HARRAN ÜNİVERSİTESİ MÜHENDİSLİK DERGİSİ

HARRAN UNIVERSITY JOURNAL of ENGINEERING

e-ISSN: 2528-8733 (ONLINE)
URL: http://dergipark.gov.tr/humder

Kablosuz Ağlarda Gerçek-zamanlı Güç Tüketim Ölçümü

Real-time Power Consumption Measurement in Wireless Networks

Yazar(lar) (Author(s)):*Mehmet Fatih TÜYSÜZ*[1]

[1]ORCID ID: 0000-0002-8955-9710

Bu makaleye şu şekilde atıfta bulunabilirsiniz (To cite to this article): Tüysüz M.F.., "Kablosuz Ağlarda Gerçek-zamanlı Güç Tüketim Ölçümü", ***Harran Üniversitesi Mühendislik Dergisi***, 4(3): 148-155, (2019).

Erişim linki (To link to this article):http://dergipark.gov.tr/humder/archive

HRU Muh Der, 4(3): 148-155 (2019)

Harran Üniversitesi

Mühendislik Dergisi

http://dergipark.gov.tr/humder

Kablosuz Ağlarda Gerçek-zamanlı Güç Tüketim Ölçümü

Mehmet Fatih TÜYSÜZ

Harran Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Haliliye/Şanlıurfa

Makale Bilgisi

Başvuru: 01/12/2019
Düzeltme: 21/12/2019
Kabul: 24/12/2019

Anahtar Kelimeler

Kablosuz Ağlar
Güç Tüketimi
Enerji Verimliliği
Android

Keywords

Wireless Networks
Power Consumption
Energy Efficiency
Android

Öz

Teknolojinin ilerlemesiyle mobil cihazlar, gün geçtikçe daha da işlevsel hale gelmekte ve daha yoğun bir şekilde kullanılmaktadır. Mobil cihazlarda kablosuz ağların kullanımı ise cihazlardaki birçok işlem için bir zorunluluktur. Bu bağlamda enerji verimliliği, mobil cihaz kullanıcılarının cihazlarını daha uzun süre kullanabilmeleri açısından, önem arz etmektedir. Bu çalışma kapsamında mobil cihazlardaki kablosuz ve hücresel ağ arayüzlerinin enerji tüketimlerini, çalışan uygulama(lar)nın tipine bağlı olarak, gerçek zamanlı ölçebilen android-temelli bir uygulamanın tasarım ve kodlama kısımları işlenmektedir. Önerilen uygulama merkezi bir bağımlılığı olmadan mobil cihaz kullanıcılarının istediği zaman, istediği yerden cihazın güç tüketimini ölçebilmesine imkan tanımaktadır.

Real-time Power Consumption Measurement in Wireless Networks

Abstract

With the advancement of technology, mobile devices are becoming increasingly functional and more intensively used day by day. The use of wireless networks in mobile devices is a necessity for many operations on the device. In this context, energy efficiency is important in terms of enabling mobile users to use their devices for a longer time. This paper deals with the design and coding aspects of a real-time android-based application that measures the energy consumption of wireless and cellular network interfaces on mobile devices, depending on the type of application they are running. The proposed application allows mobile device users to measure the power consumption of the device from any location, without a central dependency.

1. GİRİŞ

Literatürde tablet ve telefonlardaki Wi-Fi [1], 3G [2] vb. ağ arayüzlerinin enerji tüketimlerini çalışan uygulamaların tipine bağlı olarak gerçek zamanlı ölçebilen çeşitli uygulamalar bulunmaktadır. Literatürde kabul edilen en iyi uygulamalar AppScope [3] ve PowerTutor [4] uygulamalarıdır.

AppScope mobil cihazların enerji tüketimini tahmin etmek için kullanılan Android tabanlı enerji ölçüm sistemidir. AppScope çekirdek düzeyinde donanım bileşenlerinin uygulama kullanımını izler ve olay güdümlü izleme yöntemi kullanır. AppScope, sadece Root izinlerine sahip bazı android işletim sistemiyle çalışan cihazlarda kullanılabilmektedir. Güç tüketim miktarlarının görüntülenebilmesi için cihazın kabloyla bilgisayara bağlanması gerekmektedir. Android işletim sistemine sahip mobil cihaz bilgisayara bağlandıktan sonra bilgisayarda açılan AppScope Viewer programından mobil cihazın güç tüketimini görüntülenebilmektedir. Uygulama Tablet ve telefonlardaki güç tüketimini anlık telefon üzerinden görüntülemeye imkan vermemektedir. Bunun yanında birçok cihazı da desteklememektedir.

