-hoc ağlarda, ağ içindeki yolların oluşturulması ve uç noktalar arasındaki iletişimin kesintisiz devam etmesi her bir uç nokta üzerinde çalışan yönlendirme protokolleri ile sağlanır. Ad-hoc ağlarda ortaya çıkabilecek problemleri çözmeye

yönelik farklı yönlendirme protokolleri geliştirilmiştir. Şekil 2.3'de yönlendirme protokolü çeşitleri gösterilmiştir. Burada reaktif ve proaktif protokol çeşitleri üzerinde durulacaktır.

Reaktif (reactive) protokollerde ihtiyaç duyulana kadar yönlendirme protokolü başlatılmaz. Gerektiğinde ağdaki yollar ağa gönderilen mesajlar ile bulunur [15]. Yolların bulunmasında yönlendirilmiş yayın (directed broadcast) denilen kontrollü taşkın yöntemi kullanılarak sorgu-cevap işlemleri yapılır. Kaynak düğümden hedef düğüme sorgu paketleri birkaç yol üzerinden gidebilir, bu durum hedef düğüme bir veya birkaç yolu dinamik olarak oluşturur. Bu metod ile yol bilgilerinin saklanacağı varış düğümlerinin sayısı azaltılarak yol atama için gerekli olan denetim trafiği küçültülmüş olur. Böylece uç noktadaki tabloların büyüklüğü ve ağ trafiği azalır [13]. Bu tür protokollerin olumsuz yanları şu şekilde sıralanabilir; yol bulma sırasında gecikme olabilir. Sorgular tüm ağa gönderildiğinden uzaktaki düğümlere erişimde denetim trafiği büyüktür ve yol kalitesi düşüktür. Çoğu zaman en iyi yol bulunamaz ve topolojinin her değişiminde yol kalitesi düşer [13]. AODV, DSR, TORA bu protokollere örnektir [15].



Şekil 2.3. Yönlendirme protokolleri çeşitleri

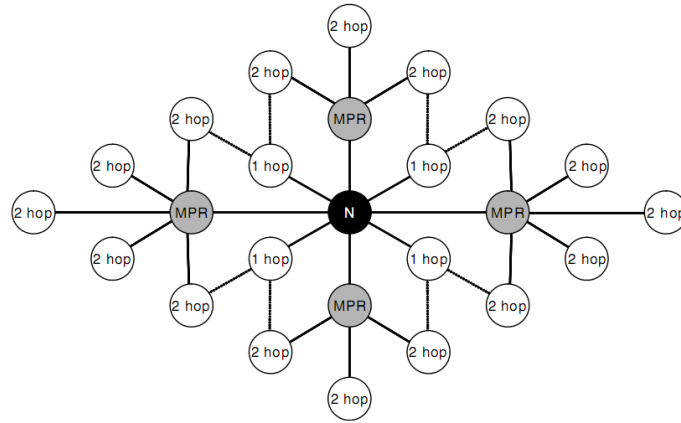
Proaktif (proactive) protokoller belirli zaman aralıklarında çalışan kontrol mesajlarının kullanılmasına dayanır. Bazı mesajlar bir uç noktanın komşularını bulması için, bazıları da tüm ağın topolojisinin bulunması için kullanılır. Bir paket

hedef uç noktaya iletilmek istendiğinde gitmesi gereken yol bellidir, paketin ulaştırılmasında gecikme yoktur. Ağın topolojisinin güncel tutulması için periyodik olarak ağa mesajlar gönderildiğinden kullanılan bant genişliği fazladır. Her bir uç nokta üzerinde sürekli olarak güncellenen yol tabloları mevcuttur. Örnek olarak OLSR, DSDV, STAR, TBRPF verilebilir [15]. Hibrid tabanlı protokoller bu iki yöntemin kullanılmasıyla ortaya çıkar.

Reaktif ve Proaktif protokollere genel olarak bakıldığında, Reaktif protokoller istekle tetiklenir, Proaktif protokoller tablolara dayanır. Her ikisi de bazı koşullarda verimli çalışırken diğer koşullarda verimli çalışmamaktadır. İstekle tetiklenen yönlendirmede aşırı yüklenme ve belirsizlikler daha fazladır. Tabloya dayalı yönlendirmede hedefe gidilen yol, daha önce belirlenmiş olan uç noktalardaki tablolar ile bulunur dolayısıyla işlem yoğunluğu ve gecikme olmaz. İstekle tetiklenen yönlendirmede hedefe giden yollardan herhangi biri seçilir bu yol genellikle bulunan ilk yoldur. Bu yüzden en iyi yol çözümünü gerçekleştirmez. Tabloya dayalı yönlendirmede ise yol için en iyi çözüm bulunur. Hangi yöntemin daha iyi olduğu ağdaki düğüm sayısına, uç noktaların hareket kabiliyetlerine, trafik yoğunluğuna, yol bulma işleminin maliyetine, topolojideki değişimlerde yolların güncellenme hızına, uzak noktalar arası iletişim gibi sorunlarına bakılarak karar verilebilir [13].

### **2.2.1. OLSR (Optimized Link State Routing) protokolü**

Kablosuz ad-hoc ağlardaki popüler proaktif yönlendirme protokolüdür ve IETF RFC 3626 ile standartlaştırılmıştır. OLSR protokolü, bağ durum (Link state) algoritmasının optimize edilmiş halidir. Bağ durum algoritmasının güvenilirliğini kullanarak, kontrol mesajlarının tüm ağa gönderilmesiyle komşu uç noktalar bulunur. Ağdaki yolların hesaplanmasında hop sayısı metriğine (hop count metric) dayalı klasik en kısa yol algoritmasını (Shortest path algorithm) kullanır. Ağın bağ durum bilgisi elde edilmesi için optimize edilmiş yayın mekanizması OLSR'de ön plana çıkan özelliktir. Şekil 2.4'de görüldüğü gibi her bir uç nokta, komşuları arasından MPR (multipoint relay) seçer, 2-Hop komşular, MPR'ler ile rebroadcast mesajlarını alırlar [16].



Şekil 2.4. OLSR yönlendirme protokolü MPR seçimi

Bu teknik, diğer taşıma tekniklerine göre mesajların getirdiği yüklenmeyi azaltır, her uç nokta aldığı her mesajın kopyasını tekrar iletir. MPR uç noktalarının seçilmesiyle bağlılık durumu oluşturulur. İkinci bir optimizasyon, ağdaki kontrol mesajlarının sayısı azaltılarak yapılır. Üçüncü optimizasyon, bir MPR uç noktası rapor mesajlarını kendine ait MPR uç noktalarına gönderebilmesiyle ilgilidir. Klasik link state algoritmalarından farklı olarak ağda kısmi bağ durum bilgisi yayılır. Bu bilgi yönlendirmenin bulunmasında kullanılır ve geçilen uç nokta sayısına (hop count) göre en iyi yolun bulunmasını sağlar. OLSR geniş ve yoğun ağlarda MPR'lerin iyi çalışmasıyla uygundur ve gezgin ad-hoc ağlar için tasarlanmıştır [17].

OLSR, üç kontrol mesajı kullanır [18]; Komşu bulunmasını ve MPR seçimini sağlayan HELLO mesajı, bağ durum sinyalleşmesini yapan TC (Topology Control Message), birden fazla ağ arayüzünde OLSR çalışan uç noktalar tarafından gönderilen MID (Multiple Interface Declaration). Bu mesaj uç noktanın sahip olduğu IP adreslerini içerir.

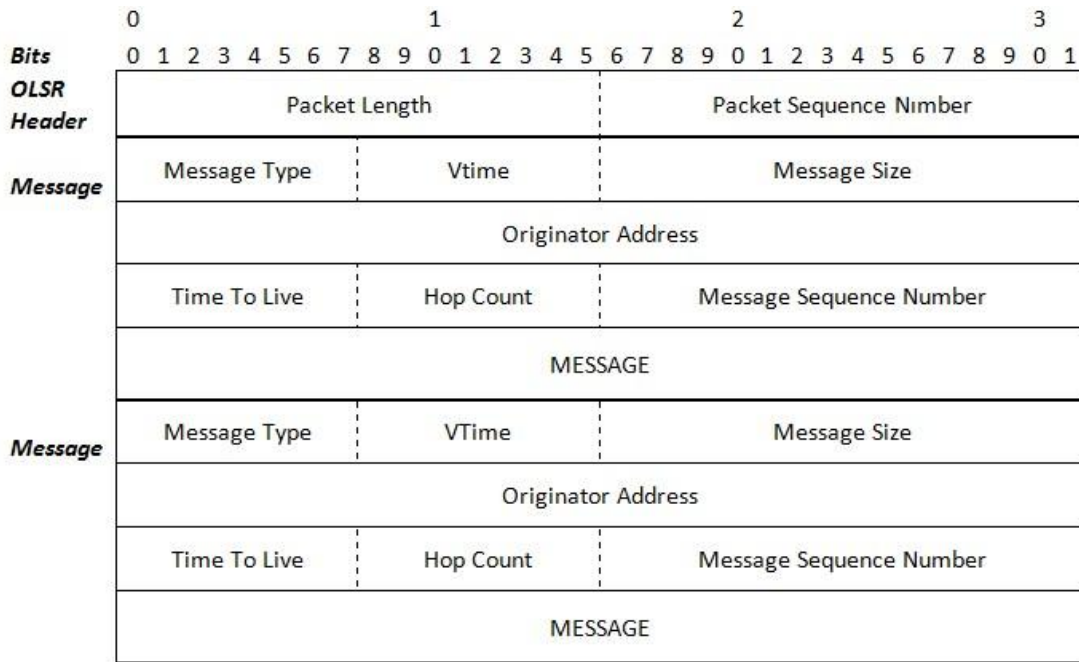
Bu kontrol mesajları ile uç noktalar üzerinde veri tabloları oluşturulur, aynı zamanda oluşturulan bu tablolar uç noktanın göndereceği kontrol mesajlarının oluşturulmasında kullanılır. OLSR bu şekilde trafiği kontrol altında tutar yani trafiği yönlendirmez, sürekli olarak güncel tuttuğu veri tablolarıyla yönlendirme tablosunu değiştirir. Bir uç noktanın OLSR protokolü çalışmasıyla beraber oluşturacağı veri tabloları şunlardır [18];



1. Çoklu Arayüz Tablosu (Multiple Interface Association Information Base), Birden fazla iletişim arayüzüne sahip uç noktaların bilgileri depolanır.
2. Bağlılık Kümesi (Link set), Komşu bağ durumu bilgisini tutar. Adres yerine Arayüz-Arayüz bağlantı bilgisi vardır.
3. Komşuluk Kümesi (Neighbor Set), Tüm direkt bağlı komşular kayıtlıdır. Bağlılık kümesine göre veriler güncellenir. Hem simetrik hem asimetrik komşular kayıtlıdır.
4. 2-Hop Komşuluk Kümesi (2-hop Neighbor Set), direkt bağlı olan uç noktalar ile ulaşılabilen tüm noktaları içerir.
5. MPR Kümesi (MPR Set), Bir uç nokta tarafından seçilen MPR'ler saklanır.
6. Topoloji Bilgi Tablosu (Topology Information Base), Uç noktalardan gelen tüm bağ-durum bilgisini depolar.
7. Mesaj Çakışma Tablosu (Duplicate Set), Son işlenen ve iletilen mesajları içerir.

OLSR yönlendirme protokolünün çalışması, lokal topolojinin bulunması için her uç noktanın periyodik olarak HELLO mesajı yayını yapmasıyla gerçekleşir. HELLO mesajları (TTL = 1) ise uç nokta komşularına iletilmez. Hello mekanizmasıyla 2-Hop komşular bilinir. HELLO mesajında bağlılık kümesi, komşu kümesi ve 2-hop komşu kümesi bilgileri bulunur. Bu bilgilerle her uç nokta kendi MPR'sini hesaplar. Ayrıca, uç noktalar, durum bilgisini içeren TC mesajlarını MPR'ler ile yayılımını optimize ederek ağa gönderir. TC mesajı kaynak noktanın komşu bilgilerini barındırır. Bu mesajların yardımıyla herbir uç nokta için, komşu uç noktaları, 2-hop komşuları, ağ içindeki tüm uç noktaların bağ durum bilgileri elde edilir. Bu bilgiler kullanılarak en kısa yol algoritması olan Dijkstra algoritması ile OLSR yönlendirme tablosu hesaplanır [16].

Tüm OLSR trafiği mesajları OLSR paketleri içerisinde gönderilir. Şekil 2.5'de OLSR mesaj paketi yapısı görülmektedir, buna göre OLSR mesajında birden fazla kontrol mesajı bulunabilir [18]. Paket yapısı detayları şu şekildedir;



Şekil 2.5. OLSR mesaj paketi yapısı [18]

Packet Length, Tüm paketin uzunluğunu belirtir, paket başlığını da içerir.

Packet Sequence Number, Göndericinin her OLSR paketi göndermesinde değeri bir artırılır. Her ağ arayüzü için ayrı "sequence number" kullanılır.

Message type, Mesaj tipini tanımlar. 0-127 OLSR için ayrılmıştır, 128-255 arası protokole ait yapılabilecek özel eklentiler için kullanılır.

Vtime, Alınan mesajların geçerlilik aralıklarını belirler.

Message Size, Mesajın uzunluğunu belirtir , mesaj başlık uzunluğunu içerir.

Originator Address, Mesajı gönderen kaynağın adresi

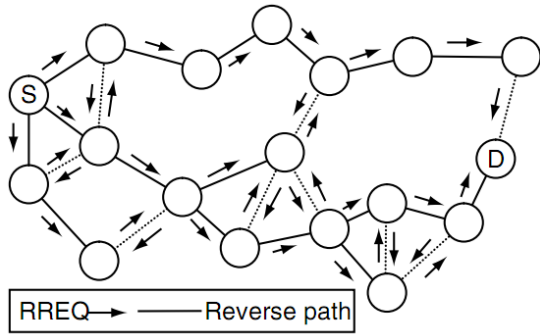
Time To Live, Mesajın geçebileceği Max. uç nokta sayısını belirler.

Hop Count, Mesajın kaç kez iletildiği bilgisidir.

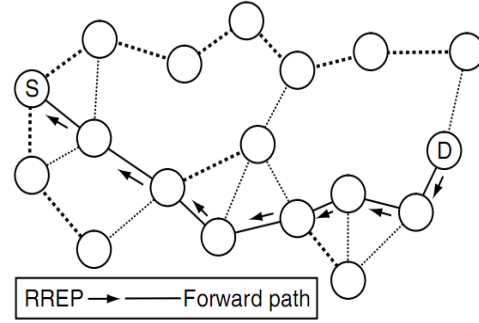
Message Sequence Number, Her yeni mesaj paketi gönderiminde bir artırılır.

### 2.2.2. AODV (Ad-Hoc On-Demand Distance vector R.) protokolü

MANET için kullanılan popüler reaktif yönlendirme (istek tetikli) protokolüdür. Yollar istenildiğinde oluşturulur. Sadece aktif yollar kullanılır. Bu özellik, ağda oluşabilecek yüklenmeyi azaltmasına karşın yolların gerektiğinde oluşması gecikmeye sebep olur. IETF RFC 3561'de standartlaştırılmıştır. Şekil 2.6 ve Şekil 2.7'de görüldüğü gibi yolların bulunması için basit istek cevap mekanizması kullanır [16].



Şekil 2.6. AODV protokolü kaynağın yaptığı yayın

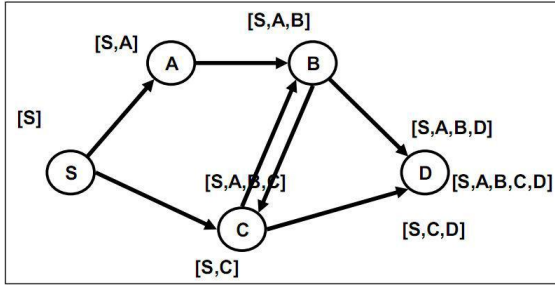


Şekil 2.7. AODV protokolü hedeften dönen cevap

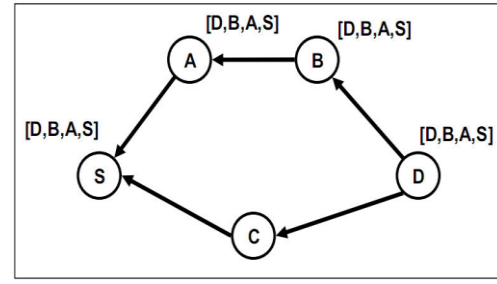
Bağlılık bilgisi için HELLO mesajları ve aktif yollardaki kopuk bağlantıların bulunması için hata mesajları kullanılır. Her yol bilgisinin ardışıl numara ile ilişkilendirilmiş zaman aşımı süresi vardır. Ardışıl numaranın kullanılması ile zaman aşımı olmuş verinin tesbit edilmesi sağlanır. Böylece en güncel yol bilgisi kullanılır. Döngüye girilmez yani klasik "distance vector" protokollerinde ortaya çıkan "counting to infinity" gibi problemlerin ortaya çıkması engellenir [16].

