腾讯课堂 学习成就梦想 + 动脑学院 DONGNAO COLLEGE

《Android架构师课程》

做一家受人尊敬的企业，做一位受人尊敬的老师

讲师介绍

动脑学院 – Alan老师

系统架构师、技术总监

- 曾任职于上海腾讯互动娱乐部
- 最高职位技术总监。具备多年大型项目开发经验。技能领域：Android（高级UI,架构师）、java web 开发、nodejs、前端技术等。
- **现为动脑学院安卓讲师。**
- 老师QQ： 3287987589

Android 消息机制

1、源码分析Android 消息机制

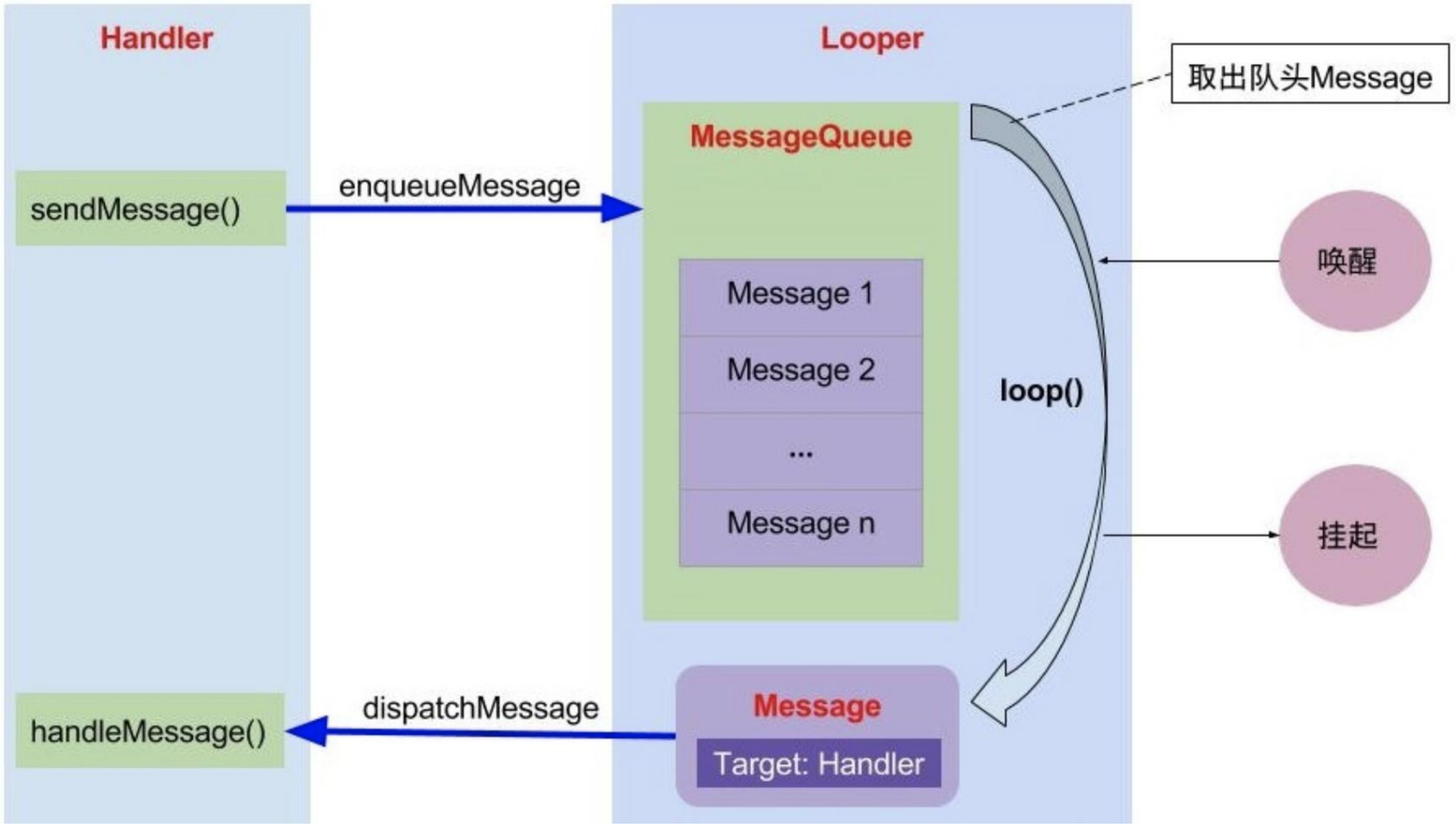