PowerTutor ise CPU, GPU, Wi-Fi arabirimleri gibi donanım bileşenleri tarafından tüketilen gücü tahmini olarak hesaplayan bir android uygulamasıdır. Android marketten doğrudan indirilebilen bu uygulama 2010 yılında Michigan Üniversitesinde kodlanmıştır. G1, G2 ve Nexus One kod adlı telefonlarda test edilerek piyasaya sürülmüştür. Günümüzde 2.3 gibi eski Android işletim sistemi sürümlerini kullanan cihazlarda çalışmasına rağmen, güncel birçok cihazı desteklememektedir. Bunun yanında PowerTutor

*Mehmet Fatih TÜYSÜZ, e-mail: ftuysuz@harran.edu.tr

uygulaması kodlanırken Wi-Fi güç tüketimi hesabında Wi-Fi'nin Power_State_Low, ve Power_State_High olmak üzere sadece 2 moddan oluştuğu varsayılmıştır. Fakat kablosuz ağlarda mobil bir istasyon dört farklı durumda olabilir: (i) iletim, (ii) alım, (iii) boşta kalma ve (iv) uyku durumu. Belirtilen nedenden ötürü PowerTutor uygulaması da kablosuz ağın gerçek-zamanlı güç tüketimi ölçümü konusunda hatalı veriler üretmekte ve dolayısıyla yetersiz kalmaktadır.

Bildiri kapsamında geliştirdiğimiz çalışan uygulama(lar)nın tipine bağlı olarak tablet ve telefonların güç tüketimini gerçek zamanlı ölçebilen android uygulaması ise herhangi bir merkezi bağımlılığı olmadan tablet veya cep telefonu kullanan kullanıcının istediği zaman, istediği yerden cihazın güç tüketimini ölçebilmektedir. Bunun yanında önerilen uygulama herhangi yeni veya eski bir Android işletim sistemine bağımlı olarak çalışmamaktadır. Ek olarak anlık tüketilen güç miktarını hesaplayan algoritmalarımız literatürde incelenen çalışmalardan daha gerçekçi bir yaklaşımla geliştirilmiştir.