### 2.2.3. DSR (Dynamic Source Routing) protokolü

IETF MANET çalışma gurubunca RFC 4728'de standart hale getirilmekte olan reaktif protokoldür. hedefe gidilecek yol bir uç noktanın herhangi bir hedef noktaya ulaşmak istemesiyle hesaplanır.



Şekil 2.8. DSR protokolü kaynağın ROUTE REQUEST mesajı yayını



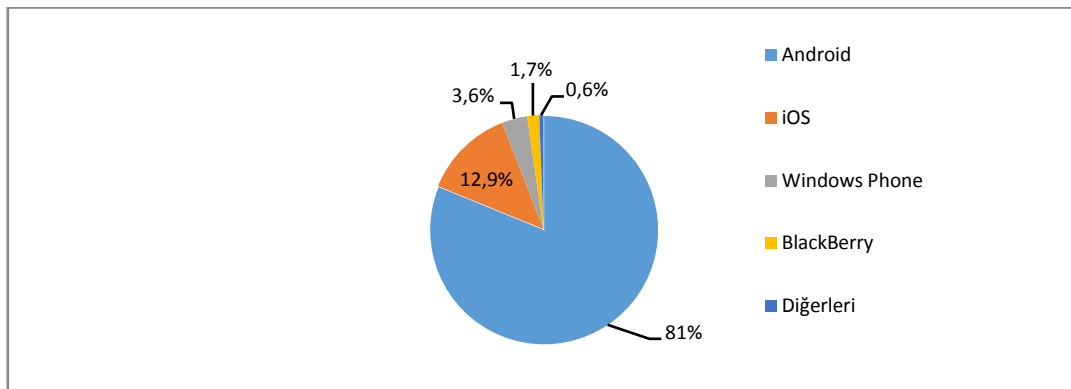
Şekil 2.9. DSR protokolü hedeften gelen ROUTE REPLY mesajı

Yolun bulunmasında "route request" ve "route reply" mesajları kullanılır (Şekil 2.8 ve Şekil 2.9). "Route request" mesajı tüm ağa gönderilir. Mesajın geçtiği uç nokta adreslerinden yol tablosu çıkarılır. "Route reply" mesajı elde edilen bu yol bilgisini kaynak noktaya göndererek hedefe ait yol üzerinden iletişim başlatır. Uç nokta kopmaları "RERR" mesajı ile ağa bildirilir [16].

## BÖLÜM 3. ANDROID İŞLETİM SİSTEMİ VE AD-HOC MODU

### 3.1. Android İşletim Sistemi

Günümüzde mobil teknolojiler (akıllı telefon, tablet) kullanımı giderek artmaktadır. Bu cihazların sahip olduğu donanımsal özelliklerin gelişmesiyle beraber mesajlaşma (SMS), konuşma gibi basit iletişim aracı olmaktan öteye geçerek medya, internet, oyun ve döküman yönetimi gibi özelliklere sahip cihazlara dönüşmüştür. Akıllı telefonların yanında tabletlerin kullanımı da artmıştır. Ticari olarak satılan mobil cihazlarda Android, IOS, Windows Phone, Blackberry OS gibi işletim sistemleri bulunmaktadır. Android, cihaz üreticileri tarafından tercih edilen yaygın işletim sistemidir, kullanım alanı akıllı telefonlar ile sınırlı kalmamış GPS navigasyon cihazları, uydu alıcılar, televizyonlar, fotoğraf makinelerinde kullanılır duruma gelmiştir. Piyasada bulunan cihazların çoğunluğunu Android ve iOS işletim sistemlerine sahip mobil cihazlar oluşturur. Şekil 3.1’de görüldüğü gibi 2013 Kasım ayı verilerine göre, Android işletim sistemine sahip cihazların kullanımı %81 oranla ilk sırada yer almakta bunu iOS işletim sistemine sahip cihazlar %12,9 oranıyla takip etmektedir [2]. Grafikte de görüldüğü gibi daha önceki senelerde yaygın olarak kullanılan BlackBerry, Windows Phone işletim sistemli cihazların kullanımında büyük bir azalma olmuştur.



Şekil 3.1. IDC verilerine göre 2013 yılı mobil cihazlarda kullanılan işletim sistemi oranları

Android işletim sistemli cihazların yaygın olması, açık kaynak kodlu olmasına, cihazın birçok kaynağını kolaylıkla kullanabilecek uygulama geliştirme alt yapısının olmasına bağlanabilir. Android, linux tabanlı, Google tarafından desteklenen açık kaynak kodlu işletim sistemidir (AOSP - Android Open Source Project). Başlangıçta Android şirketi tarafından geliştirilmeye başlanmış 2005 yılında Google bu şirketi satın almıştır. 2007 yılında HTC, Samsung, Sony gibi büyük firmalar ve donanım, telekomünikasyon şirketlerinin içinde bulunduğu OHA (Open Handset Alliance) konsorsiyumu Android işletim sisteminin geliştirilmesine destek vermeye başladı [19]. İlk Android işletim sistemli telefon HTC Dream 2008 yılında piyasaya sürüldü. Daha sonraki yıllarda diğer cihaz üreticileri de Android işletim sistemli cihazlar piyasaya sürerek kullanım yaygınlaştı. Bu sırada da işletim sistemi sürekli geliştirilerek ve yeni özellikler ile yeni versiyonlar çıkarıldı. Tablo 3.1’de 2009’dan 2013’e kadar çıkarılan Android işletim sistemi versiyonları ve isimleri görülmektedir. Versiyonların kısa aralıklar ile yayınlanması dikkat çekmektedir.

Tablo 3.1. Android işletim sistemi versiyonları

Sürüm	İsim	Yayın tarihi
1.5	Cupcake	30.04.2009
1.6	Donut	15.09.2009
2.0/2.1	Eclair	26.10.2009
2.2	Froyo	20.05.2010
2.3	Gingerbread	06.12.2010
3.0/3.1/3.2 (sadece tablet için)	Honeycomb	01.02.2011
4.0	Ice Cream Sandwich	19.10.2011
4.1	Jelly Bean	09.07.2012
4.2	Jelly Bean	29.10.2012
4.3	Jelly Bean	24.07.2013
4.4	KitKat ®	31.10.2013

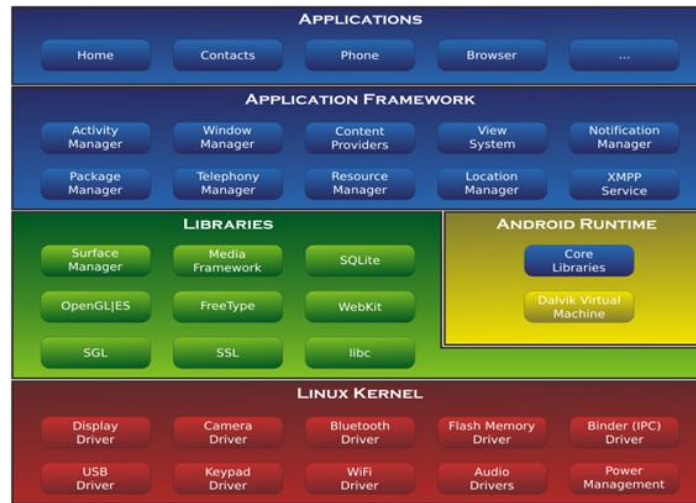
Android uygulamaları, oluşturulmuş olduğu API’nin ileri versiyonlarıyla çalışabilmektedir [20]. Yani Gingerbread versiyonu için hazırlanmış bir uygulama KitKat sürümü için de çalışabilmektedir. Bu durum geçmişte yapılmış uygulamaların çalışabildiği cihaz sayısını artırmaktadır.

Uygulama geliştiricilerin uygulamalarını yayınlayabileceği Google tarafından yönetilen uygulama deposu mevcuttur. Başlangıçta ismi “Android Market” olan depo daha sonra “Google Play” olarak değiştirilmiştir. Uygulama geliştiricilerin uygulamalarını ücretli veya ücretsiz olarak yayınlayabileceği ortam sunmaktadır.

Android cihaz kullanıcıları sahip oldukları Google hesaplarıyla çoğu ücretsiz bu uygulamaları indirebilmekte, güncelleyebilmektedirler.

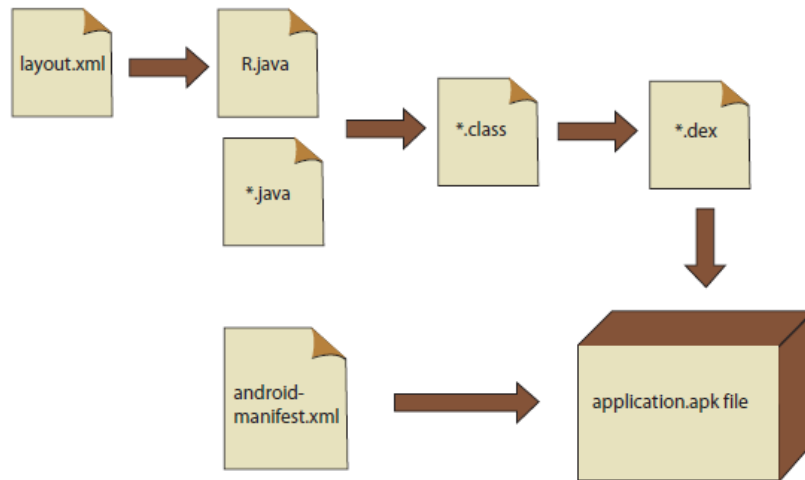
### 3.2. Android İşletim sistemi Mimarisi

Android işletim sistemi Şekil 3.2’de gösterildiği gibi beş katmandan oluşmaktadır.



Şekil 3.2. Android işletim sistemi yığın yapısı [19]

En üst katmanda uygulamaların çalıştığı uygulama katmanı vardır. Uygulamalar Java dili kullanılarak geliştirilir. Uygulamalar Java virtual machine yerine Dalvik virtual machine üzerinde "application sandbox" ile birbirinden yalıtılmış olarak çalışır [21].



Şekil 3.3. Android uygulamasının derlenme süreci [22]

Uygulamalar DVM üzerinde çalıştığından java kodunun derlenmesiyle oluşturulan ".class" dosyaları bu sanal makinenin anlayabileceği ".dex" dosya formatına çevrilmelidir. Şekil 3.3'de android ".apk" uzantılı uygulama paketinin derlenme aşamaları genel olarak gösterilmiştir. Kullanıcı arayüzü ile ilgili kullanılan ".xml" dosyaları "aapt" aracıyla ".xml dosya uzantılı" binary dosyasına çevrilerek R.java dosyası elde edilir. Java dosyaları ".class" dosyalarına çevrildikten sonra ".dex" binary formatına çevrilir. Projedeki "Drawables" ve "values" klasörlerinde bulunan dosyalar ile ".dex" dosyası ".apk" uzantılı arşiv haline getirilir. Derleme süreci ADT (Android Debug Tool) tarafından yönetilir [22].

Uygulama katmanı altında uygulama geliştiricilere hazır kütüphaneler sunan uygulama çatısı katmanı (application framework) katmanı bulunur. C/C++ kütüphaneleri üzerinde çalışarak java kodu ile erişimini sağlar.

Kütüphaneler (Libraries), çekirdeğin (Kernel) yönetimini sağlayan C/C++ ile yazılmış kütüphanelerden oluşan katmandır. Donanıma birbirinden farklı kütüphaneler olarak erişilmesini sağlar.

Android runtime, DVM ve kütüphanelerden oluşan katmandır. DVM, java sanal makinesinin Android için özelleştirilmiş halidir [23]. Bellek kullanımı optimize edilerek mobil cihazlar için verimli çalışacak duruma getirilmiştir.

Linux çekirdeği katmanı (Linux kernel layer), donanım ile işletim sistemi arasında iletişimi sağlayarak donanımı kullanılabilir duruma getiren katmandır.

Tablo 3.2. Android versiyonları ve kullandıkları Linux çekirdeği versiyonu

<b>Android Versiyon</b>	<b>Linux Çekirdek Sürümü</b>
1.6	2.6.29
2.2	2.6.32
2.3	2.6.35
3.0	2.6.36
4.0	3.0.1
4.1-4.4	3.0.31



Linux çekirdeği, Android için özelleştirilerek kullanılmıştır. Tablo 3.2'de Android versiyonlarında kullanılan Linux çekirdek versiyonları belirtilmiştir.

Linux çekirdeği GPL (General Public License) lisansına, çekirdek haricinde işletim sistemini kapsayan kısım Apache lisansına sahiptir [22]. Android versiyonlarının kaynak kodu "source.android.com" sitesinden elde edilebilir. Açık kaynak kodlu olmasının etkisiyle özelleştirilmiş ROM'lar (customized ROM) üreten bazı topluluklar oluşmuştur. Bu topluluklar Android işletim sisteminin kaynak kodunda iyileştirmeler yaparak piyasada bulunan cihazlar üzerinde çalışmasını sağlarlar. En iyi Android ROM'ları şu şekilde sıralanabilir; Paranoid Android, AOKP, CyanogenMod, Liquid Smooth, MIUI, Jelly BAM, Avatar ROM, SlimBean [24]. En yaygın olarak kullanılan ve cihaz yelpazesi geniş olanı CyanogenMod ROM'larıdır. Özelleştirilmiş ROM'ları cihazlara yüklemek için cihaz "unlock" yapılmalı ve "root" haklarına sahip kullanıcısı aktif hale getirilmeli ve bootloader değiştirilmelidir. Bundan sonra varolan ROM (stock ROM) değiştirilebilir.

### **3.3. Android Haberleşme Alt Yapısı ve Ad-hoc Modu**

Android cihazlar donanımsal olarak gelişmiş özelliklere sahiptir. Çoğu cihazda ivme ölçer, yönelim sensörü, mesafe sensörü, ışık sensörü, GPS, bluetooth, Wi-Fi, GSM modem gibi donanımlar bulunmaktadır. Bu donanımlar uygulama geliştirici tarafından esnek olarak kolaylıkla kullanılabilir. Bluetooth, GPS, Wi-Fi ve cihazın diğer özelliklerini kullanan çok sayıda uygulamanın google play'de bulunması bu durumu kanıtlayabilir.

Cihazlarda bulunan bluetooth özellikleri şu şekilde sıralanabilir; iki cihaz arasında direkt iletişimi sağlar, iletişim mesafesi kısadır ve konfigürasyonu uzun sürer.

Cihazlarda bulunan Wi-Fi arayüzü, IEEE 802.11 b/g/n alt yapılarını desteklemektedir. Wi-Fi ile modem gibi dağıtıcılara bağlanarak iletişim sağlanır. Kapalı alanda iletişim mesafesi 30 metre, açık alanda yaklaşık 300 metre olmaktadır. Android cihazlar, bir dağıtıcıya bağlanarak iletişim kurabilirken direkt birbirine bağlanarak iletişim kuramamaktadır. Google ve cihaz üreticilerine [25]'de belirtildiği

gibi ad-hoc modunun destekler hale getirilmesi için yoğun talep vardır. Ad-hoc modunun aktif hale getirilmesi bir bilgisayarla internet kullanımının paylaşılmasını veya ad-hoc ağlar kurulabilmesini sağlayacaktır. Özel ROM üreten gruplar hazırladıkları ROM'da ad-hoc desteği sağlayabilmektedir. Örneğin CyanogenMod grubu, hazırladıkları ROM'ların bazılarında ad-hoc modunu aktif tutabilmektedir. Cihazın ad-hoc modunu desteklemesi için çekirdek ya da işletim sistemi seviyesinde değişiklikler yapılması gerekir. Farklı model cihazlar aynı donanım ve çekirdeğe sahip olamayabilirler. Bu da cihazın ad-hoc modunun aktif hale getirilmesi için yapılması gereken çalışmayı cihaza özgü hale getirmektedir.

## BÖLÜM 4. AKILLI CİHAZLAR İLE ALTERNATİF HABERLEŞME

### 4.1. Android Cihazların Ad-hoc Modu Aktifleştirilmesi

Yapılan çalışma kapsamında HTC Desire S (Şekil 4.1) ve ASUS NEXUS 7 (Flo) (Şekil 4.2) cihazları kullanılmıştır. Cihazlarda ad-hoc modu aktif değildir ve haberleşme ağının kurulabilmesi için aktif hale getirilmelidir. Aktifleştirilmesi için yapılması gerekenler iki cihazda farklı olmak ile beraber yapılan işlemler genel olarak şu şekildedir; cihazlar "unlock" yapılarak önyükleyici (bootloader) değiştirilebilir duruma getirilmiştir, önyükleyici (bootloader) değiştirilerek özel ROM ve kernel yüklenmiştir.