2、手写Handler消息机制

3、Handler中常见问题分析

Android 消息机制

Android 的消息机制主要是指Handler得运行机制

Android 消息机制



Android 消息机制

以上模型的解释：

1.以Handler的sendMessage方法为例，当发送一个消息后，会将此消息加入消息队列MessageQueue中。

2.Looper负责去遍历消息队列并且将队列中的消息分发给对应的Handler进行处理。

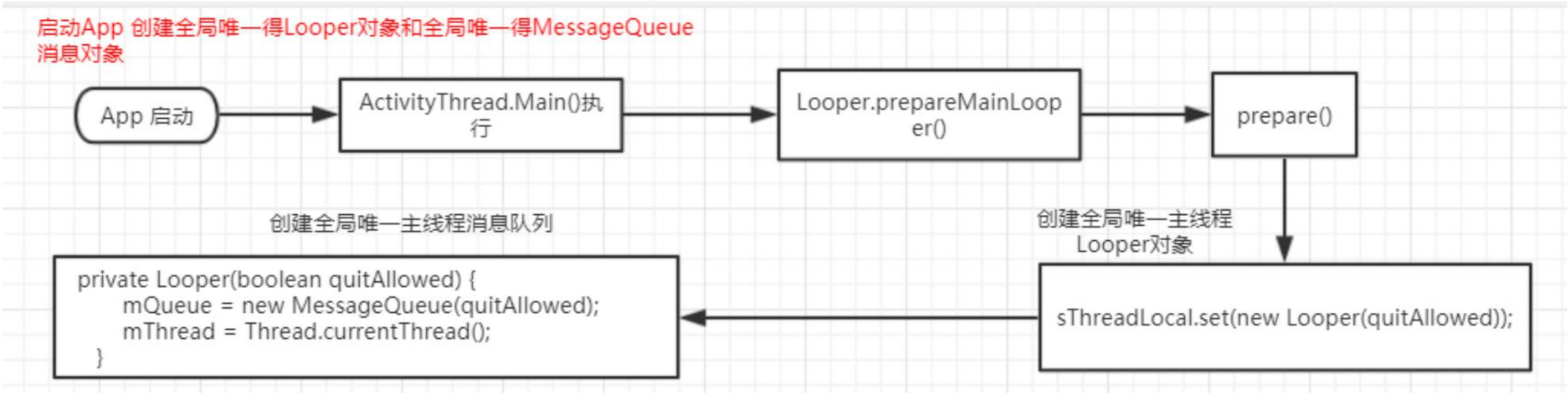
3.在Handler的handleMessage方法中处理该消息，这就完成了一个消息的发送和处理过程。