2. GÜÇ TÜKETİM ÖLÇÜMÜ

2.1. WiFi Güç Tüketim Ölçümü

Çalışma kapsamında geliştirilen uygulama Wi-Fi güç tüketimini hesaplarken mobil cihazın Power_State_Low, Power_State_Transmission, Power_State_Receive ve son olarak Power_State_Idle olmak üzere 4 ayrı Wi-Fi modunda olabileceğini kabul etmiştir [5]. Cep telefonu veya tabletin hangi Wi-Fi modunda olduğunu aşağıda kodu verilen updateState fonksiyonu algılamaktadır.

```
public void updateState(long transmitPackets, long receivePackets,
        long transmitBytes, long receiveBytes) {
    long curTime = SystemClock.elapsedRealtime();
    if (lastTime != -1 && curTime > lastTime) {
        double deltaTime = curTime - lastTime;
        lastUplinkRate = (transmitBytes - lastTransmitBytes) / 1024.0
                * 7.8125 / deltaTime;
        lastPackets = receivePackets + transmitPackets
                - lastReceivePackets - lastTransmitPackets;
        deltaUplinkBytes = transmitBytes - lastTransmitBytes;
        deltaDownlinkBytes = receiveBytes - lastReceiveBytes;

        if (transmitPackets != lastTransmitPackets) {
            lastAverageTransmitPacketSize = 0.9
                    * lastAverageTransmitPacketSize + 0.1
                    * (transmitBytes - lastTransmitBytes)
                    / (transmitPackets - lastTransmitPackets);
        }
        if (receivePackets != lastReceivePackets) {
            lastAverageReceivePacketSize = 0.9
                    * lastAverageReceivePacketSize + 0.1
                    * (receiveBytes - lastReceiveBytes)
                    / (receivePackets - lastReceivePackets);
        }
        if (receiveBytes != lastReceiveBytes
                || transmitBytes != lastTransmitBytes) {
            inactiveTime = 0;
        } else {
            inactiveTime += curTime - lastTime;
            powerState = POWER_STATE_idle;
        }
        if (transmitPackets == lastTransmitPackets || receivePackets == lastReceivePackets) {
            powerState = POWER_STATE_idle;
        }
        deltaUplinkBytes = transmitBytes - lastTransmitBytes;
        deltaDownlinkBytes = receiveBytes - lastReceiveBytes;


        if( deltaDownlinkBytes > deltaUplinkBytes  ) {

            if (powerState == POWER_STATE_Receive) {
                stateTime++;
            }else {
                powerState = POWER_STATE_Receive;
                stateTime = 0;
            }
        } else {

            if (powerState == POWER_STATE_Transmission) {
                stateTime++;
            }else {
                powerState = POWER_STATE_Transmission;
                stateTime = 0;
            }
        }
        boolean inactive = deltaUplinkBytes == 0 && deltaDownlinkBytes == 0;
        if(inactive) {
                stateTime = 0;
                powerState = POWER_STATE_idle;
            }
    }
    lastTime = curTime;
    lastTransmitPackets = transmitPackets;
    lastReceivePackets = receivePackets;
    lastTransmitBytes = transmitBytes;
    lastReceiveBytes = receiveBytes;
}
```

Kod 1. Wi-Fi için updateState fonksiyonu

Mobil cihazın hangi Wi-Fi modunda olduğu algılandıktan sonra telefonunun gerçek zamanlı güç tüketimi ise aşağıdaki getWifiPower fonksiyonu ile hesaplanmıştır. Bu fonksiyondaki sayısal değerler [5]'teki çalışmada kullanılan sayısal değerlerdir.

```
public double getWifiPower(WifiData data) {

    Log.w("Murat LOG",
            "wifi power state "+data.powerState+ "   ");

      if(!data.wifiOn) {
        return 0;
      }
      if(data.powerState == Wifi.POWER_STATE_LOW) {
        return coeffs.wifiLowPower();
      }
      if(data.powerState == Wifi.POWER_STATE_Transmission) {
        return 1300;
      }
      if(data.powerState == Wifi.POWER_STATE_Receive) {
        return 900;
      }
      if(data.powerState == Wifi.POWER_STATE_idle) {
        return 740;
      }
      throw new RuntimeException("Unexpected power state");
    }
```

Kod 2. getWifiPower fonksiyonu

2.2. 3G Güç Tüketim Ölçümü

Önerilen uygulama 3G arayüzünün enerji tüketimini de çalışan uygulama(lar)nın tipine bağlı olarak gerçek-zamanlı olarak ölçebilmektedir. Bu kapsamda 3G güç tüketimi sırasında mobil cihazın Power_State_DCH, Power_State_FACH ve Power_State_İdle olmak üzere 3 ayrı 3G modunda olabileceği kabul edilmiştir [6]. Mobil cihazın hangi 3G modunda olduğunu aşağıda kodu verilen updateState fonksiyonu algılamaktadır.

```
public void updateState(long transmitPackets, long receivePackets,
                        long transmitBytes, long receiveBytes,
                        int dchFachDelay, int fachIdleDelay,
                        int uplinkQueueSize, int downlinkQueueSize) {
  long curTime = SystemClock.elapsedRealtime();
  if(lastTime != -1 && curTime > lastTime) {
    double deltaTime = curTime - lastTime;
    deltaPackets = transmitPackets + receivePackets -
                   lastTransmitPackets - lastReceivePackets;
    deltaUplinkBytes = transmitBytes - lastTransmitBytes;
    deltaDownlinkBytes = receiveBytes - lastReceiveBytes;
    boolean inactive = deltaUplinkBytes == 0 && deltaDownlinkBytes == 0;
    inactiveTime = inactive ? inactiveTime + curTime - lastTime : 0;

              switch(powerState) {
                case POWER_STATE_IDLE:
                  if(!inactive) {
                    powerState = POWER_STATE_FACH;
                  }
                  break;
                case POWER_STATE_FACH:
                  if(inactive) {
                    stateTime++;
                    if(stateTime >= fachIdleDelay * timeMult) {
                      stateTime = 0;
                      powerState = POWER_STATE_IDLE;
                    }
                  } else {
                    stateTime = 0;
                    if(deltaUplinkBytes > 0 ||
                       deltaDownlinkBytes > 0) {
                      powerState = POWER_STATE_DCH;
                    }
                  }
                  break;
                default:
                  if(inactive) {
                    stateTime++;
                    if(stateTime >= dchFachDelay * timeMult) {
                      stateTime = 0;
                      powerState = POWER_STATE_FACH;
                    }
                  } else {
                    stateTime = 0;
                  }
              }
              lastTime = curTime;
              lastTransmitPackets = transmitPackets;
              lastReceivePackets = receivePackets;
              lastTransmitBytes = transmitBytes;
              lastReceiveBytes = receiveBytes;
            }
```