Şekil 4.1. HTC Desire S



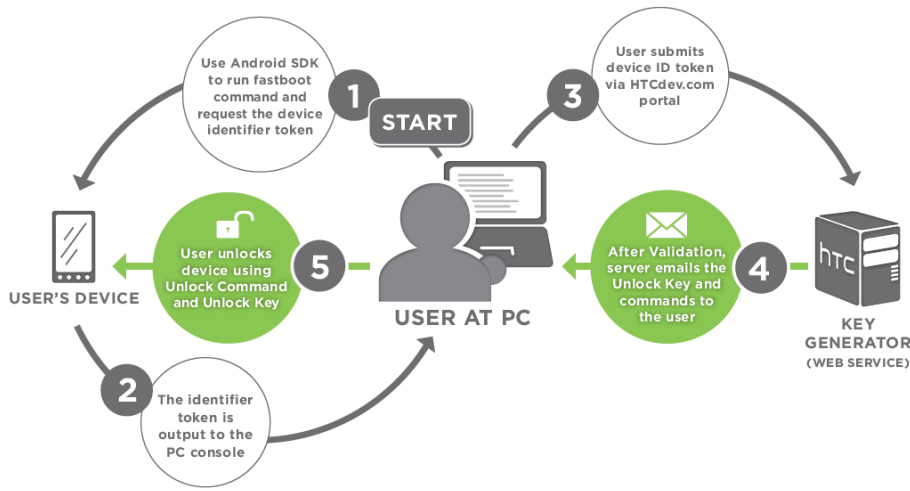
Şekil 4.2. ASUS Nexus 7 (flo)

#### 4.1.1. HTC Desire S ile yapılan çalışma

HTC Desire S cihazının işletim sistemi dosyaları üzerinde değişiklik yapılabilmesi için "unlock" yapılması gerekir. Telefon başlangıçta "lock" durumundadır ve bootloader yazılımı değiştirilemez. "unlock" olması "boot", "system", "recovery" gibi alanlara yazma erişiminin olması ile ilgilidir. HTC telefonlarında önyükleyici yazılımının "lock" ve "unlock" olması S-ON/S-OFF (Security-ON/Security-OFF) ile

ifade edilir. Cihaz üretildiğinde S-ON olmasının sebebi, hboot gibi tüm diğer alanlara erişen bir alanın yanlışlıkla değiştirilmesi cihazı onarılamaz hale getirmesi ile ilgilidir [26].

HTC firması, resmi olarak cihazları S-OFF olması için desteklemektedir. Yapılması gerekenler HTCDEV'de [27] detaylı bir şekilde anlatılmıştır. Şekil 4.3'de bu adımlar görülmektedir. Bu adımların yerine getirilmesi sırasında cihaz kullanılamaz duruma gelebilir, olumsuz duruma karşı ilgili kaynakta gerekli uyarı yapılmıştır.



Şekil 4.3. HTC Desire S S-OFF yapılması aşamaları [28]

Cihazın S-OFF yapılma işlemi, Windows 7 işletim sistemli bilgisayar kullanılarak HTC'nin sunmuş olduğu yöntemlerle gerçekleştirilmiştir. Bilgisayar üzerinde Java Runtime Environment (JRE), HTC Sync, Android SDK kurulu olmalıdır. HTC Sync programı, cihaz sürücülerine sahip olduğundan önemlidir. Android SDK kurulu olduğu dizin altındaki "tools" klasörü Windows işletim sistemi "path" değişkeninde tanımlanması gerekir. HTCDEV sitesinde kullanıcı hesabı açılmalıdır. Siteye kullanıcı hesabıyla giriş yaptıktan sonra ilgili cihaz seçilmelidir [28]. Bundan sonra izlenmesi gereken adımlar şu şekilde açıklanmıştır [27];

Telefon ön yükleyici ekranının açılması için Şekil 4.5'de görüldüğü gibi *Ayarlar > Güç* altında *Hızlı Açılış* özelliğinin kaldırılması gerekir.

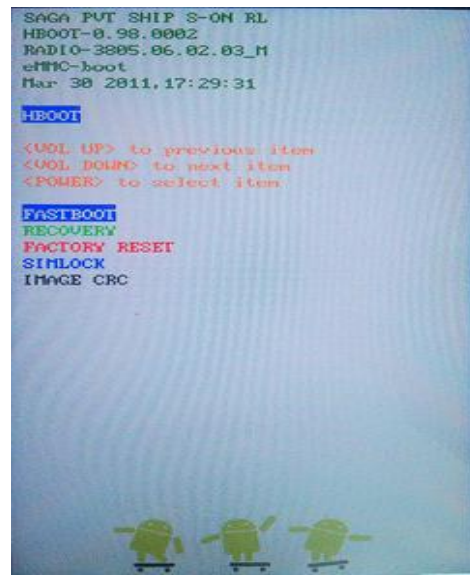


Şekil 4.4. HTC Desire S Önyükleyici ekranının başlatılması [26]

Cihaz, Şekil 4.4'deki gibi *Power+Volume Down* tuşları kombinasyonu ile açıldığında Şekil 4.6'da görüldüğü gibi boot loader açılış ekranı gelecektir. Gelen ekranın üst tarafında "S-ON" yazılı olması boot loader "locked" durumunda olduğunu gösterir.



Şekil 4.5. HTC Desire S hızlı açılış ayarı



Şekil 4.6. HTC Desire S boot loader lock ekranı

Telefonun ses azaltma ve artırma tuşları kullanılarak *FASTBOOT* seçilir (Şekil 4.7).



Şekil 4.7. Fastboot seçilmesi

Telefon USB kablosu ile bilgisayara bağlanır (Şekil 4.8)



Şekil 4.8. Telefonun bilgisayara bağlanması

Bilgisayarda komut satırı açılır. “fastboot oem get\_identifier\_token” komutu çalıştırılır (Şekil 4.9).

```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.0.6002]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.

C:\Users\Admin>cd \Android
C:\Android>fastboot oem get_identifier_token
  
```

Şekil 4.9. Fastboot komutunun çalıştırılması

Ekranda Şekil 4.10’daki gibi kodlar çıkacaktır. “<<<< Identifier Token Start >>>>” kısmı dahil <<<<Identifier Token End >>>> arası seçilerek kopyalanır.

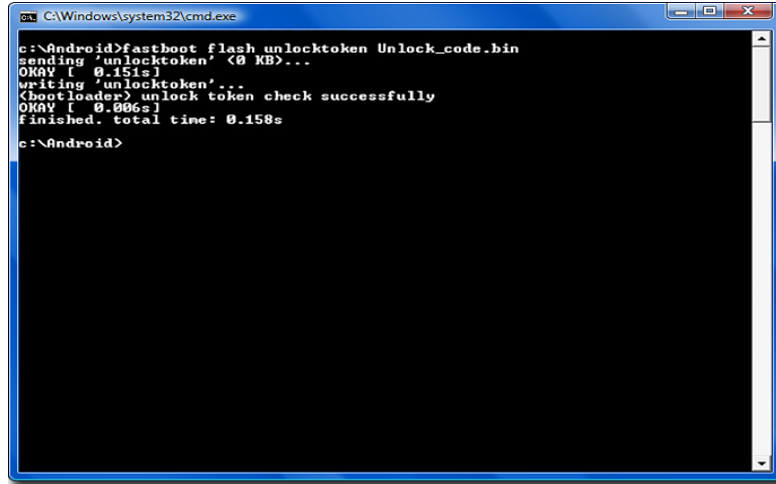
```

C:\Android>fastboot oem get_identifier_token
-- INFO
INFO<<<< Identifier Token Start >>>>
INFO439CCCF6C76A1F517CD550B29518A44F
INFOEC16ED2072C92B296FBC0932D9470B6E
INFOC8B7C1790A7E2487513F4922B8E573F
INFO4F8A1C604CA40B96B3A07791767613D
INFOD3799A9D207FAA7E36DDEA819E6785
INFODEA4084505F18DB293BF15129DA0328
INFO359538ACFA5B0CEB326C0B654DD1B238
INFOB1350E7E7F8A392A1C8267AA276F29FF
INFO43F5AC77C06AAEC9FEC22906DE942AA5
INFO80D7101BE5001FF00A3F57B17A8A09
INFOB84DF803E0C996CD2FCE327F6670890
INFOD7FA30A479850582CCE6BC0A47A28943
INFO43E225525310A0E87E5C48B496B346041
INFOF96E811868740D6D2C502115987DCD72
INFO6C46C80C4BB31109E657FCC9ABD6445F
INFO50D8B8B99E310A96052DDFA4DB80D21
INFO<<<< Identifier Token End >>>>
OKAY [ 0.063s]
finished. total time: 0.063s
C:\Android>_
  
```

Şekil 4.10. Cihaz tanımlayıcı kodu

HTCDEV sitesinde daha önce girilmiş olan hesapta “My Device Token” alanına yapıştırılır ve gönderilir. HTC tarafından telefonu “unlock” yapan telefona özel, ikili

kodlu (binary) “Unlock\_code.bin” dosyası kullanıcı mail hesabına gönderilir. Elde edilen dosya “fastboot.exe” ile aynı dizinde olacak şekilde yerleştirilir. Komut satırında “**fastboot flash unlocktoken Unlock\_code.bin**” komutu çalıştırılır (Şekil 4.11).



```

C:\Windows\system32\cmd.exe
c:\Android>fastboot flash unlocktoken Unlock_code.bin
sending 'unlocktoken' <0 KB>...
OKAY [ 0.151s]
writing 'unlocktoken'...
[bootloader] unlock token check successfully
OKAY [ 0.006s]
Finished. total time: 0.158s
c:\Android>

```

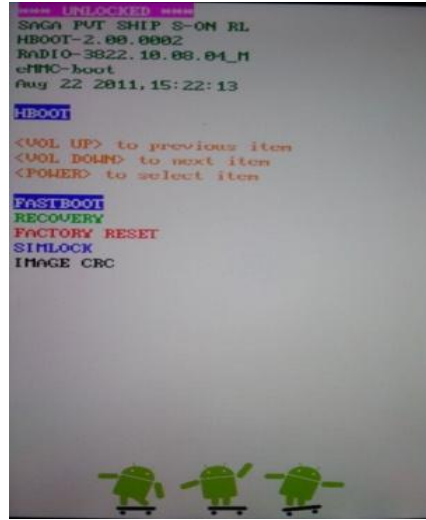
Şekil 4.11. Fasboot komutu ile unlock yapılması

Telefonun ekranında Şekil 4.12’de görüldüğü gibi “unlock” yapılıp yapılmak istenmediğini soran onay ekranı çıkar. Onaylanırsa telefon fabrika ayarlarına döndürülerek “unlock” yapılır. Onaylanmazsa, telefon herhangi bir değişiklik olmadan tekrar başlar.



Şekil 4.12. Unlock onay ekranı

Telefon "unlock" yapıldığında önyükleyici ekranı Şekil 4.13'deki gibi olmalı ve ekranın üst tarafında "UNLOCKED" yazısı görülmelidir. Buraya kadar olan kısımda telefon önyükleyici kilidi açılarak telefonun özel alanlarının değiştirilme hakkı elde edilmiş oldu. Telefonu süper kullanıcı haklarına yükseltme ya da özel ROM yüklemek için telefonda varolan kurtarma (Recovery) sisteminin değiştirilmesi gerekir. Android cihazlar için geliştirilmiş olan varolan sistemden çok daha gelişmiş özelliklere sahip "ClockworkMod Recovery" (CWM) kurulmalıdır.

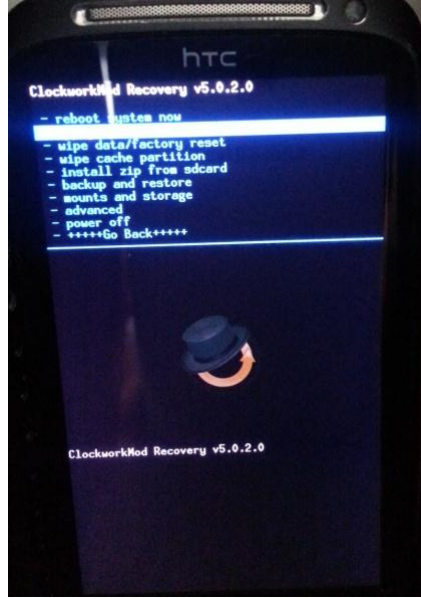


Şekil 4.13. HTC Desire S önyükleyici ekranı

CWM kurulumunda aşağıdaki adımlar izlenir;

1. "ROM Manager" web sitesinde [29] HTC Desire S için olan "recovery-clockwork-5.0.2.0-saga.bin" dosyası seçilerek CWM elde edilir. Bilgisayarda "fastboot.exe" ile aynı dizine konur.
2. Telefon önyükleyici ekranı Power+Volume Down tuşları kombinasyonu ile açılır. Fastboot menüsüne girilir.
3. Telefon bilgisayara USB kablosu ile bağlanır. Komut satırında "fastboot flash recovery recovery-clockwork-5.0.2.0-saga.bin" çalıştırılır. Yükleme tamamlanmıştır. Telefon kapatılıp açılır ve önyükleyici ekranında "recovery" seçilir, yüklenen CWM menüsü Şekil 4.14'de görülmektedir.





Şekil 4.14. HTC Desire S yüklenen CWM

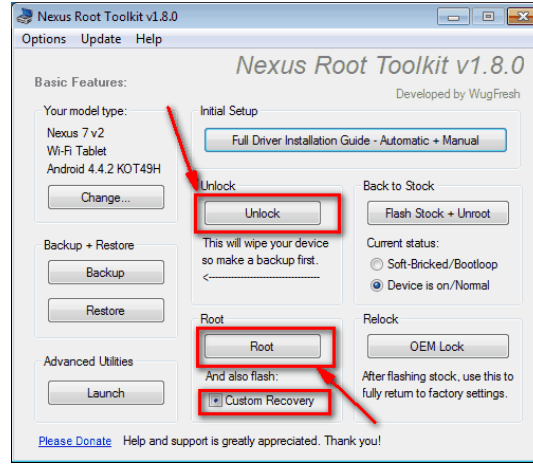
Cihazın süper kullanıcı haklarına yükseltilmesi için CWM ile "UPDATE-SuperSU-v1.34.zip" süper kullanıcı dosyasının cihaza kurulması gerekir. Dosya SD karta konmalıdır. CWM menüsünde, "install zip from sdcard" sekmesi altında "choose zip from sdcard" seçilir. SD kart dizini içinde daha önce yerleştirilen süper kullanıcı dosyası bulunarak kurulum gerçekleştirilir. Telefon yeniden başlatıldığında uygulamalar arasında superuser uygulaması görünecektir, telefon süper kullanıcı haklarına sahip olmuştur.

Telefonda ad-hoc modunu aktif hale getirmek için "wpa\_supplicant" dosyasının değiştirilmesi gerekir. [30]'da HTC Desire S telefonu için uygun dosya bulunmaktadır. CWM kullanılarak varolan "wpa\_supplicant" değiştirilir. Ad-hoc modu aktif hale getirilmiştir. Cihaz, artık ortamda varolan ad-hoc ağlarına bağlanabilecektir.

#### 4.1.2. ASUS Nexus 7 (flo) ile yapılan çalışma

Cihazın ad-hoc modunu aktif hale getirmek için Cyanogenmod 10.2.0 ROM versiyonu kurulmuştur. yüklemenin gerçekleşebilmesi için bootloader "unlock" yapılmalı, varolan kurtarma sistemi yerine CWM (ClockWorkmod Recovery)

yüklenmelidir. Bu adımları ayrı ayrı yapmak yerine WugFresh [31] tarafından, hepsini bir arada kolaylıkla gerçekleştiren yazılım geliştirilmiştir.



Şekil 4.15. Nexus Tool Kit ekran görünümü

Nexus 7 cihazının "unlock" yapıp CWM kurulması için Nexus Root Toolkit programındaki Şekil 4.15'de belirtilen "Unlock" ve Root tuşları kullanılması yeterli olacaktır [32]. Başlangıçta bilgisayara "Full Driver Installation Guide" ile gerekli sürücüler kurulur. Nexus 7 için "USB Debugging" aktif hale getirilir. Tablet, bilgisayara USB ile bağlanır. Sürücülerin testi başarıyla yapıldıktan sonra önyükleyiciyi "unlock" yapmak için programdaki "Unlock" butonuna basılır ve bu işlemi program otomatik olarak gerçekleştirir. Tablet ekranında onay ekranı çıkar, onaylandıktan sonra "unlock" işlemi tamamlanır. CWM kurulup süper kullanıcı haklarına yükseltmek için, programdaki "Custom Recovery" seçilip "Root" tuşuna basılır. İşlem bittiğinde tablet, özel ROM yüklemeye hazırdır. Cyanogenmod grubunun Nexus 7 için hazırlamış olduğu 10.2.0 versiyonu kullanılır [33]. Tablet önyükleyicisi açılır, CWM menüsünde "install zip from sdcard" sekmesi altında "choose zip from sdcard" seçilerek özel ROM yüklenir. Tablet artık ad-hoc ağlara bağlanabilecek duruma getirilmiştir.