这里从图中可以看到参与消息处理有四个对象，它们分别是 Handler, Message, MessageQueue，Looper。

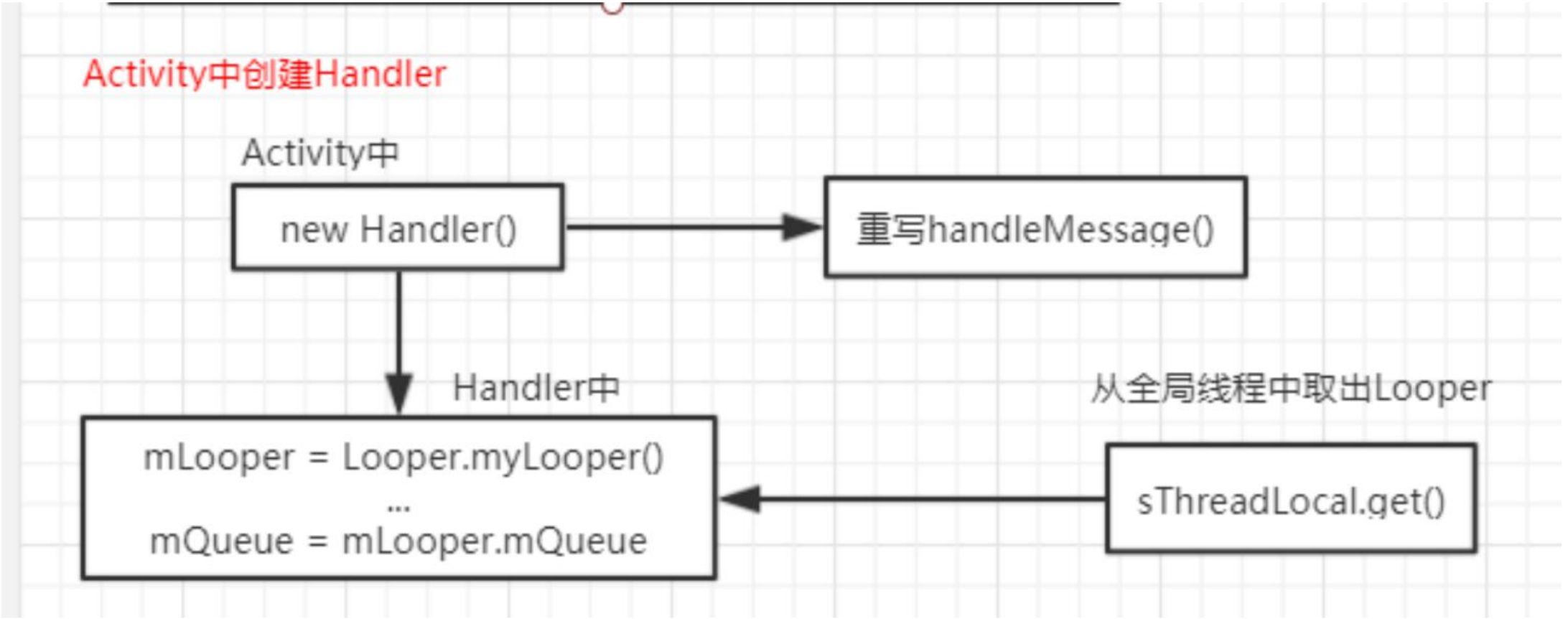
ThreadLocal 的工作原理

定义：ThreadLocal 是一个线程内部的数据存储类，通过它可以在指定的线程中存储数据，数据存储以后，只有再指定线程中可以获取到存储的数据，对于其他线程来说则无法获取到数据。

