Android (/tags/#Android) frameworks (/tags/#frameworks) AlarmManager (/tags/#AlarmManager)

AlarmManagerService之MTK AmPlus源码

反编译了MTK的Alarm对齐方案中的AmPlus部分核心代码

Posted by Cheson on March 6, 2017

1. 介绍

在 AlarmManagerService之设置alarm流程 (https://chendongqi.github.io/blog/2017/02/14/SetAlarmFlow/)中介绍了MTK的Alarm对齐方案，其中涉及的核心代码MTK以jar包形式释放，本篇中附上了反编译并翻译之后的代码。jar包在MTK AOSP源码中的位置为：vendor//libs//com_mediatek_amplus。

2. AlarmManagerPlus.java

```
package com.android.server;

import android.app.PendingIntent;
import android.content.Context;
import com.android.server.PowerSavingUtils;

public class AlarmManagerPlus {

    private PowerSavingUtils mPowerSavingUtils = null;


    public AlarmManagerPlus(Context context) {
        mPowerSavingUtils = new PowerSavingUtils(context);
    }

    public long getMaxTriggerTime(int type, long triggerElapsed,
            long windowLength, long interval, PendingIntent operation) {
        return mPowerSavingUtils.getMaxTriggerTime(type, triggerElapsed,
                windowLength, interval, operation);
    }

}
```