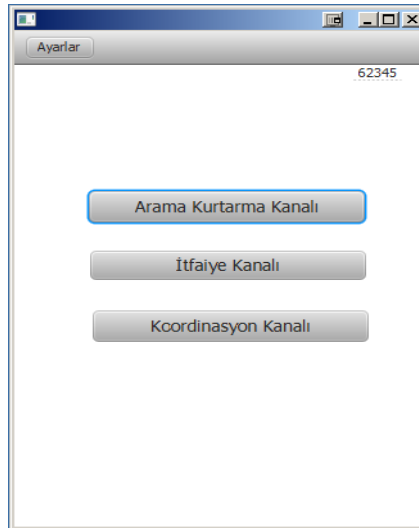
#### 4.2. Windows/Linux Bas Konuş Yazılımı

Mobil cihazlar için geliştirilen bas konuş uygulamasının kullanım alanını genişletmek için x86-x64 işlemcili dizüstü ve masaüstü bilgisayarlarda çalışan

versiyonu geliştirilmiştir. Farklı çeşit cihazların desteklenmesi, kurtarma birimlerine android cihaza sahip olmadan da dahil olmayı sağlayacaktır. Geliştirilen yazılım Windows ve Linux işletim sistemleri ile uyumludur ve gerçekleşmesinde Java dili kullanılmıştır.

Yazılım geliştirme ortamı olarak Netbeans 7.4 kullanılmıştır. Yazılım kullanıcı arayüzü tasarımları JavaFX Scene Builder 2.0 ile gerçekleştirilmiştir. Geliştirme ortamlarının sorunsuz çalışması için JDK 7 update 45 üstü versiyonunun kurulu olması gerekir. Netbeans yazılım geliştirme platformunda "JavaFX FXML Application" projesi oluşturularak yazılım geliştirilmiştir. Çalışması için herhangi bir kurulum gerekmez.

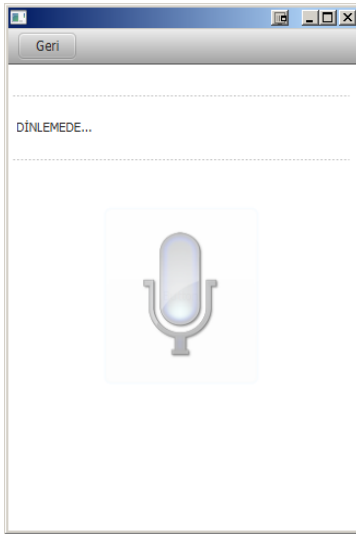
Yazılım ana ekranı Şekil 4.16'da görüldüğü gibidir. Üç farklı birim olduğu düşünülerek arama kurtarma kanalı, itfaiye kanalı, koordinasyon kanalları tanımlanmıştır. Herhangi bir kanala girildiğinde, ağ içinde sadece o kanalda olanlar ile iletişim kurulabilir.



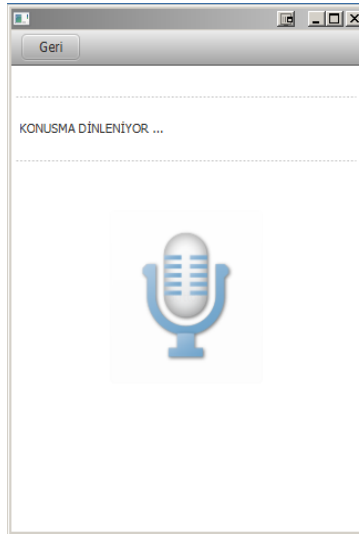
Şekil 4.16. Win/Linux bas konuş yazılımı ana ekranı

Yazılım ana ekranında herhangi bir kanal seçildiğinde konuşma durumlarını yöneten ve gösteren ekran gelecektir. Şekil 4.17 **normal durum** ekranını göstermektedir. Bu durum, kanalda konuşanın olmadığı ve kullanıcının konuşmadığı durumdur. Şekil 4.18 **dinleme durumu** ekran görüntüsüdür. Ağdan gelen komutla **dinleme durumuna** geçilmiştir ve hoparlörden kanalda konuşan kullanıcının sesi

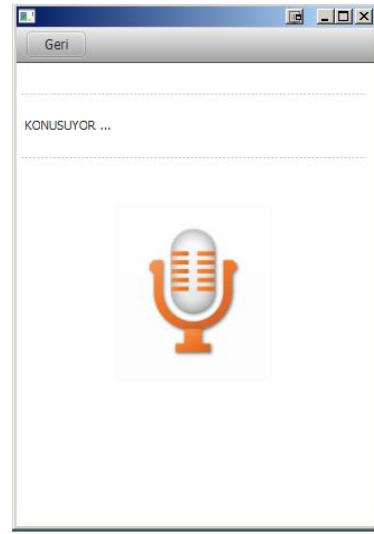
duyulmaktadır. Şekil 4.19 **konuşma durumu** ekranını göstermektedir. Bu durumda, mikrofondan alınan kullanıcının sesi kanaldaki diğer kullanıcılara ağ üzerinden iletilmektedir. Konuşma ve dinleme durum değişimleri "mouse click" ile yapılmaktadır. Fare ile mikrofon resmi üzerinde "click down" yapıldığında **konuşma durumuna**, bırakıldığında ise **normal** duruma geçilir. Ekranın sol köşesinde bulunan Geri tuşuyla kanaldan çıkılarak ana ekrana dönlür.



Şekil 4.17. Normal durum ekranı



Şekil 4.18. Dinleme ekranı



Şekil 4.19. Konuşma ekranı

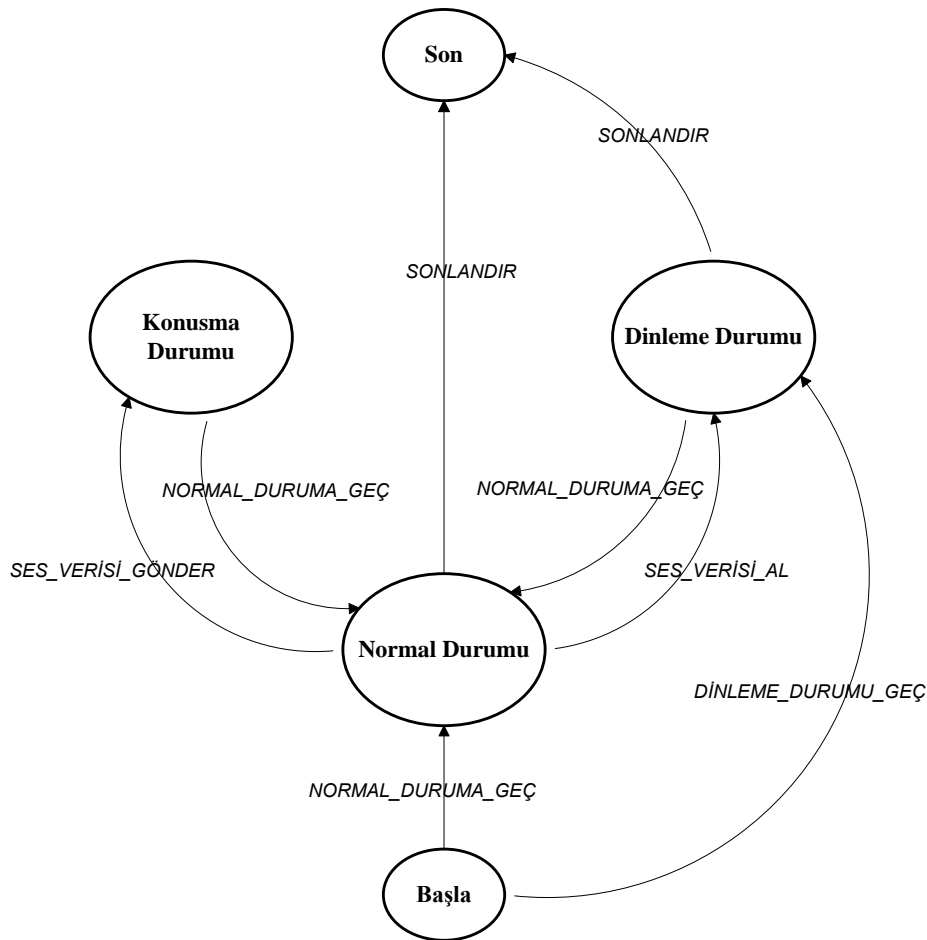
#### 4.2.1. Yazılım tasarımı

Yazılımda üç farklı kanal, yazılım kodunda önceden tanımlanmış portlar ile ayrılmıştır. Kanala girildiğinde üç durumdan birinde olabilir. Durum geçişleri, ağdan gelecek komutlar ve kullanıcı grafik arayüzünden kullanıcının girdileri ile olur. Yazılım durum makinesi Şekil 4.20'de görülmektedir. Kanala girilmesiyle başlangıçta kanalda konuşan olup olmadığı tespiti yapılır. Buna göre **normal duruma** veya **dinleme durumuna** geçilir.

Durumlar ve geçişleri şu şekildedir;

1. Normal Durum - Konuşma Durumu geçişi, kullanıcının ekrandaki mikrofon tuşuna basılı tutmasıyla gerçekleşir.

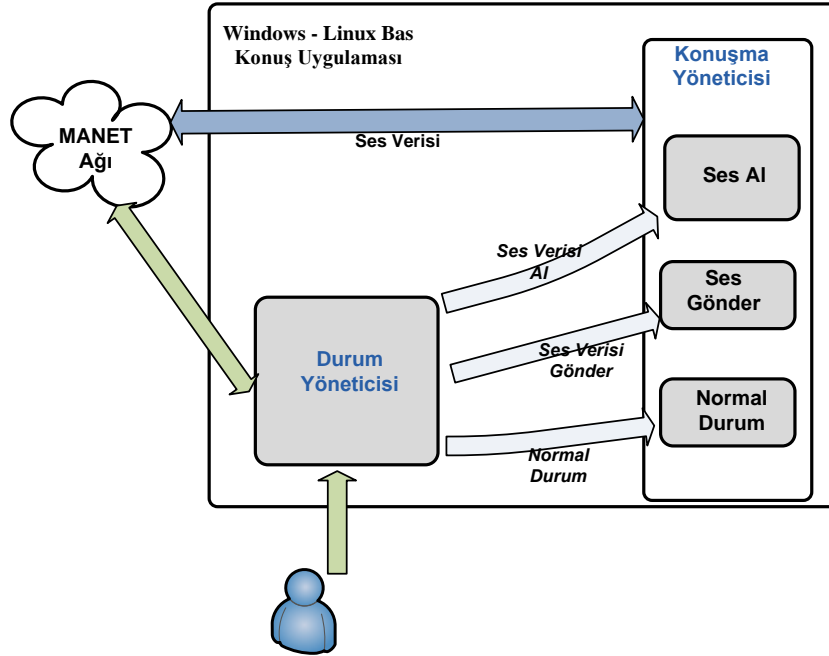
2. Konusma Durumu - Normal Durum geçişi, kullanıcının ekrandaki mikrofon tuşunu bırakmasıyla gerçekleşir.
3. Normal Durum - Dinleme Durumu geçişi, ağdan gelen dinlemeye geç komutuyla olur. Ağ içinde kanalda bulunan bir kullanıcı, kanaldaki diğer kullanıcılara dinlemeye geç komutu göndermiştir.
4. Dinleme Durumu - Normal Durum geçişi, ağdan gelen normal duruma geç komutuyla olur. Kanalda konuşmaya devam eden kullanıcı, kanaldaki diğer kullanıcılara normal duruma geç komutunu göndermiştir.



Şekil 4.20. Win/Linux bas konuş yazılımı kanal durumu makinesi

Ekrandaki "Geri" tuşunun kullanılmasıyla kanal seçme ekranına dönülür ve ses dinleme ve komut dinleme aktiviteleri kapatılarak kanal sonlanır. **Dinleme durumunda** ve **normal durumda** kanal sonlanabilir.

Yazılım tasarım blok diagramı Şekil 4.21'de gösterilmiştir. Kanala girildikten sonra konuşma ve dinleme durumunu yöneten, ses iletişimini sağlayan yöneticiler tanımlanmıştır.



Şekil 4.21. Windows-Linux bas konuş yazılım tasarımı blok diagramı

Durum yöneticisi, durum değişimlerini kontrol etmesinin yanı sıra konuşma yöneticisi durumlarını da değiştirir. Kanala girildiğinde ilk olarak, ilgili kanalın ses alma portunda UDP socket açılarak gelen ses verisinin olup olmadığı kontrol edilir. Gelen ses paketleri var ise kanalda konuşan vardır ve **dinleme durumuna** geçilir. Gelen herhangi bir ses paketi yok ise **normal durumuna** geçilir. Yapılan bu test sırasında ağdan gelen komutları almak için komut dinleme portunda UDP socket açılarak komut beklenir. Ağdan gelecek ses verisi al ve normal duruma geç komutlarının alınması ve değerlendirilmesi durum yöneticisinde olur. Kullanıcının grafik arayüzünden girdileri konuşma durumuna ve normal duruma geçişleri sağlar. Kullanıcı konuşma tuşuna bastığında ilk olarak UDP socket açılarak ağa ses verisi al komutunu gönderir ve konuşma yöneticisinin durumunu da ses verisi göndermesi için değiştirir. Kullanıcı konuşma tuşunu bıraktığında ağa normal duruma geç komutunu gönderir ve konuşma yöneticisinin normal duruma geçmesini sağlar. Yazılımın durum geçişlerini kontrol etmesinde önemli bir nokta, ağdan gelen

komutla durum deęiřimi olmuřsa yine aędan gelen komutla dięer duruma geilir. Yani kullanıcının girdileri etkisizdir. Aynı řekilde kullanıcının girdileri ile durum deęiřimi olmuřsa aędan gelen komutlar etkisizdir ve kullanıcının girdileri ile durum deęiřtirilebilir.

Konuřma yneticisi, durum geiřlerine gre javanın ses kartına eriřim alt yapısını kullanarak mikrofondan ses alıp aęa gndermeyi ve aędan alınan sesin hoparlrden verilmesini saęlar. **Dinleme durumunda** ilgili kanala ait ses dinleme portunda ses paketlerini alan UDP socket aılır ve alınan ses paketleri hoparlre verilir. Aędan gelecek *normal duruma ge* komutuyla **normal duruma** geilir ve ses verisinin alındıęı socket kapatılır, ses alma durdurulur. Hoparlre ses verisi verilmez. **Normal durumda** iken kullanıcının konuřma isteęiyle, ilgili kanala ait ses verisinin gnderileceęi portta UDP socket aılır. Ortamdaki ses mikrofondan alınarak aęa gnderilir.

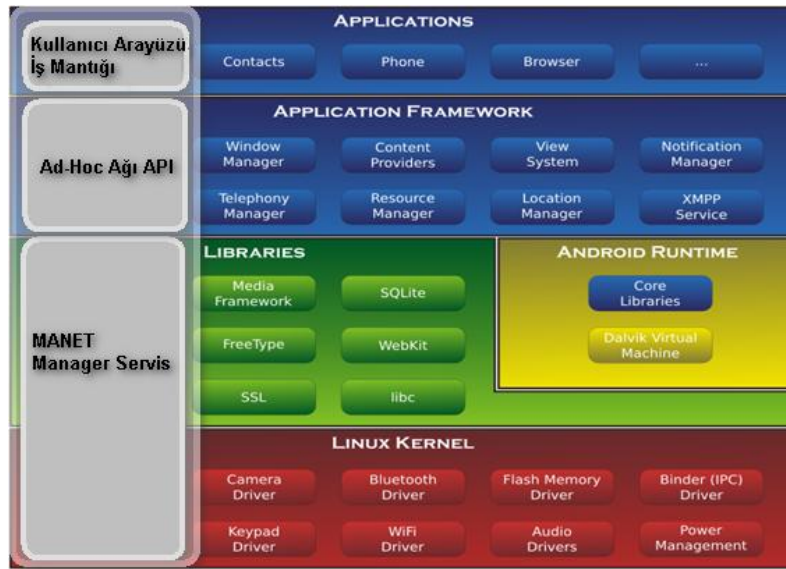
### 4.3. Android Bas Konuř Uygulaması

Afet durumlarında, MANET aęında kurtarma birimleri arasında sesli haberleřmeyi saęlayacak Android uygulaması geliřtirilmiřtir. Geliřtime ortamı olarak Windows 7 ve Ubuntu Linux iřletim sistemleri kullanılmıřtır. Yazılım projesi, ADT (Android Debug Tool) eklentili Eclipse geliřtirme ortamında oluřturularak geliřtirilmiřtir. Projenin derlenip alıřtırılabilmesi iin Android SDK, geliřtirmede kullanılan bilgisayardaki iřletim sisteminde kurulu olmalıdır. Uygulama, Android 2.3.3 ve sonrası versiyonlu tablet ve telefonlarda alıřmaktadır. Uygulamanın kurulumu, cihazlar bilgisayara USB ile baęlandıktan sonra Eclipse veya adb (Android debug bridge) ile yapılabilir.