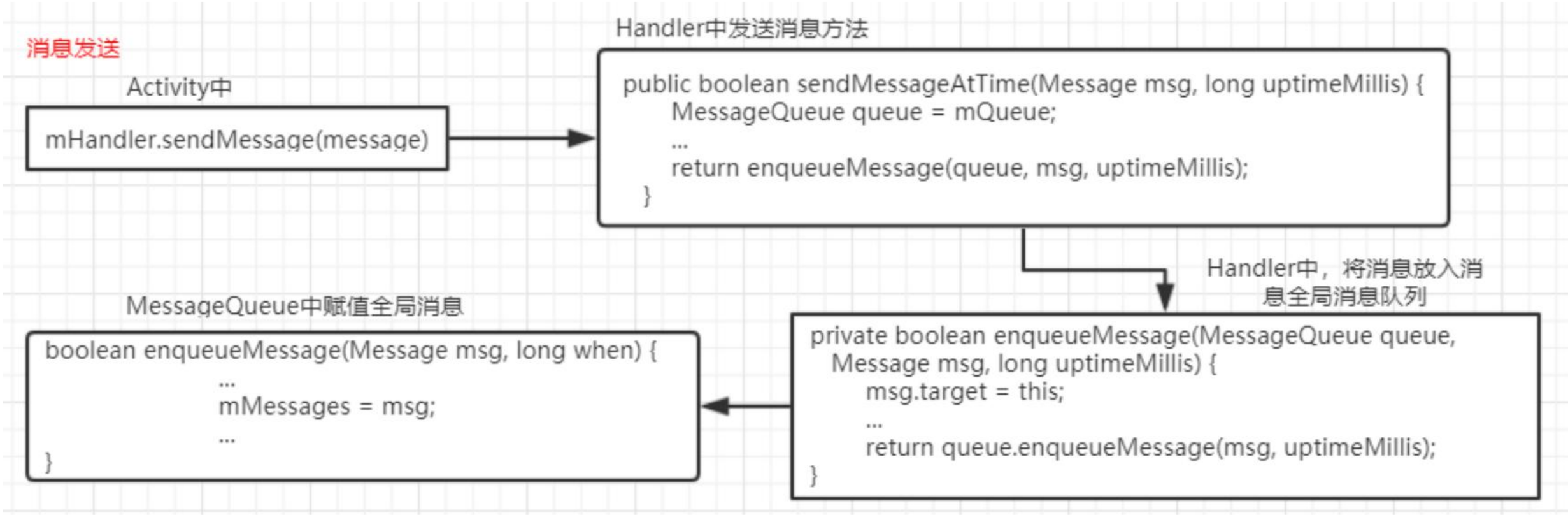
启动App 创建全局唯一得Looper对象和全局唯一得MessageQueue消息对象



Activity中创建Handler



消息发送


消息发送
Activity中
mHandler.sendMessage(message)
Handler中发送消息方法
public boolean sendMessageAtTime(Message msg, long uptimeMillis) {
MessageQueue queue = mQueue;
...
return enqueueMessage(queue, msg, uptimeMillis);
}
Handler中，将消息放入消息全局消息队列
private boolean enqueueMessage(MessageQueue queue,
Message msg, long uptimeMillis) {
msg.target = this;
...
return queue.enqueueMessage(msg, uptimeMillis);
}
MessageQueue中赋值全局消息
boolean enqueueMessage(Message msg, long when) {
...
mMessages = msg;
...
}

消息发送

第一种

```
public void test(View view) {
    new Thread((Runnable) () -> {
            Message message = mHandler.obtainMessage( what: 0);
            message.what = 1;
            mHandler.sendMessage(message);
    }).start();
}
```

点击事件子线程

第二种

```
public void test(View view) {
    new Thread((Runnable) () -> {
            mHandler1.post(new Runnable() {
                @Override
                public void run() {
                    textView.setText("大家，晚上好！");
                }
            });
    }).start();
}
```

发送消息

处理消息

消息发送

处理消息

ActivityThread.main方法中

```
Looper.loop();
```

Looper.loop()开启消息循环

```
public static void loop() {
        final Looper me = myLooper();  //获取全局唯一Looper对象
        final MessageQueue queue = me.mQueue;//获取消息队列
        ...
        for (;;) {    //开启是循环
            Message msg = queue.next(); // 从消息列表中不断获取消息
            ...
            msg.target.dispatchMessage(msg); //调用Handler中得方法
            ...
        }
        ...
}
```