Kod 3. 3G için updateState Fonksiyonu

Cep telefonu veya tabletin hangi 3G modunda olduğu algılandıktan sonra telefonunun gerçek zamanlı güç tüketimi ise aşağıdaki getThreeGPower fonksiyonu ile hesaplanmıştır.

```
public double getThreeGPower(ThreegData data) {
  if(!data.threegOn) {
    return 0;
  } else {
    switch(data.powerState) {
      case Threeg.POWER_STATE_IDLE:
        return coeffs.threegIdlePower(data.oper);
      case Threeg.POWER_STATE_FACH:
        return coeffs.threegFachPower(data.oper);
      case Threeg.POWER_STATE_DCH:
        return coeffs.threegDchPower(data.oper);
    }
  }
  return 0;
}
```

Kod 4. getThreeGPower fonksiyonu

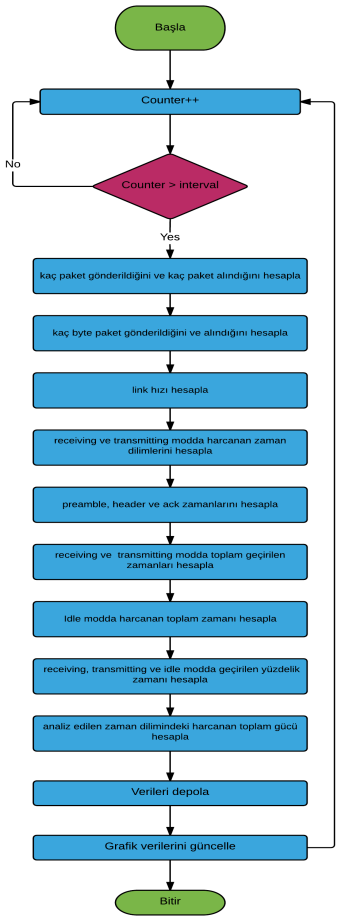
3. ÖNERİLEN YAKLAŞIM

Yapılan çalışmalar kapsamında geliştirilen uygulamanın Algoritma şeması genel hatlarıyla Şekil 1'de verilmiştir. Şekil 1'de görüldüğü gibi geliştirilen uygulama analiz edilen zaman dilimi içerisinde kaç paket gönderildiği ve kaç paket alındığının hesaplanması ile başlamaktadır. Bu işlemler için aşağıda belirtilen Formül 1 ve Formül 2 kullanılmıştır.

$$currentReceivedpackets = lastReceivePackets - previousReceivePackets \quad (1)$$

$$currentTransmittedpacket = lastTransmitPackets - previousTransmitPacket \quad (2)$$

burada previousReceivePackets ve previousTransmitPackets verileri bir önceki zaman diliminde lastReceivePackets ve lastTransmitPacket verileri atanarak elde edilmiştir.