Uygulamanın sesli haberleřme dıřında ad-hoc aęına baęlanma, IP ayarlarını deęiřtirme, ynlendirme protokolnn alıřtırılması gibi grevleri vardır. SPAN projesinin alt projesi olan MANET Manager aık kaynak kodlu atısı kullanılarak bu zellikler kullanılabilir. MANET Manager atısının kullanılabilmesi iin C/C++ kodlarının Android uygulamalarında kullanılmasını saęlayan Android NDK geliřtirme bilgisayarında kurulu olmalıdır [34]. Android NDK, C/C++ kodlarını

"cross compile" yaparak Android işletim sistemli cihazlarda çalışacak dinamik veya statik kütüphaneler hazırlar. Java, JNI (Java Native Interface) arayüzü ile bu kütüphaneleri kullanır.

Şekil 4.22'de uygulamanın Android işletim sistemi çatısı üzerindeki yeri gösterilmiştir. Uygulama katmanında kullanıcı ile etkileşimi ve konuşma yönetimini sağlayan kısım bulunur. "Application Framework" katmanında ad-hoc ağının kurulmasını ve yönlendirme protokolünün çalıştırılmasını sağlayan kısım bulunur. Çekirdek ve kütüphaneler katmanında MANET Manager Servisi çalışır



Şekil 4.22. Android bas konuş uygulamasının işletim sistemindeki görünümü

Uygulama ile ad-hoc ağına bağlanması ve OLSR protokolünün çalıştırılması için ana ekranda sağ üstte bulunan tuş kullanılmalıdır. Ad-hoc ağına bağlandığında ağdaki diğer cihazların IP'leri Şekil 4.23'de olduğu gibi gözükecektir. Şekil 4.24'de uygulama menüsü gösterilmiştir. Uygulama menüsü altındaki "Topoloji" sekmesinde yönlendirme protokolünün kullandığı tablolar Şekil 4.25'deki gibi gösterilecektir.

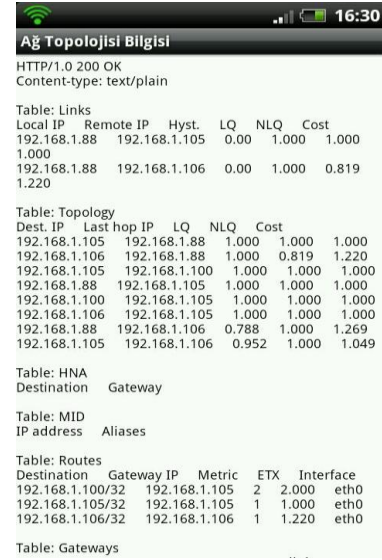




Şekil 4.23. Android bas konuş uyg ana ekranı



Şekil 4.24. Android bas konuş uyg menüsü



Şekil 4.25 Android bas konuş uyg ağ topoloji bilgisi ekranı

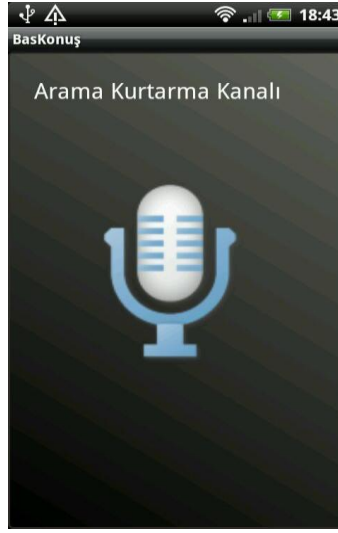
Uygulamada Şekil 4.23’de görüldüğü gibi arama kurtarma kanalı, itfaiye kanalı, koordinatör kanalı olmak üzere üç kanal tanımlanmıştır:

Kanallar, önceden tanımlanmış birbirinden farklı portlar kullanılarak ayrılmıştır. Uygulamayı kullanan bir kullanıcı istediği kanala girebilmektedir. Kanala giren bir kullanıcı ilgili kanaldaki kullanıcılara ses verisini gönderebilmekte ve diğer kullanıcıların ses verisini alabilmektedir. Bir kanaldaki kullanıcı, konuşma başlattığında o kanaldaki diğer tüm kullanıcıların dinleme moduna geçmesi için komut gönderir kanalı kullanan diğerleri dinleyici moduna geçer. Kanalda konuşan varsa bir başkası kanalda konuşamaz, yani aynı anda sadece bir kişi kanalda konuşan, diğerleri dinleyici olabilir.

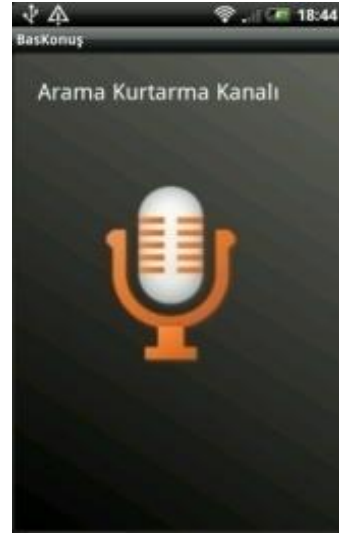
Şekil 4.26, konuşmanın yapılmadığı ya da ses verisinin alınmadığı **normal durum** ekranını göstermektedir.Şekil 4.27, kanalda herhangi bir kullanıcı konuşmaya başladığında **dinleme durumuna** geçildiği ve kullanıcının sesinin alındığı ekran görüntüsüdür. Şekil 4.28, kullanıcının ekrana dokunmasıyla değişen **konuşma durumundaki** ekran görüntüsüdür.



Şekil 4.26. Normal durum ekranı



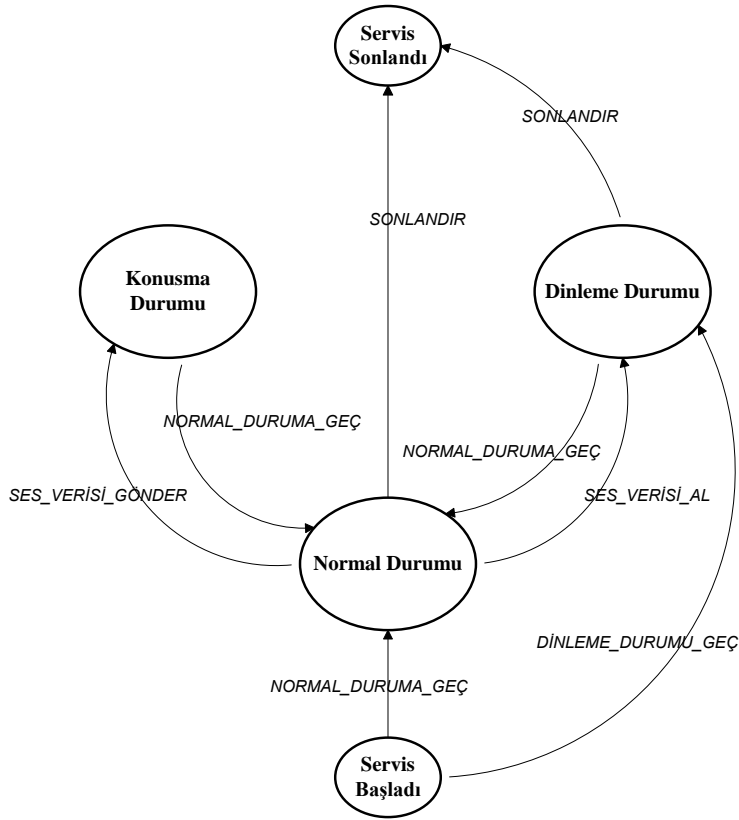
Şekil 4.27. Dinleme durum ekranı



Şekil 4.28. Konuşma durumu ekranı

#### 4.3.1. Yazılım tasarımı

Uygulama yazılım detayları şu şekildedir; uygulamada bir kanala girildiğinde konuşma başlatılabilmesi ve ses verisinin alınabilmesi için Android servis kullanımına ihtiyaç duyulmuştur. Bu durumları kontrol eden arka planda çalışacak servis geliştirilmiştir. Android servisi kanala girildiğinde çalışır ve kanaldan çıkıldığında kapatılır. Çalışma durumu olarak **dinleme durumu**, **konuşma durumu** ve kanalda herhangi bir iletimin olmadığı **normal durumlarından** birinde olabilir. Şekil 4.29'da servise ait sonlu durum makinesi gösterilmiştir. Servisin durumu ağdan gelen komutlarla veya kullanıcı komutlarıyla değiştirilebilir. Kullanıcının vereceği komutlarla **konuşma durumuna** veya **normal duruma** geçebilir. Kanala ait portlarda dinleme yaparak ağdan gelen komutlarla **dinleme durumuna** ve **normal duruma** geçebilir. Örneğin bir kullanıcının ağdan göndereceği *ses verisi al* komutunun alınmasıyla **dinleme durumuna** geçer ve ağdan gelen ses paketleri cihaz hoparlöründen ses olarak dış ortama verilir. Aynı şekilde kullanıcının vereceği *ses gönder* komutuyla **normal durumdan konuşma durumuna** geçer.



Şekil 4.29. Android bas konuş uygulaması servis durum makinesi

Uygulama ana ekranında herhangi bir kanalın seçilmesiyle kanala girilir. Kanala girildiğinde arka planda çalışan Android servisi başlar ve ağdan ses verisinin gelip gelmediği kontrol edilir. Gelen veri yoksa **normal duruma**, aksi durumda kanalda aktif olarak konuşan vardır ve **dinleme durumuna** geçilir.

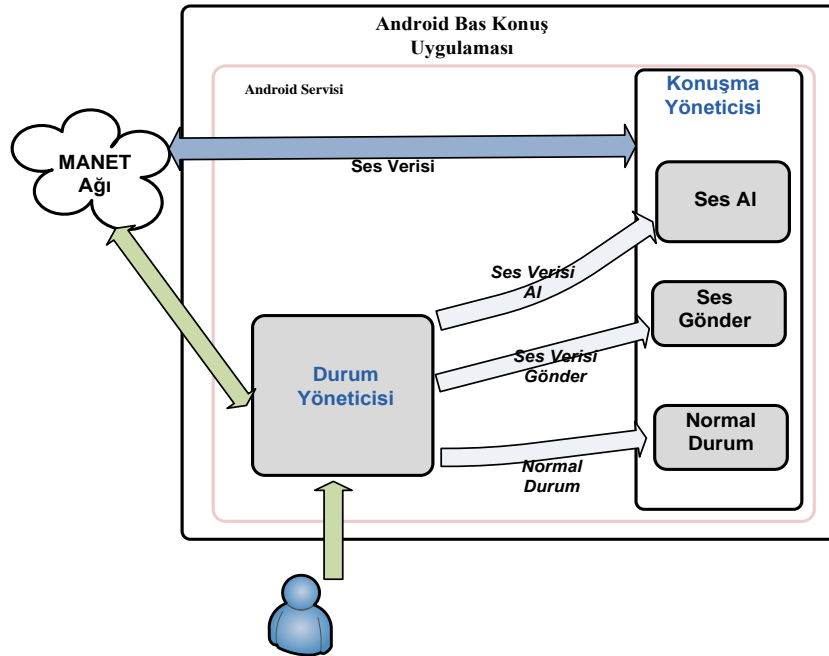
Normal Durum - Konusma Durumu geçişi, kullanıcının ekrandaki mikrofon tuşuna basılı tutmasıyla gerçekleşir. Kanaldaki diğer kullanıcılara *ses verisi al* komutu gönderilir.

Konusma Durumu - Normal Durum geçişi, kullanıcının ekrandaki mikrofon tuşunu bırakmasıyla gerçekleşir.

Normal Durum - Dinleme Durumu geçişi, ağdan gelen *ses verisi al* komutuyla olur. Ağ içinde kanalda bulunan bir kullanıcı, kanaldaki diğer kullanıcılara *ses verisi al* komutu göndermiştir.

Dinleme Durumu - Normal Durum geçişi, ağdan gelen *normal duruma geç* komutuyla olur. Kanalda konuşmaya devam eden kullanıcı, kanaldaki diğer kullanıcılara *normal duruma geç* komutunu göndermiştir. Kanaldan **dinleme durumu** veya **normal durumda** çıkılabilir. Kanaldan çıkılmasıyla beraber servis sonlandırılır.

Android bas konuş uygulaması yazılım tasarımı blok diagramı Şekil 4.30'da gösterilmiştir. Yazılımda Android servisinin kontrolünde çalışan Durum yöneticisi ve Konuşma yöneticisi tanımlanmıştır.



Şekil 4.30. Android bas konuş uygulaması yazılım tasarımı blok diagramı

Durum yöneticisi, kanala girildiğinde Android servisinin başlatılmasıyla oluşturulur. Uygulamanın durum değişimlerini yönetir ve aynı zamanda konuşma yöneticisi durumlarını da değiştirir. Servisin başlatılmasıyla, ilk olarak durumun belirlenmesi için ilgili kanalın ses alma portunda UDP socket açılarak gelen ses verisinin olup olmadığı testi yapılır. Açılan sokete 1000 ms boyunca herhangi bir ses paketi gelmezse socket zaman aşımına uğrar ve **normal duruma** geçilir. Gelen ses paketi var ise kanalda konuşan vardır ve **dinleme durumuna** geçilir. Diğer yandan ağdan gelen komutları almak için komut dinleme portunda UDP socket açılarak komut beklenir.

Ağdan gelecek ses *verisi al* ve *normal duruma geç* komutlarının alınması ve değerlendirilmesi durum yöneticisinde olur. Kullanıcı grafik arayüzünden girdiler **konuşma durumuna** ve **normal duruma** geçişleri sağlar. Kullanıcı ekrandaki konuşma tuşuna bastığında UDP socket açılarak ağa *ses verisi al* komutu gönderilir. Durum yöneticisi, konuşma yöneticisinin durumunu ses verisi göndermesi için değiştirir. Kullanıcı, konuşma tuşunu bıraktığında ağa *normal duruma geç* komutunu gönderir. Konuşma yöneticisinin normal duruma geçmesini sağlar ve mikrofondan alınan ses verisinin ağa gönderimi durdurulur. Durum yöneticisinin kullanılmasıyla, uygulama durum geçişlerinin tanımsız duruma geçişi engellenmiş olur. **Normal duruma** geçiş daha önce durum geçişini sağlayan etki yani ağdan gelen komut ya da kullanıcı ile olur. Örneğin kullanıcı konuşma yaptığı sırada ağdan gelecek *ses verisi al* komutu etkisiz olacaktır ya da dinleme durumundayken kullanıcının girdileri etkisiz olacaktır.

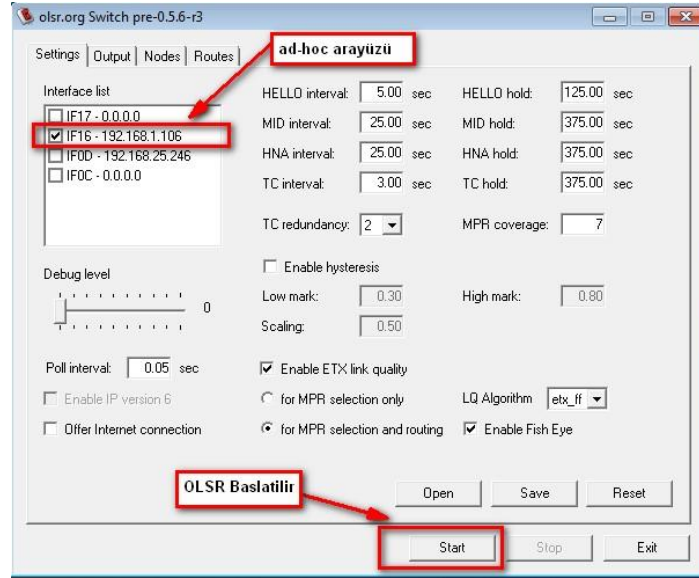
Konuşma yöneticisi, durum yöneticisinin kontrolünde ses alışı verisini yönetir. Durum geçişlerine göre Android ses kartına erişim alt yapısını kullanarak mikrofondan ses verisini alıp ağa göndermeyi ve ağdan alınan ses paketlerinin hoparlörden dış ortama verilmesini sağlar. Ayrıca uygulamada, giden ve gelen ses paketlerini Speex kodeği ile kodlama ve kod çözme seçeneği vardır. Böylece ağ içerisinde dolaşan birim zamandaki veri miktarı azaltılacaktır. **Dinleme durumunda** ilgili kanala ait ses dinleme portunda ses paketlerini alan UDP socket açılır ve alınan ses paketleri hoparlöre verilir. Ağdan gelecek *normal duruma geç* komutuyla **normal duruma** geçilir ve ses verisinin alındığı socket kapatılır, ses alma durdurulur. Hoparlöre ses verisi verilmez. **Normal durumdayken** kullanıcının ekrandaki tuşa basmasıyla ilgili kanala ait ses verisinin gönderileceği portta UDP socket açılır ortamdaki ses verisi mikrofondan alınarak ağa gönderilir.