3. PowerSavingUtils.java

```
package com.android.server;


import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;

import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.ApplicationInfo;
import android.content.pm.PackageManager;
import android.os.Binder;
import android.os.SystemClock;
import android.util.Log;

public class PowerSavingUtils {

    private static final long SCREENOFF_TIME_INTERVAL_THRESHOLD = 300000L;

    private final Context mContext;
    private boolean mScreenOff = false;
    private long mScreenOffTime = 0L;
    private boolean mIsUsbConnected = false;


    private PowerSavingReceiver mPowerSavingReceiver;

    private static final String TAG = "AlarmManager";
    private static final String FILEPATH = "/system/etc/alarmplus.config";

    final ArrayList<String> mWhitelist = new ArrayList<String>();
    private static final long MIN_FUZZABLE_INTERVAL = 10000L;

    public PowerSavingUtils(Context cxt) {
        this.mContext = cxt;
        init();
    }

    void init() {
        readList();
        mPowerSavingReceiver = new PowerSavingReceiver();

    }

    private void readList() {
        File WhitelistFile = new File(FILEPATH);
        if (!(WhitelistFile.exists())) {
            return;
    }

    BufferedReader br = null;
    try {
         br = new BufferedReader(
                new FileReader(WhitelistFile));
        String line = br.readLine();
        while (line != null) {
            mWhitelist.add(line);
            line = br.readLine();
        }

    } catch (IOException e) {
        e.printStackTrace();
    }finally{
        if(br!=null){
            try {
                br.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
```

```
    }

    private boolean isAlarmNeedAlign(int type, PendingIntent operation,
            boolean isExactAlarm) {

        if (!isPowerSavingStart()) {
            return false;
        }

        boolean isAlarmNeedAlign = false;
        outer: if (type == 0 || type == 2) {
            String packageName = operation.getCreatorPackage();
            if (packageName == null) {
                Log.v(TAG, "isAlarmNeedAlign : packageName is null");
            } else {
                for (int i = 0; i < mWhitelist.size(); ++i) {
                    if ((mWhitelist.get(i)).equals(packageName)) {
                        Log.v(TAG, "isAlarmNeedAlign : packageName = "
                                + packageName + "is in whitelist");
                        return false;
                    }
                }
                if (isExactAlarm) {
                    PackageManager pm = mContext.getPackageManager();
                    try {
                        ApplicationInfo info = pm.getApplicationInfo(packageName, 0);

                        if (((info.flags & 0x1) != 0) && (packageName.startsWith("com.android"
                            Log.v(TAG, "isAlarmNeedAlign : "+ packageName + " skip!");
                            break outer;
                        }
                    } catch (PackageManager.NameNotFoundException e) {
                        Log.v(TAG, "isAlarmNeedAlign : packageName not fount");
                        break outer;
                    }
                }
                //Log.v(TAG, "isAlarmNeedAlign = true");
                isAlarmNeedAlign = true;
            }
        }
        return isAlarmNeedAlign;
    }

    public  boolean isPowerSavingStart() {

        if (mIsUsbConnected) {
            return false;
        }

        if (mScreenOff) {
            long currentTime = System.currentTimeMillis();
            long screenOffThreshold = 30000L;
            if (currentTime - mScreenOffTime < screenOffThreshold) {
                Log.v(TAG, "mScreenOff time is not enough");
                return false;
            }
        } else {
            return false;
        }
        Log.v(TAG, "isPowerSavingStart = true");
        return true;
    }

    private long getMMaxTriggerTime(int type, PendingIntent operation,long triggerAtTime) {

        if ((isAlarmNeedAlign(type, operation, true))) {
            return (triggerAtTime + SCREENOFF_TIME_INTERVAL_THRESHOLD);
        }
        return triggerAtTime;
    }

    private long adjustMaxTriggerTime(long now, long triggerAtTime,
            long interval, PendingIntent operation, int type,
            boolean isExactAlarm) {

        long futurity = (interval == 0L) ? triggerAtTime - now : interval;
```

```
        if (futurity < MIN_FUZZABLE_INTERVAL) {
            futurity = 0L;
        }

        long maxTriggerAtTime = triggerAtTime + (long) (.75 * futurity);

        if ((isAlarmNeedAlign(type, operation, isExactAlarm)) && (maxTriggerAtTime - triggerAt
            return (triggerAtTime + SCREENOFF_TIME_INTERVAL_THRESHOLD);
        }
        return maxTriggerAtTime;
    }

    public long getMaxTriggerTime(int type, long triggerElapsed,
            long windowLength, long interval, PendingIntent operation) {

        long maxElapsed;
        long nowElapsed = SystemClock.elapsedRealtime();

        if (windowLength == 0L) {
            maxElapsed = getMMaxTriggerTime(type, operation, triggerElapsed);
        } else if (windowLength == -1L) {
            maxElapsed = adjustMaxTriggerTime(nowElapsed, triggerElapsed,interval, operation,
        } else {
            maxElapsed = triggerElapsed + windowLength;
        }
        return maxElapsed;
    }



    class PowerSavingReceiver extends BroadcastReceiver {
        public PowerSavingReceiver() {
            IntentFilter filter = new IntentFilter();
            filter.addAction("android.intent.action.SCREEN_OFF");
            filter.addAction("android.intent.action.SCREEN_ON");
            filter.addAction("android.hardware.usb.action.USB_STATE");
            mContext.registerReceiver(this, filter);
        }

        public void onReceive(Context context, Intent intent) {

            if (Intent.ACTION_SCREEN_OFF.equals(intent.getAction())) {
                mScreenOff = true;
                mScreenOffTime = System.currentTimeMillis();

            } else if (Intent.ACTION_SCREEN_ON.equals(intent.getAction())) {

                mScreenOff = false;
                mScreenOffTime = 0L;

            } else if ("android.hardware.usb.action.USB_STATE".equals(intent.getAction())) {
                mIsUsbConnected = intent.getBooleanExtra("connected", false);

                }
            }
        }
    }
```

PREVIOUS
ALARMMANAGERSERVICE之ALARMGROUP机制剖析 **(/2017/03/06/ALARMGROUP/)**

NEXT
一场ANDROID PERFORMANCE的追根溯源之旅 **(/2017/03/08/ANDROID_PER_PATTERNS_OVERVIEW/)**

FEATURED TAGS (/tags/)

前端 (/tags/#前端) Android (/tags/#Android) frameworks (/tags/#frameworks) AlarmManager (/tags/#AlarmManager) Performance (/tags/#Performance) systrace (/tags/#systrace) PowerManager (/tags/#PowerManager) Wakelock (/tags/#Wakelock) Guitar (/tags/#Guitar) 民谣 (/tags/#民谣) 赵雷 (/tags/#赵雷) Doze (/tags/#Doze) Android Performance Patterns (/tags/#Android Performance Patterns)

FRIENDS

待遇见志同道合的你 (https://github.com)　小明 (http://www.betterming.cn)

(https://twitter.com/chendongqi)

(https://www.zhihu.com/people/chendongqi)

(http://weibo.com/chendongqi) (https://www.facebook.com/chendongqi)

(https://github.com/chendongqi)

(https://www.linkedin.com/in/firstname-lastname-idxxxx)