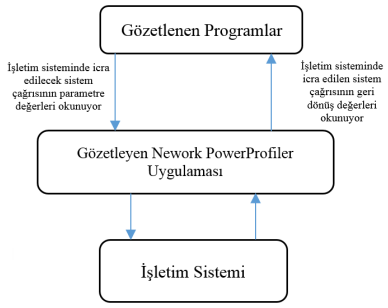


Şekil 1. Network PowerProfiler uygulaması algoritma şeması

Formül 1 ve Formül 2'de kullanılan lastReceivePackets ve lastTransmitPackets verileri ise mobil cihazın /proc dizininden okunarak elde edilmiştir. Unix sistemlerindeki proc dosya sistemi /proc dizinine mount edilmiştir. Bu dosya sistemi gerçek bir dosya sistemi değildir. Bu dosya sistemi içerisindeki çeşitli dosyalar sistemin durumu hakkında bilgiler içerirler. Yani o dosyaları açıp onların içerisinden okuma yapmakla çekirdeğe ilişkin bilgiler edinilebilir.

/proc dosya sistemi, işletim sistemindeki her sürecin adres alanına erişimi sağlar. Sistemde oluşturulan bütün süreçler /proc dizini altında temsili dosyalar şeklinde tutulurlar. /proc dizininde bulunan her bir dosya, ilgili sürecin ID'si ile isimlendirilmiştir ve her bir dosyanın sahibi sürecin gerçek kullanıcı (real user) ID'si ile belirlenir. Birçok Unix programı (gdb, ps, top ve truss gibi) süreçleri kontrol etmek ve onlarla ilgili olarak bilgi toplamak için /proc dosya sistemini kullanır.

Süreç gözetleme mekanizmaları, bir sürecin diğer bir sürecin sistem kaynaklarını kullanımını kolay bir şekilde takip edebilmesini sağlarlar. Bu nedenle, bir süreç gözetleme işlemi en az iki süreç içerir; gözetleyen süreç ve gözetlenen süreç. Şekil 2'de gözetlenen süreçleri ve gözetleyen Network PowerProfiler süreci ile işletim sistemi ilişkisi gösterilmiştir.



Şekil 2. Süreç gözetleme mimarisi

Bir süreç gözetlenmeye başlanmadan önce hangi sistem aktiviteleri (sistem çağrıları, sinyaller, bazı donanım hataları) ile ilgileniliyor ise bu aktiviteler işletim sistemine gözetlemeyi yapan süreç tarafından bildirilir. Gözetlenen süreç çalışmaya başlaması ile gözetleyen süreç gözetlenen sürecin durdurulmasını bekler. Gözetlenen süreç bu aktivitelerden birini gerçekleştireceği zaman işletim sistemi tarafından durdurulur ve gözetleyen sürece haber verilir. Bir süreç için durdurulma noktası, icra kontrolünün işletim sistemi çekirdeğine verildiği ve geri alındığı noktadır. Diğer bir ifade ile gözetlenen sürecin sistem çağrısı yaptığı ve bu çağrıdan geri dönüldüğü noktalarda onun adres alanına erişilir. Adres alanındaki okuma ve yazma işlemleri sadece gözetlenen sürecin icrası durdurulduktan sonra gerçekleştirilebilir. Gerekli okuma işlemleri gerçekleştikten sonra icrasının devam ettirilmesi gerekmektedir. Gözetlenen sürecin durdurulması ya da icrasının başlatılması işletim sisteminin kontrolündedir.

Bildiri kapsamında yapılan çalışmalarda /proc dizininden analiz edilen zaman dilimi içerisinde mobil cihazda çalışan bütün programların toplam aldığı/verdiği paket sayıları okunmuştur. /proc dizininden okuma işlemi sonrasında Formül 1 ve Formül 2 uygulanmış, yani her iterasyon bir önceki iterasyonla karşılaştırılıp, o zaman diliminde kaç tane paket alındığı/verildiği hesaplanmıştır.