## **BÖLÜM 5. SINAMA ORTAMI KURULMASI VE TEST SONUÇLARI**

Mobil ad-hoc ağları kurularak cihazlar üzerinde yönlendirmenin gerçekleştirildiğini ve iletişimin sağlandığını göstermek için geliştirilen yazılımlar ile bazı testler yapılmıştır. Test ortamında ad-hoc ağının kurulumu için HTC Desires S, ASUS Nexus 7 tablet, Ubuntu ve Windows 7 işletim sistemi olan dizüstü bilgisayarlar kullanılmıştır. Cihazlar üzerinde, OLSR yönlendirme protokolü çalıştırılmış üzerlerinden geçen ağ paketlerinin yönlendirilerek hedef noktalarına gitmeleri sağlanmıştır. OLSR yönlendirme protokolünün çalışması işletim sisteminden bağımsız olduğu için Windows veya Android tabanlı işletim sistemli cihazlar arasında MANET kurulması problem oluşturmayacaktır.

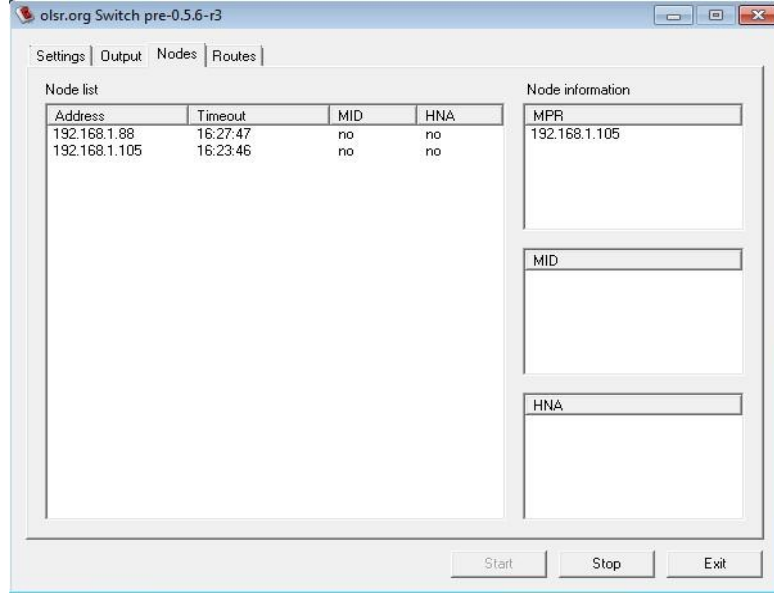
Windows 7 ve Ubuntu işletim sistemli dizüstü bilgisayarlarda kablosuz ağa bağlanması için kablosuz USB adaptörler kullanılmıştır. Bu adaptörler konfigürasyonların daha hızlı yapılmasını sağlayarak pratiklik kazandıracaktır. Android cihazlarda varolan kablosuz alt yapısı kullanılmıştır.

Windows 7 işletim sistemli dizüstü bilgisayarlarda OLSR Switch yazılımı ile yönlendirme protokolü çalıştırılmıştır. Şekil 5.1'de OLSR Switch yazılımı konfigürasyon ekranı gösterilmiştir. Programın çalıştırılması şu şekildedir; Ekranda ad-hoc ağı için kullanılan kablosuz ağ arayüzü seçilir. Aynı ekranda başlat butonu ile gerekli modülleri yükleyerek OLSR başlatılır.



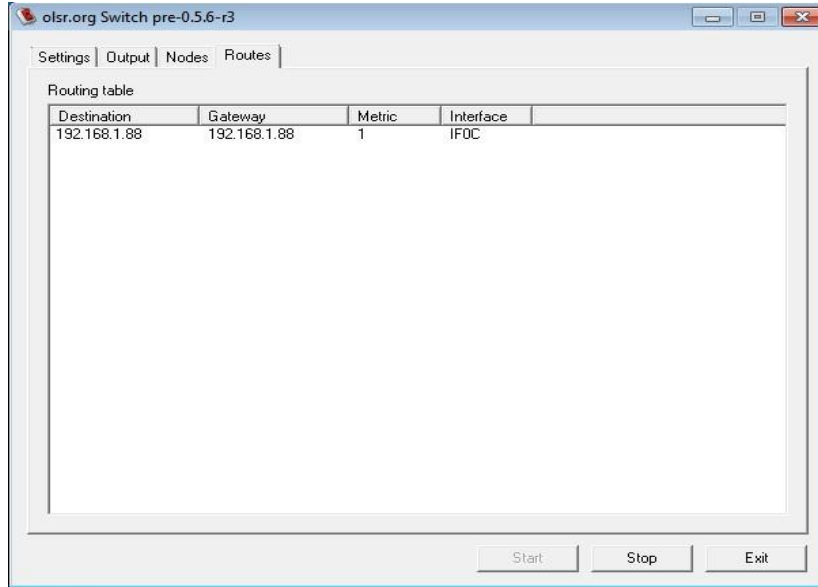
Şekil 5.1. OLSR Switch yazılımı konfigürasyon ekranı

Şekil 5.2'de görüldüğü gibi OLSR'nin başlatılmasıyla HELLO paketlerinin gönderilmesiyle bağlı uç noktalar ve komşuları bulunur ve “Nodes” sekmesi altında görülebilir.



Şekil 5.2. OLSR Switch yazılımı çalıştırılması - Bağlı uç nokta IP bilgileri

Şekil 5.3'de görüldüğü gibi “Routes” sekmesi altında uç noktalara erişmesinde yönlendirici olan IP'ler gösterilir.



Şekil 5.3. OLSR Switch yazılımı çalıştırılması - Yönlendirici uç nokta IP bilgileri

Ubuntu işletim sistemli dizüstü bilgisayarlarda paket yöneticisi kullanılarak OLSR yönlendirme protokolü yazılımı kurulabilir. Terminalden yönlendirme protokolü çalıştırılır. Oluşan ağ topolojisi ile ilgili bilgiler terminal ekranında görülebilir.

Geliştirilen bas konuş uygulaması HTC ve Nexus 7 cihazlarına yüklenmiştir. Yüklenen bu uygulama kullanılarak kablosuz ağa bağlanılmış ve yönlendirme protokolü çalıştırılmıştır. Oluşan ağ topolojisi bilgileri uygulamanın menü kısmında görülebilmektedir.

Yapılan testlerde, cihazların ağ adaptörleri el ile ayarlanarak aynı ad-hoc ağına bağlanmaları sağlanmıştır ve aynı SSID, subnet ayarı yapılmış fakat farklı IP adresleri verilmiştir. Bu konfigürasyonlar yapıldıktan sonra sabit yerlerde tutulan cihazlar ile MANET ağı kurulmuştur.

Test için kullanılan HTC Desires S, ASUS Nexus 7 tablet, Windows 7 veya Ubuntu işletim sistemi olan dizüstü bilgisayarlar anlatımı kolaylaştırmak için sırayla Cihaz 1, Cihaz 2, Cihaz 3 olarak adlandırılmıştır ve Tablo 5.1'de IP adresleri belirtilmiştir. Burada Ubuntu bilgisayar ile yapılan testler anlatılmamıştır Cihaz 3 ile belirtilen Windows 7 bilgisayardır.

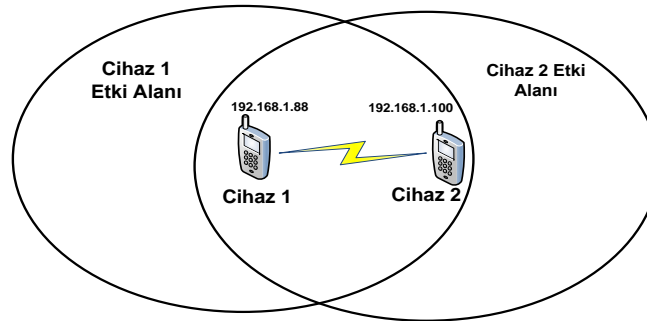


Tablo 5.1. Cihaz IP bilgileri

Cihaz 1	192.168.1.88
Cihaz 2	192.168.1.100
Cihaz 3	192.168.1.12

## 5.1. İki Cihaz Arasında MANET

Cihaz 1 ve Cihaz 2 ile birbirlerinin etki alanları içerisinde olacak şekilde MANET oluşturulmuştur (Şekil 5.4).



Şekil 5.4. Test - İki cihaz arasında MANET

Ağın oluşmasıyla birlikte OLSR yönlendirme protokolü çalışmaktadır. Şekil 5.5 ve Şekil 5.6'da görüldüğü gibi bağlılık tablolarında cihazların IP'leri görülmektedir. Cihazlar birbirlerine direkt olarak bağlıdır.

Ağ Topolojisi Bilgisi						
Table: Links						
Local IP	Remote IP	Hyst.	LQ	NLQ	Cost	
192.168.1.88	192.168.1.100	0.00	1.000	1.000	1.000	
Table: Topology						
Dest. IP	Last hop IP	LQ	NLQ	Cost		
192.168.1.100	192.168.1.88	1.000	1.000	1.000		
192.168.1.88	192.168.1.100	1.000	1.000	1.000		
Table: HNA						
Destination		Gateway				
Table: MID						
IP address		Aliases				
Table: Routes						
Destination	Gateway IP	Metric	ETX	Interface		
192.168.1.100/32	192.168.1.100	1	1.000	eth0		
Table: Gateways						
Status	Gateway IP	ETX	Hopcnt	Uplink	Downlink	Prefix
Downlink	IPv4	IPv6	Prefix			
Table: Interfaces						
Name	State	MTU	WLAN	Src-Address	Mask	Dst-Address
eth0	UP	1472	Yes	192.168.1.88	255.255.255.0	192.168.1.255
Table: Neighbors						
IP Address	SYM	MPR	MPRS	Will.		
2 Hop Interface Address						
192.168.1.100	YES	NO	NO	3		

Şekil 5.5. Cihaz 1 bağlılık tablosu

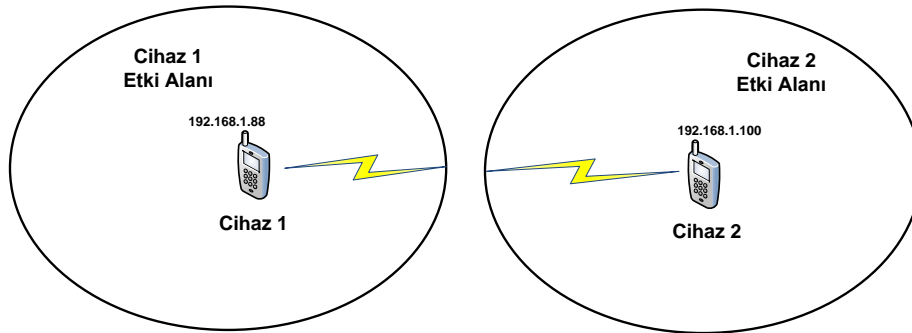
Ağ Topolojisi Bilgisi						
HTTP/1.0 200 OK						
Content-type: text/plain						
Table: Links						
Local IP	Remote IP	Hyst.	LQ	NLQ	Cost	
192.168.1.100	192.168.1.88	0.00	1.000	1.000	1.000	
Table: Topology						
Dest. IP	Last hop IP	LQ	NLQ	Cost		
192.168.1.100	192.168.1.88	1.000	1.000	1.000		
192.168.1.88	192.168.1.100	1.000	1.000	1.000		
Table: HNA						
Destination		Gateway				
Table: MID						
IP address		Aliases				
Table: Routes						
Destination	Gateway IP	Metric	ETX	Interface		
192.168.1.88/32	192.168.1.88	1	1.000	wlan0		
Table: Gateways						
Status	Gateway IP	ETX	Hopcnt	Uplink	Downlink	Prefix
Table: Interfaces						
Name	State	MTU	WLAN	Src-Address	Mask	Dst-Address
wlan0	UP	1472	Yes	192.168.1.100	255.255.255.0	192.168.1.255
Table: Neighbors						
IP Address	SYM	MPR	MPRS	Will.		
2 Hop Interface Address						
192.168.1.88	YES	NO	NO	3		

Şekil 5.6. Cihaz 2 bağlılık tablosu

Her iki cihazda da bas konuş uygulaması çalıştırılıp ses verisi iletimi ile ilgili denemeler yapılmıştır. Ses verisinin başarılı bir şekilde iletildiği görülmüştür.

## 5.2. İki Cihaz Etki Alanları Dışında MANET

Cihaz 1 ve Cihaz 2 birbirlerinin etki alanları dışında olacak şekilde konumlandırılmıştır (Şekil 5.7).



Şekil 5.7. Test - İki cihaz etki alanı dışında

Bu konumdayken cihazlara ait ağ topolojileri bilgileri ekran görüntüleri Şekil 5.8 ve Şekil 5.9'da gösterilmiştir. Buna göre bağlılık tablosunda cihazların iletişimde olduğu herhangi bir IP gözükmemektedir.

15:21

**Ağ Topolojisi Bilgisi**

HTTP/1.0 200 OK  
Content-type: text/plain

Local IP	Remote IP	Hyst.	LQ	NLQ	Cost

Table: Topology

Dest. IP	Last hop IP	LQ	NLQ	Cost

Table: HNA

Destination	Gateway

Table: MID

IP address	Aliases

Table: Routes

Destination	Gateway IP	Metric	ETX	Interface

Table: Gateways

Status	Gateway IP	ETX	Hopcnt	Uplink
DownInk	IPv4	IPv6	Prefix	

Table: Interfaces

Name	State	MTU	WLAN	Src-Address	Mask
eth0	UP	1472	Yes	192.168.1.88	255.255.255.0
wlan0	DOWN			192.168.1.255	

Table: Neighbors

IP Address	SYM	MPR	MPRS	Will.
2 Hop Interface Address				

Şekil 5.8. Etki alanı dışında Cihaz 1 bağlılık tablosu

3:22

**Ağ Topolojisi Bilgisi**

HTTP/1.0 200 OK  
Content-type: text/plain

Local IP	Remote IP	Hyst.	LQ	NLQ	Cost

Table: Topology

Dest. IP	Last hop IP	LQ	NLQ	Cost

Table: HNA

Destination	Gateway

Table: MID

IP address	Aliases

Table: Routes

Destination	Gateway IP	Metric	ETX	Interface

Table: Gateways

Status	Gateway IP	ETX	Hopcnt	Uplink	DownInk	IPv4	IPv6	Prefix

Table: Interfaces

Name	State	MTU	WLAN	Src-Address	Mask	Dst-Address
wlan0	DOWN					

Table: Neighbors

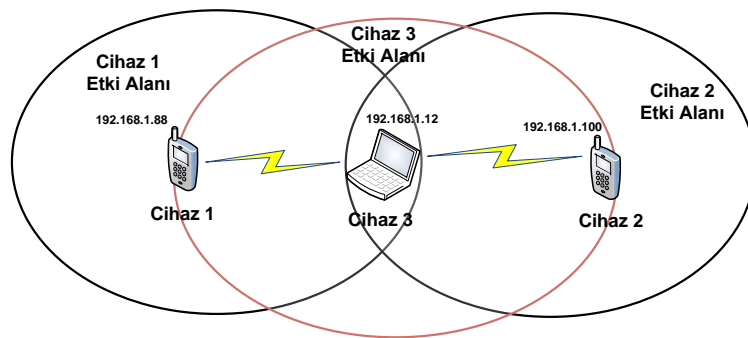
IP Address	SYM	MPR	MPRS	Will.
2 Hop Interface Address				

Şekil 5.9. Etki Alanı dışında Cihaz 2 bağlılık tablosu

İki cihaz getirilen son konumlarında, aralarındaki mesafeden dolayı kablosuz ağ oluşturamadı.