Handler中消费消息（调用接口或重写方法）

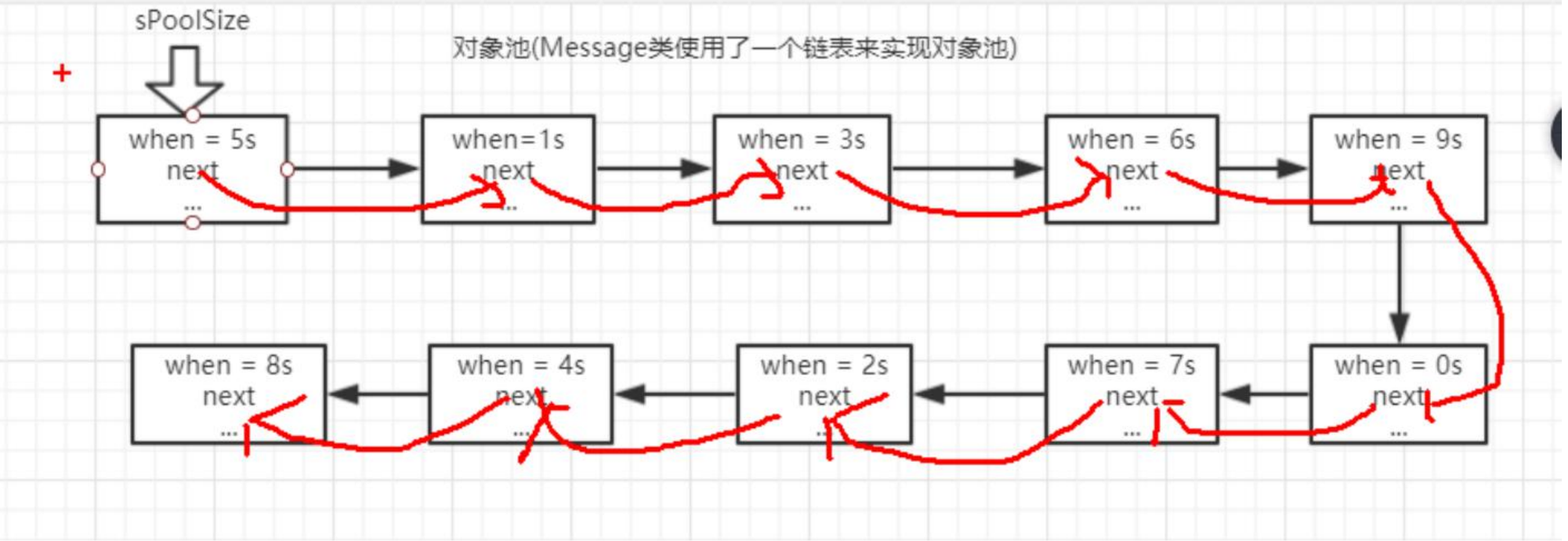
```
public void dispatchMessage(Message msg) {
    if (msg.callback != null) {
        handleCallback(msg);
    } else {
        if (mCallback != null) {
            if (mCallback.handleMessage(msg)) {
                return;
            }
        }
        handleMessage(msg);
    }
}
```

分析消息阻塞（阻塞时间）、唤醒、延时入队

Looper 的阻塞主要是靠 MessageQueue 来实现的，在next()@MessageQuese 进行阻塞，在 enqueueMessage()@MessageQueue 进行唤醒。主要依赖 native 层的 Looper 依靠 epoll 机制进行的。

nativePollOnce(ptr, nextPollTimeoutMillis);
这里调用naive方法操作管道，由nextPollTimeoutMillis决定是否需要阻塞
nextPollTimeoutMillis为0的时候表示不阻塞，为-1的时候表示一直阻塞直到被唤醒

对象池简单理解图


sPoolSize
对象池(Message类使用了一个链表来实现对象池)
when = 5s
next
...
when=1s
next
...
when = 3s
next
...
when = 6s
next
...
when = 9s
next
...
when = 0s
next
...
when = 7s
next
...
when = 2s
next
...
when = 4s
next
...
when = 8s
next
...

课后作业

1、简单描述Android 消息机制。

2、简单描述消息机制得阻塞、唤醒、延时加入消息队列。

3、手写Handler 实现。

下节课

Android Binder机制

谢谢观看