Uygulamanın ikinci aşamasında analiz edilen zaman dilimi içerisinde kaç byte paket gönderildiği ve alındığı hesaplanmıştır. Bu işlemler için Formül 3 ve Formül 4 kullanılmıştır.

$$currentByteReceived = lastByteReceived - previousByteReceived \quad (3)$$

$$currentByteTransmitted = lastByteTransmited - previousByteTransmited \quad (4)$$

Formül 3 ve formül 4'te kullanılan previousByteReceived ve previousByteTransmited verileri bir önceki zaman diliminde sırasıyla lastByteReceived ve lastByteTransmited verileri atanarak elde edilmiştir. Formül 3 ve Formül 4'te kullanılan lastByteReceived ve lastByteTransmited verileri ise mobil cihazın /proc dizininden okunarak elde edilmiştir. Formül 3 ve Formül 4 kullanılarak yapılan işlemler sonucunda her iterasyon bir önceki iterasyonla karşılaştırılıp, o zaman diliminde kaç byte paket alındığı/verildiği hesaplanmıştır.

Uygulamanın üçüncü aşamasında analiz edilen zaman dilimi içerisindeki link hızı hesaplanmıştır. Dördüncü aşamasında ise analiz edilen zaman dilimi içerisindeki alım, boşta bekleme ve iletim modunda harcanan zaman dilimleri hesaplanmıştır. Bu işlemler için Formül 5 ve 6 kullanılmıştır.

$$T_{receiving} = \frac{CurrentByte_{received}}{LinkSpeed_{Byte}} \tag{5}$$

$$T_{transmitting} = \frac{CurrentByte_{transmitting}}{LinkSpeed_{Byte}} \tag{6}$$

Uygulamanın beşinci aşamasında analiz edilen zaman dilimi içerisindeki preamble, header ve ack zamanları hesaplanmıştır. Bu işlemler için Formül 7, 8, 9, 10, 11 ve 12 kullanılmıştır.

$$T_{preambleReceiving} = \text{currentReceivedpackets} * T_{preamble} \tag{7}$$

$$T_{preambleTransmitting} = \text{currentTransmittedpackets} * T_{preamble} \tag{8}$$

$$T_{headerReceiving} = \text{currentReceivedpackets} * \text{T}_{\text{header}} \tag{9}$$

$$T_{headerTranmitting} = \text{currentTransmittedpackets} * \text{T}_{\text{header}} \tag{10}$$

$$T_{ackReceiving} = \text{currentTransmittedpackets} * \text{T}_{\text{ack}} \tag{11}$$

$$T_{ackTransmitting} = \text{currentReceivedpackets} * \text{T}_{\text{ack}} \tag{12}$$

Uygulamanın altıncı aşamasında alım ve iletim modunda toplam geçirilen zamanlar hesaplanmıştır. Bu işlemler için Formül 13 ve Formül 14 kullanılmıştır.

$$Total_{receivingTime} = T_{receiving} + T_{preambleReceiving} + T_{headerReceiving} + T_{ackReceiving} \tag{13}$$

$$Total_{transmittingTime} = T_{transmitting} + T_{preambleTransmitting} + T_{headerTranmitting} + T_{ackTransmitting} \tag{14}$$

Bu işlemler sonucunda alım ve iletim modlarında harcanan toplam zamanlar elde edilir. Boşta kalma modunda harcanan toplam güç ise Formül 15 de gösterildiği gibi hesaplanmaktadır.

$$Total_{idleTime} = zamanDilimi - Total_{receivingTime} - Total_{transmittingTime} \tag{15}$$

Uygulamanın yedinci aşamasında analiz edilen zaman dilimi içerisinde alım, iletim ve boşta kalma modlarında ne kadar kalındığının yüzdelik oranı hesaplanmaktadır. Bu işlemler için Formül 16, 17 ve 18 kullanılmıştır.

$$idleRate = \frac{Total_{idleTime}}{zamanDilimi} \tag{16}$$

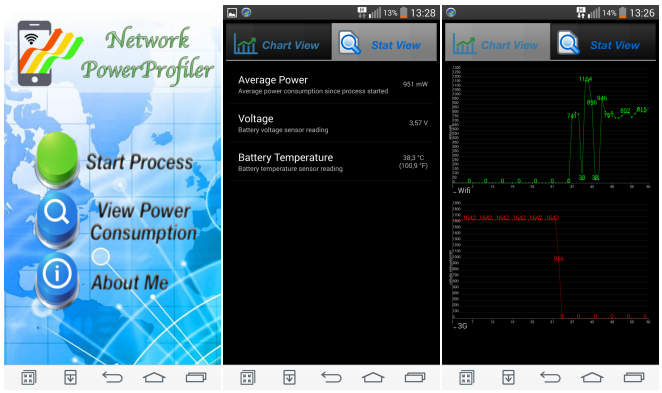
$$receivingRate = \frac{Total_{receivingTime}}{zamanDilimi} \quad (17)$$