### 5.3. İki Android Cihaz ve Dizüstü Bilgisayar Arasında MANET

Cihaz 1 ve Cihaz 2 birbirlerinin etki alanları dışındayken cihazları etki alanı içine alacak şekilde Cihaz 3 konumlandırıldı (Şekil 5.10) ve “OLSR switch” programı ile OLSR protokolü çalıştırılarak Cihaz 3'ün yönlendirici olarak çalışması sağlandı.



Şekil 5.10. Test - Cihaz 1, Cihaz 2 ve Cihaz 3 ile MANET ağı kurulması

MANET ağı kurulduğunda Cihaz 1 ve Cihaz 2'ye ait ağ topolojisi ve uç noktaların bağlılık bilgileri ekranı Şekil 5.11 ve Şekil 5.12'de görüldüğü gibidir.

Ağ Topolojisi Bilgisi					
Table: Links					
Local IP	Remote IP	Hyst.	LQ	NLQ	Cost
192.168.1.88	192.168.1.12	0.00	0.553	0.925	1.954
Table: Topology					
Dest. IP	Last hop IP	LQ	NLQ	Cost	
192.168.1.88	192.168.1.12	0.819	0.553	2.206	
192.168.1.100	192.168.1.12	1.000	0.913	1.094	
192.168.1.12	192.168.1.88	0.553	0.925	1.954	
192.168.1.12	192.168.1.100	0.913	1.000	1.094	
Table: HNA					
Destination	Gateway				
Table: MID					
IP address	Aliases				
Table: Routes					
Destination	Gateway IP	Metric	ETX	Interface	
192.168.1.12/32	192.168.1.12	1	1.954	eth0	
192.168.1.100/32	192.168.1.12	2	3.048	eth0	
Table: Gateways					
Status	Gateway IP	ETX	Hopcnt	Uplink	
DownInk	IPv4	IPv6	Prefix		
Table: Interfaces					
Name	State	MTU	WLAN	Src-Address	Mask
eth0	UP	1472	Yes	192.168.1.88	255.255.255.0
Table: Neighbors					

Şekil 5.11. Cihaz 1 bağlılık durum bilgileri

Ağ Topolojisi Bilgisi					
HTTP/1.0 200 OK					
Content-type: text/plain					
Table: Links					
Local IP	Remote IP	Hyst.	LQ	NLQ	Cost
192.168.1.100	192.168.1.12	0.00	1.000	1.000	1.000
Table: Topology					
Dest. IP	Last hop IP	LQ	NLQ	Cost	
192.168.1.88	192.168.1.12	1.000	0.827	1.208	
192.168.1.100	192.168.1.12	1.000	1.000	1.000	
192.168.1.12	192.168.1.88	0.827	0.905	1.334	
192.168.1.12	192.168.1.100	1.000	1.000	1.000	
Table: HNA					
Destination	Gateway				
Table: MID					
IP address	Aliases				
Table: Routes					
Destination	Gateway IP	Metric	ETX	Interface	
192.168.1.12/32	192.168.1.12	1	1.000	wlan0	
192.168.1.88/32	192.168.1.12	2	2.306	wlan0	
Table: Gateways					
Status	Gateway IP	ETX	Hopcnt	Uplink	DownInk
IPv4	IPv6	Prefix			
Table: Interfaces					
Name	State	MTU	WLAN	Src-Address	Mask
wlan0	UP	1472	Yes	192.168.1.100	255.255.255.0
Table: Neighbors					
IP Address	SYM	MPR	MPRS	Will.	
2 Hop Interface Address					
192.168.1.12	YES	YES	NO	3	192.168.1.88

Şekil 5.12. Cihaz 2 bağlılık durum bilgileri

Cihaz 1 direkt olarak Cihaz 3 ile bağlantıdadır aynı şekilde Cihaz 2, Cihaz 3 ile direkt bağlantıdadır. Bu sırada Cihaz 3’de komut satırında “route print” komutu yazılarak yönlendirme tablosu Şekil 5.13’deki gibi görüntülenmiştir. Buna göre Cihaz 3 diğerlerine direkt olarak bağlıdır.

```

C:\Users>route print
=====
Interface List
12...64 70 02 16 da 39 .....TP-LINK Wireless USB Adapter
12...c 65 94 b8 07 ed .....DHL501 Wireless-N WLAN Half-Mini Card
11...10 4d a2 aa a8 63 .....Realtek RTL8168D/8111D Family PCI-E Gigabit Ethernet NIC (NDIS 6.20)
1.....Software Loopback Interface 1
30...00 00 00 00 00 00 e0 Microsoft ISATAP Adapter
13...00 00 00 00 00 00 e0 Teredo Tunneling Pseudo-Interface
28...00 00 00 00 00 00 e0 Microsoft ISATAP Adapter #3
29...00 00 00 00 00 00 e0 Microsoft ISATAP Adapter #4
=====

IPv4 Route Table
=====
Active Routes:
Network Destination        Netmask          Gateway          Interface        Metric
127.0.0.0                  255.0.0.0        On-link          127.0.0.1         306
127.255.255.255           255.255.255.255 On-link          127.0.0.1         306
192.168.1.0                255.255.255.0   On-link          192.168.1.12     286
192.168.1.12              255.255.255.255 On-link          192.168.1.12     286
192.168.1.88              255.255.255.255 192.168.1.88    192.168.1.12     30
192.168.1.100             255.255.255.255 192.168.1.100   192.168.1.12     30
192.168.1.255             255.255.255.255 On-link          192.168.1.12     286
224.0.0.0                 240.0.0.0        On-link          192.168.1.12     286
255.255.255.255           255.255.255.255 On-link          127.0.0.1         306
255.255.255.255           255.255.255.255 On-link          192.168.1.12     286
=====
Persistent Routes:
Network Address      Netmask  Gateway Address  Metric
0.0.0.0              0.0.0.0   192.168.1.1      Default
=====

IPv6 Route Table
=====
Active Routes:
If Metric Network Destination      Gateway
1 306 ::1/128 On-link
17 286 fe80::c64 On-link
17 286 fe80::2098:4d12:6e23:519e/128 On-link
1 306 ff00::/8 On-link
17 286 ff00::/8 On-link
=====
Persistent Routes:
None

```

Şekil 5.13. Cihaz 3 yönlendirme IP bilgileri

Cihaz 1'e ait "Routes" tablosunda, 192.168.1.12 IP'li Cihaz 3'ün ağ geçidi görevi üstlendiği görülmektedir. Cihaz 1, Cihaz 2'ye ağ üzerinden paket gönderdiğinde Cihaz 3 üzerinden hedefine ulaşacaktır. Aynı durum Cihaz 2 için de geçerlidir. Cihaz 2'den Cihaz 1'e gönderilen paket Cihaz 3 üzerinden geçerek hedefine varacaktır.

Cihaz 1 ve Cihaz 2’de bas-konuş uygulaması çalıştırılmıştır. Cihaz 1’deki sesin Cihaz 2’ye iletilmesi testi yapılmıştır. Bu sırada Cihaz 3’de wireshark ağ protokol analizi programı çalıştırılarak üzerinden geçen ses paketleri izlenmiştir. Şekil 5.14'de Hedef IP adresi Cihaz 2'ye ait olmasına rağmen "Ethernet" alanına bakıldığında Cihaz 3'ün MAC adresi görülmektedir ve ses paketi başlangıçta Cihaz 3'e gelmiştir.

No.	Time	Source	Destination	Protocol	Length	Info
2365	433.825142	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2366	433.825165	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2367	433.828001	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2368	433.834012	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2369	433.834012	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2370	433.834848	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2371	433.838731	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2372	433.838756	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2373	433.840621	192.168.1.88	192.168.1.12	UDP	57	Source port: 47176 Destination port: 50002
2374	433.845376	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2375	433.845399	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2376	433.847750	192.168.1.88	192.168.1.12	UDP	57	Source port: 47176 Destination port: 50002
2377	433.852626	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2378	433.852649	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2379	433.853643	192.168.1.88	192.168.1.12	UDP	57	Source port: 47176 Destination port: 50002

Frame 2368: 57 bytes on wire (456 bits) - 57 bytes captured (456 bits)
Ethernet II, Src: d4:20:6d:49:e8:be (d4:20:6d:49:e8:be), Dst: 64:70:02:16:da:39 (64:70:02:16:da:39)
Internet Protocol Version 4, Src: 192.168.1.88 (192.168.1.88), Dst: 192.168.1.100 (192.168.1.100)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
Total Length: 43
Identification: 0x0000 (0)
Flags: 0x02 (Don't Fragment)
Fragment offset: 0
Time to live: 64
Protocol: UDP (17)
Header checksum: 0xb6b5 [correct]
Source: 192.168.1.88 (192.168.1.88)
Destination: 192.168.1.100 (192.168.1.100)
User Datagram Protocol, Src Port: 47176 (47176), Dst Port: 50002 (50002)
Source port: 47176 (47176)
Destination port: 50002 (50002)
Length: 23
Checksum: 0x1f91 [validation disabled]
[good Checksum: False]
[bad Checksum: False]
Data (15 bytes)
Data: 13976f973e9faaee83c1c46ecb9960
[Length: 15]

Şekil 5.14. Yönlendirilen paketin Wireshark görüntüsü-1

Şekil 5.15'de Cihaz 3'e gelen bu ses paketinin Cihaz 2'ye gönderilmesi görülmektedir. "Ethernet" alanına bakıldığında kaynak MAC adresi Cihaz 3'e ve hedef MAC adresi Cihaz 2'ye aittir. Paket, kaynak adresi değiştirilmeden hedefe yönlendirilmektedir.

No.	Time	Source	Destination	Protocol	Length	Info
2365	433.825142	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2366	433.825165	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2367	433.828001	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2368	433.834012	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2369	433.834012	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2370	433.834848	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2371	433.838731	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2372	433.838756	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2373	433.840621	192.168.1.88	192.168.1.12	UDP	57	Source port: 47176 Destination port: 50002
2374	433.845376	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2375	433.845399	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2376	433.847750	192.168.1.88	192.168.1.12	UDP	57	Source port: 47176 Destination port: 50002
2377	433.852626	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2378	433.852649	192.168.1.88	192.168.1.100	UDP	57	Source port: 47176 Destination port: 50002
2379	433.853643	192.168.1.88	192.168.1.12	UDP	57	Source port: 47176 Destination port: 50002

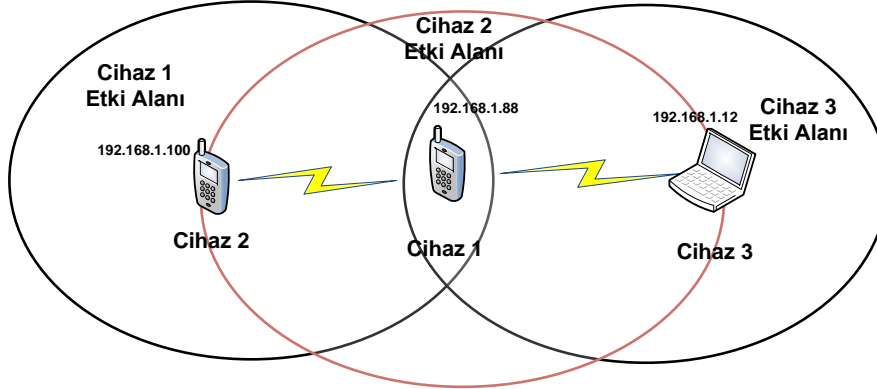
  

Frame 2369: 57 bytes on wire (456 bits) - 57 bytes captured (456 bits)
Ethernet II, Src: 64:70:02:16:da:39 (64:70:02:16:da:39), Dst: ac:22:0b:45:86:09 (ac:22:0b:45:86:09)
Destination: ac:22:0b:45:86:09 (ac:22:0b:45:86:09)
Source: 64:70:02:16:da:39 (64:70:02:16:da:39)
Type: IP (0x0800)
Internet Protocol Version 4, Src: 192.168.1.88 (192.168.1.88), Dst: 192.168.1.100 (192.168.1.100)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
Total Length: 43
Identification: 0x0000 (0)
Flags: 0x02 (Don't Fragment)
Fragment offset: 0
Time to live: 63
Protocol: UDP (17)
Header checksum: 0xb7b5 [correct]
Source: 192.168.1.88 (192.168.1.88)
Destination: 192.168.1.100 (192.168.1.100)
User Datagram Protocol, Src Port: 47176 (47176), Dst Port: 50002 (50002)
Source port: 47176 (47176)
Destination port: 50002 (50002)
Length: 23
Checksum: 0x1f91 [validation disabled]
[good Checksum: False]
[bad Checksum: False]
Data (15 bytes)
Data: 13976f973e9faaee83c1c46ecb9960
[Length: 15]

Şekil 5.15. Yönlendirilen paketin Wireshark görüntüsü-2

Cihaz 2'deki ses verisinin Cihaz 1'e iletilmesi testlerinde de aynı işlemler uygulanarak aynı sonuçlar elde edilmiştir.

Cihazlar bu konumlarındayken Cihaz 3 ile Cihaz 1 yer değiştirilmiştir. Şekil 5.16'daki gibi Cihaz 1 yönlendirici görevini üstlenecek şekilde testler tekrarlanmıştır.



Şekil 5.16. Test - Cihaz 1 yönlendirici, MANET ağı

Ağ kurulduğunda Cihaz 1 ve Cihaz 2'nin ağ topoloji bilgileri Şekil 5.17 ve Şekil 5.18'de görüldüğü gibidir.

Ağ Topolojisi Bilgisi					
Table: Links					
Local IP	Remote IP	Hyst.	LQ	NLQ	Cost
192.168.1.88	192.168.1.12	0.00	1.000	1.000	1.000
192.168.1.88	192.168.1.100	0.00	1.000	1.000	1.000
Table: Topology					
Dest. IP	Last hop IP	LQ	NLQ	Cost	
192.168.1.88	192.168.1.12	1.000	1.000	1.000	
192.168.1.12	192.168.1.88	1.000	1.000	1.000	
192.168.1.100	192.168.1.88	1.000	1.000	1.000	
192.168.1.88	192.168.1.100	1.000	1.000	1.000	
Table: HNA					
Destination	Gateway				
Table: MID					
IP address	Aliases				
Table: Routes					
Destination	Gateway IP	Metric	ETX	Interface	
192.168.1.12/32	192.168.1.12	1	1.000	eth0	
192.168.1.100/32	192.168.1.100	1	1.000	eth0	
Table: Gateways					
Status	Gateway IP	ETX	Hopcnt	Uplink	Downlink
Table: Interfaces					
Name	State	MTU	WLAN	Src-Address	Mask
eth0	UP	1472	Yes	192.168.1.88	255.255.255.0
Table: Neighbors					
IP Address	SYM	MPR	MPRS	Will	
192.168.1.12	NO	NO	NO	6	192.168.1.88
192.168.1.88	YES	YES	NO	3	192.168.1.12

Şekil 5.17. Cihaz 1 Yön.-Ch1 Görüntüsü

Ağ Topolojisi Bilgisi					
HTTP/1.0 200 OK					
Content-type: text/plain					
Table: Links					
Local IP	Remote IP	Hyst.	LQ	NLQ	Cost
192.168.1.100	192.168.1.88	0.00	1.000	1.000	1.000
192.168.1.100	192.168.1.12	0.00	1.000	0.000	INFINITE
Table: Topology					
Dest. IP	Last hop IP	LQ	NLQ	Cost	
192.168.1.88	192.168.1.12	1.000	1.000	1.000	
192.168.1.12	192.168.1.88	1.000	1.000	1.000	
192.168.1.100	192.168.1.88	1.000	1.000	1.000	
192.168.1.88	192.168.1.100	1.000	1.000	1.000	
Table: HNA					
Destination	Gateway				
Table: MID					
IP address	Aliases				
Table: Routes					
Destination	Gateway IP	Metric	ETX	Interface	
192.168.1.12/32	192.168.1.88	2	2.000	wlan0	
192.168.1.88/32	192.168.1.88	1	1.000	wlan0	
Table: Gateways					
Status	Gateway IP	ETX	Hopcnt	Uplink	Downlink
Table: Interfaces					
Name	State	MTU	WLAN	Src-Address	Mask
wlan0	UP	1472	Yes	192.168.1.100	255.255.255.0
Table: Neighbors					
IP Address	SYM	MPR	MPRS	Will	
192.168.1.12	NO	NO	NO	6	192.168.1.88
192.168.1.88	YES	YES	NO	3	192.168.1.12

Şekil 5.18. Cihaz 1 Yönlendirici - Cihaz 2 Görüntüsü

Cihaz 2'de bas konuş uygulaması çalıştırılarak Cihaz 3'e ses paketleri gönderilmiştir. Cihaz 3'de Wireshark programı çalıştırılarak gelen paketler izlenmiştir. Gelen bir ses paketinin Wireshark programı ekran görüntüsü Şekil 5.19'da görüldüğü gibidir. Buna göre paketin kaynak IP adresi Cihaz 2'ye aittir ve hedef IP adresi Cihaz 3'ündür. "Ethernet" alanına bakıldığında kaynak MAC adresi Cihaz 1'e aittir yani paket Cihaz 1 üzerinden geçerek hedef adrese ulaşmıştır.