$$transmittingRate = \frac{Total_{transmittingTime}}{zamanDilimi} \quad (18)$$

Bu işlemler sonucunda analiz edilen zaman dilimi içerisinde hangi güç tüketim modunda ne kadar zaman geçirildiği oransal olarak elde edilmiştir. Uygulamanın sekizinci aşamasında ise analiz edilen zaman dilimi içerisinde ne kadar güç harcandığı hesaplanmıştır. Bu işlem için Formül 19 kullanılmıştır.

$$Total_{power} = P_{idle} * idle_{rate} + P_{receiving} * receiving_{rate} + P_{transmitting} * transmitting_{Rate} \quad (19)$$

Uygulamamızla ilgili görseller aşağıda paylaşılmaktadır. Görüldüğü üzere bu çalışma kapsamında önerilen protokole ilişkin kod bilgileri sunulmuştur. Çalışma ile ilgili detaylı analizlere, benzetim ortamı kurulumu ve senaryolarına, ve sonuçların senaryo bazlı değerlendirilmesine tarafımızca sunulan ilgili diğer [7-9] çalışmalardan erişilebilir.



Şekil 3. Network PowerProfiler Uygulaması ekran çıktıları

3. SONUÇ

Bildiri kapsamında geliştirdiğimiz tablet ve telefonların kablosuz ağ arayüzünün güç tüketim miktarını gerçek zamanlı ölçebilen bir android uygulaması ile herhangi bir merkezi bağımlılığı olmadan kullanıcının istediği zaman istediği yerden mobil cihazının kablosuz ağ arayüzünün güç tüketimini ölçebilmesine imkan tanınmıştır. Bunun yanında, tasarlanan uygulama herhangi yeni veya eski bir Android işletim sistemine bağımlı olarak çalışmamaktadır. Ek olarak, anlık tüketilen güç miktarını hesaplayan algoritmalarımız literatürde incelenen çalışmalardan daha gerçekçi bir yaklaşımla geliştirilmiştir. Bildiri kapsamında önerilen uygulamanın tasarım ve kodlama kısımları işlenmiştir. Sayfa sınırı nedeniyle, gerçekleştirilen detaylı analizlere, hazırlanan ortam senaryolarına ve test sonuçlarına bu çalışmada yer verilemediğinden dolayı, bahsi geçen konuların başka bir yayın ile aktarılması düşünülmektedir.

KAYNAKLAR

[1] IEEE Std 802.11, “Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications”, 1999 Edition.

[2] Smith, Clint. *3G wireless networks*. McGraw-Hill, Inc., 2006.

[3] Yang, Z. "Powertutor-a power monitor for android-based mobile platforms." *EECS, University of Michigan, retrieved September* 2 (2012): 19.

[4] Yoon, Chanmin, et al. "AppScope: Application Energy Metering Framework for Android Smartphone Using Kernel Activity Monitoring." *USENIX Annual Technical Conference*. Vol. 12. 2012.

[5] Tuysuz, Mehmet Fatih. "An energy-efficient QoS-based network selection scheme over heterogeneous WLAN–3G networks." *Computer Networks* 75 (2014): 113-133.

[6] Harjula, Erkki, Otso Kassinen, and Mika Ylianttila. "Energy consumption model for mobile devices in 3G and WLAN networks." *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*. IEEE, 2012.

[7] Tuysuz, M. F., & Uçan, M. (2017). Energy-aware network/interface selection and handover application for android-based mobile devices. Computer Networks, 113, 17-28.

[8] Tuysuz M.F., (2017) "Energy-efficient Vertical Handover Parameters, Classification and Solutions over Wireless Heterogeneous Networks: A Survey", 97(1), 1155-1184.

[9] Tuysuz, M. F., Ucan, M., & Trestian, R. (2019). A real-time power monitoring and energy-efficient network selection tool for android smartphones. Journal of Network and Computer Applications, 127, 107-121.