No.	Time	Source	Destination	Protocol	Length	Info
316	167.411668	192.168.1.100	192.168.1.12	UDP	57	Source port: 42797 Destination port: 50002
317	167.425905	192.168.1.100	192.168.1.12	UDP	57	Source port: 42797 Destination port: 50002
318	167.451414	192.168.1.100	192.168.1.12	UDP	57	Source port: 42797 Destination port: 50002
319	167.466530	192.168.1.100	192.168.1.12	UDP	57	Source port: 42797 Destination port: 50002
320	167.493901	192.168.1.100	192.168.1.12	UDP	57	Source port: 42797 Destination port: 50002

<b>Cihaz 2 IP Adresi</b>						
Frame 317: 57 bytes on wire (456 bits), 57 bytes captured (456 bits)						
Ethernet II, Src: d4:20:6d:49:e8:be (d4:20:6d:49:e8:be), Dst: 64:70:02:16:da:39 (64:70:02:16:da:39)						
Internet Protocol Version 4, Src: 192.168.1.100 (192.168.1.100), Dst: 192.168.1.12 (192.168.1.12)						
Version: 4						
Header length: 20 bytes						
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECT-Capable Transport))						
Total Length: 43						
Identification: 0x0000 (0)						
Flags: 0x02 (Don't Fragment)						
Fragment offset: 0						
Time to live: 63						
Protocol: UDP (17)						
Header checksum: 0xb801 [correct]						
Source: 192.168.1.100 (192.168.1.100)						
Destination: 192.168.1.12 (192.168.1.12)						
User Datagram Protocol, Src Port: 42797, Dst Port: 50002 (50002)						
source port: 42797 (42797)						
Destination port: 50002 (50002)						
Length: 23						
Checksum: 0x9835 [validation disabled]						
Data (15 bytes)						
Data: 10d02ed1d12b5f4999e0e319320ce						
[Length: 15]						

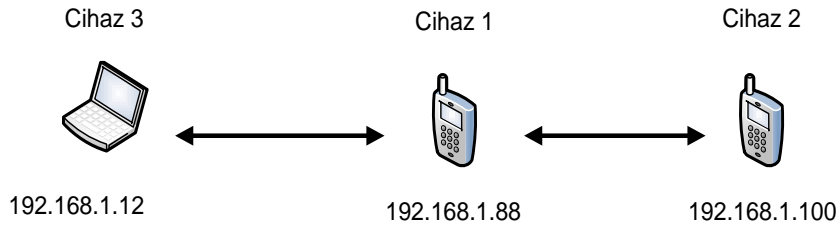
0000	64 70 02 16 da 39 d4 20 6d 49 e8 be 08 00 45 00	dp...9...mI...E..
0010	00 2b 00 00 40 00 3f 11 b8 01 c0 a8 01 64 c0 a8	...@?....d..
0020	01 0c a7 2d c3 52 00 17 98 35 10 d0 2e d1 d1 2b	...R...5....4
0030	5f 49 99 e0 0e 31 93 20 ce	...i...1..

Şekil 5.19. Cihaz 3'e gelen paketlerin Wireshark programındaki görüntüsü

#### 5.4. Ağ Performans Testleri

Üç cihaz ile kurulan MANET ağında ağ kurulum süresinin, band genişliğinin ve paket gecikme süresinin belirlenmesi ile ilgili testler yapılmıştır. Cihazların birbirlerini bulma süresi ~9 sn olarak belirlenmiştir. Herhangi bir cihaz ayrıldığında diğer cihazların bağlantı tablolarında bu bilgiyi güncellemesi ~12 sn'dir.

Band genişliğini belirlemek için pingb programı kullanılmıştır. Yazılım kaynaktan hedefe ICMP paketleri göndererek aldığı cevaba göre band genişliğini belirler. Şekil 5.20'de görüldüğü gibi Cihaz 1 yönlendirici durumunda çalışacak şekilde MANET ağı oluşturulmuştur.



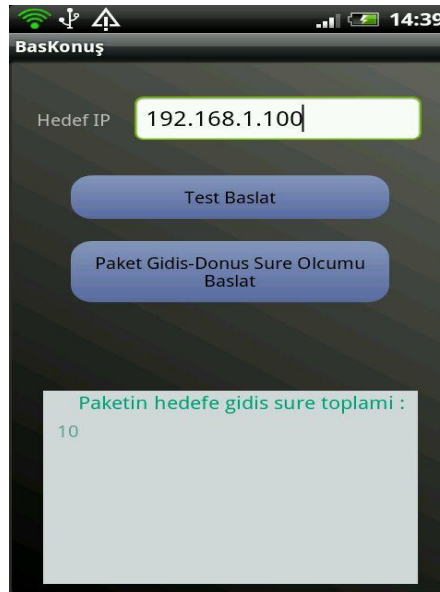
Şekil 5.20. Cihaz 1 yönlendirici durumunda test MANET ağı

Cihaz 3'de pingb yazılımı Cihaz 1 ve Cihaz 2 hedef alınarak ayrı ayrı çalıştırılarak band genişliği belirleme testleri yapılmıştır. Tablo 5.2'de kaynak ve hedef adresleri arasında elde edilen yaklaşık sonuçlar verilmiştir. Veri iletimini direkt olarak gerçekleştiren cihazlar arasında veri aktarım hızı yüksek olmasına rağmen bir uç noktanın yönlendirme yapmasıyla gerçekleştirilen haberleşmede veri aktarım hızı büyük oranda düştüğü görülmüştür.

Tablo 5.2. Pingb yazılımı ile veri aktarım hızı ölçüm sonuçları

Kaynak IP Adresi	Hedef IP Adresi	Band Genişliği (Kbps)
192.168.1.12	192.168.1.88	~33559
192.168.1.12	192.168.1.100	~1472

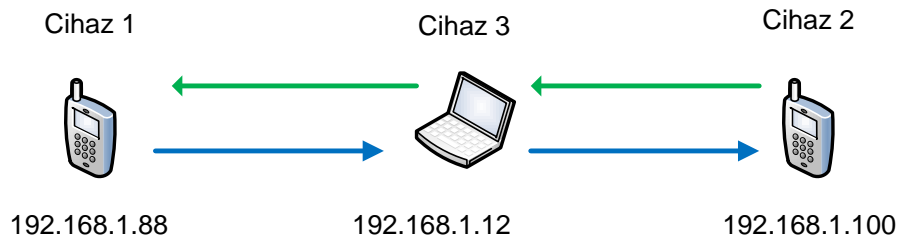
Ağ içinde hedefe gönderilen bir paketin gecikmesini belirlemek için test uygulaması geliştirilmiştir. Şekil 5.21'de ekran görünümü görülmektedir.



Şekil 5.21. Test uygulaması ekran görünümü

Gecikme süresinin belirlenmesi, hedefe gönderilen paketin hedeften tekrar kaynağa dönünceye kadar geçen süreye bağlı olarak belirlenmiştir. Şekil 5.22'deki gibi Cihaz 3 yönlendirici olarak çalışacak şekilde MANET ağı kurulmuştur.





Şekil 5.22. MANET ağı paket gecikme süresinin belirlenmesi

Cihaz 1'den gönderilen paketin Cihaz 2'ye ulaştıktan sonra tekrar Cihaz 1'e ulaşması ortalama 10.76 ms'dir. Bu durumda paketin bir uç noktanın yönlendirmesiyle hedefe varış süresi 5.38 ms'dir.

## **BÖLÜM 6. SONUÇLAR VE ÖNERİLER**

Doğal afetlerde (deprem, fırtına gibi) hücresel şebekelerin aşırı yüklenme, arızalanma gibi sebeplerle devre dışı kaldıkları bilinmektedir. Bu zaman dilimlerinde akıllı cihazların WiFi arayüzü kullanılarak hücresel sistemlere alternatif haberleşme alt yapısının geliştirilmesi bu tez kapsamında sunulmuştur. Günümüzde yaygın olarak kullanılan Android işletim sistemli akıllı cihazların ad-hoc modu etkinleştirilerek ve OLSR yönlendirme protokolü Android cihazlarda çalıştırılarak alternatif haberleşme sisteminin kurulumu gerçekleştirilmiştir. Oluşturulan MANET ağı üzerinde Android uygulama geliştirilerek uçtan uca çalışan ses haberleşmesi başarıyla gösterilmiştir. Kurulumu hızlı ve ucuz olan bu yöntemle arama kurtarma ekipleri arasında koordinasyonu sağlayabilecek alternatif bir haberleşme sisteminin prototipi başarıyla gerçekleştirilmiştir.

Çalışma kapsamında ad-hoc ağ kurulması için HTC Desire S, Nexus 7 (flo), Windows 7 ve Ubuntu işletim sistemli dizüstü bilgisayarlar kullanılmıştır. Dizüstü bilgisayarlar ile kablosuz ağa bağlanabilmesi için USB kablosuz adaptörler kullanılmıştır. Android cihazların kablosuz ağa bağlanması için varolan kablosuz ağ alt yapısından faydalanılmıştır. Bu kapsamda üreticilerin ürettiği Android işletim sistemli cihazlar ad-hoc modunu desteklemediğinden testler sırasında kullanılan ASUS Nexus 7’de özel kernel kullanılmasıyla ve HTC Desire S telefonunda “wpa\_supplicant” dosyasının değiştirilmesiyle ad-hoc modu etkin hale getirilmiştir. Cihazlar ile MANET’in kurulması için MANET Manager açık kaynak kodlu çatısı kullanılmıştır. Arama kurtarma ekipleri arasında sesli haberleşme ile koordinasyonu sağlayacak örnek bir bas konuş uygulaması geliştirilmiştir. geliştirilen sistemin ölçeklenebilir olduğunu göstermek için, Java tabanlı bas konuş yazılımı geliştirilmiştir. Geliştirilen bu yazılım, Ubuntu ve Windows 7 işletim sistemlerinde çalışarak daha fazla sayıda düğümden oluşan ağ yapısının emülasyonunda kullanılmıştır.

Yapılan bu çalışmadaki uygulama yazılımı daha ileri bir seviyeye taşınarak afetzedeleri de kapsayacak şekilde mesaj metni ile haberleşilmesi ve harita üzerinde kişilerin konumlarının gösterilmesi gibi özellikler eklenebilir. Gerçek kullanım senaryolarında cihaz sayısının yüksek sayıda olacağı beklenmektedir. Bu durumda MANET ağının verimli çalışması önem kazanmaktadır. Simülasyon araçları yardımıyla yüzlerce düğümden oluşan farklı topolojiler kurularak MANET ağının performansının artırılması için gerekli araştırmalar yapılması planlanmaktadır.

## KAYNAKLAR

- [1] SIDCARD, L., MARCOVIS, M. VE MANTHIOS, G., An Ad-hoc Network of Android Phones Using BATMAN. the Pervasive Computing Course. Project Report SPVC-E2010, 2010.
- [2] <http://www.idc.com/getdoc.jsp?containerId=prUS24442013>, Erişim Tarihi: 24.01.2014.
- [3] CHITAKORNKIJSIL, P., Disaster And Risk Management In A Global World. International Journal of Organizational Innovation 3(2): 97-113, 2010.
- [4] <http://www.fema.gov/mobile-emergency-response-support-telecommunications>, Erişim Tarihi: 26.01.2014.
- [5] ZHUANG, T., BASKETT, P. VE SHANG Y., Managing Ad Hoc Networks of Smartphones. International Journal of Information and Education Technology, 3(5): 540-546, 2013.
- [6] <http://ows.edb.utexas.edu/site/collaborative-bluetooth-edumanet-bednet>, Erişim Tarihi: 25.04.2014.
- [7] GOHS, R., SIDOROV, G., SIGUOUR, R. VE GLENSTRUP, AJ., Beddernet: application-level platform-agnostic MANETs. Distributed Applications and interoperable Systems. Springer Berlin Heidelberg, pp. 165-178, 2011
- [8] ANZALDI, D., ORWAR: a delay-tolerant protocol implemented on the Android platform, 2010.
- [9] RABIE, JK. VE LASSE, SR., Ad-hoc network on Android, 2010.
- [10] <http://www.servalproject.org/home>, Erişim Tarihi: 25.01.2014.
- [11] <http://www.technologyreview.com/view/517106/a-crowdfunding-campaign-to-set-smartphones-free-from-cellular-network>, Erişim Tarihi: 25.01.2014.
- [12] [http://www.olsr.org/docs/report\\_html/node9.html](http://www.olsr.org/docs/report_html/node9.html), Erişim Tarihi: 08.04.2014

- [13] SOYTÜRK, M., HARMANCI, E. VE ÇAYIRCI, E., Gezin Ad Hoc Ağlar ve Yol Atama, Bilişim Zirvesi '01, İstanbul, 4-7 Eylül 2000
- [14] [http://www.olsr.org/docs/report\\_html/node10.html](http://www.olsr.org/docs/report_html/node10.html), Erişim Tarihi: 08.04.2014.
- [15] JACQUET, P., MÜHLETHALER, P. VE LAOUITI, A., Optimized Link State Routing Protocol for Ad Hoc Networks. pp. 62-68, 2001.
- [16] ZHANG, Y. VE LUON, J., Wireless Mesh Networking, pp. 122-126, 2007.
- [17] <http://www.networksorcery.com/enp/rfc/rfc3626.txt>, Erişim Tarihi: 10.04.2014.
- [18] TONNESEN, A., Implementing and extending the Optimized Link State Routing Protocol, 2004.
- [19] [http://en.wikipedia.org/wiki/Android\\_%28operating\\_system%29](http://en.wikipedia.org/wiki/Android_%28operating_system%29), Erişim Tarihi: 10.04.2014.
- [20] <http://developer.android.com/guide/topics/manifest/uses-sdk-element.html>, Erişim Tarihi: 10.04.2014.
- [21] <http://developer.android.com/training/articles/security-tips.html>, Erişim Tarihi: 11.04.2014.
- [22] ABLESON, WF., SEN, R. ve KING, C., Android in Action. Second Edition, 2011.
- [23] <http://source.android.com/devices/tech/dalvik/>, Erişim Tarihi: 10.04.2014.
- [24] [http://www.phonearena.com/news/Meet-the-best-custom-ROMs-for-Android\\_id44353](http://www.phonearena.com/news/Meet-the-best-custom-ROMs-for-Android_id44353), Erişim Tarihi: 11.04.2014.
- [25] <https://code.google.com/p/android/issues/detail?id=82>, Erişim Tarihi: 11.04.2014.
- [26] [http://www.htcdev.com/bootloader/about\\_unlock\\_process](http://www.htcdev.com/bootloader/about_unlock_process), Erişim Tarihi: 05.01.2014.
- [27] <http://www.htcdev.com/bootloader>, Erişim Tarihi: 05.01.2014.
- [28] [http://www.htcdev.com/bootloader/preview\\_unlock\\_process](http://www.htcdev.com/bootloader/preview_unlock_process), Erişim Tarihi: 05.01.2014.
- [29] <https://www.clockworkmod.com/rommanager>, Erişim Tarihi: 05.01.2014.

- [30] <http://forum.xda-developers.com/attachment.php?attachmentid=603664&d=1306045709>, Eriřim Tarihi: 06.01.2014.
- [31] <http://www.wugfresh.com>, Eriřim Tarihi: 08.01.2014.
- [32] <http://www.youtube.com/watch?v=aPYNpqC8tKo&feature=youtu.be>, Eriřim Tarihi: 20.01.2014.
- [33] <http://download.cyanogenmod.org/?device=flo>, Eriřim Tarihi: 20.01.2014.
- [34] <https://developer.android.com/tools/sdk/ndk/index.html>, Eriřim Tarihi: 24.01.2014.

## ÖZGEÇMİŞ

Yusuf ÇERİ, 29.11.1983'de İzmir'de doğdu. İlk, orta ve lise eğitimini İzmir'de tamamladı. 2001 yılında İzmir Betontaş Lisesi'nden mezun oldu. 2007 Yılında İstanbul Teknik Üniversitesi Bilgisayar Mühendisliği bölümünden mezun oldu. 2011 yılında Sakarya Üniversitesi Bilgisayar ve Bilişim Mühendisliği Bölümü'nde yüksek lisansa başladı. 2008 yılından beri TÜBİTAK'da çalışmaktadır.