**移动感知中间件**

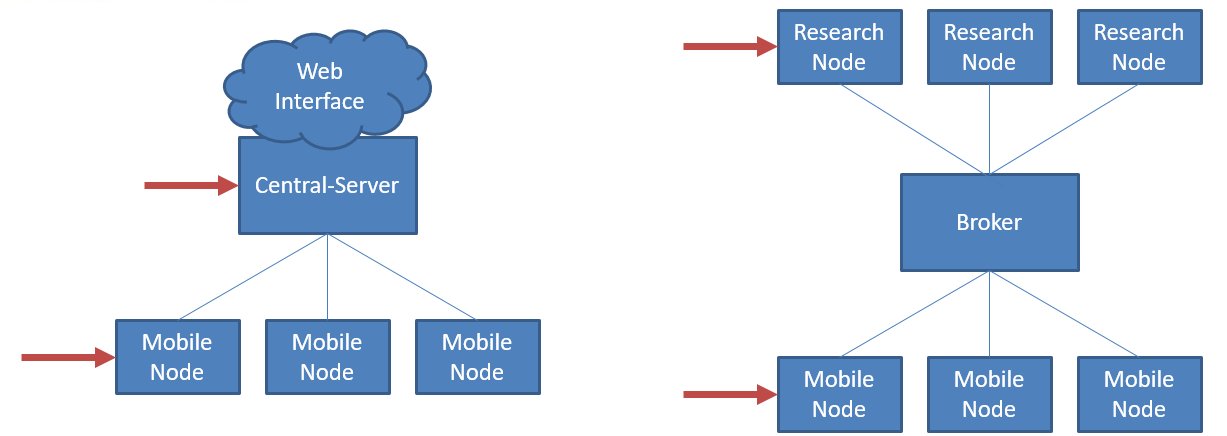
为什么会出现移动感知中间件，是因为手机有着丰富的传感器资源和强大的处理性能，而且使用规模也相当广泛，差不多人手一部。这为一些科研实验或者是特殊应用提供了强有力的保障。

但问题是，当下，缺少一个中间件体系将手机的感知资源高效的整合起来，使得应用开发者可以**透明得**去操纵大规模的手机感知，进而开发复杂应用。

两篇典型论文：

**Paper 1 ：Pogo, a Middleware for Mobile Phone Sensing**Niels Brouwers@Middleware 2012

第一篇paper主要是设计了一款中间件Pogo，目的是为了能够快捷方便的利用大规模的移动设备部署科研实验，缩短实验应用的开发周期。另外，还提出了一种与其他应用程序自动同步数据传输的传输方案，大大降低了实验数据的传输能耗。



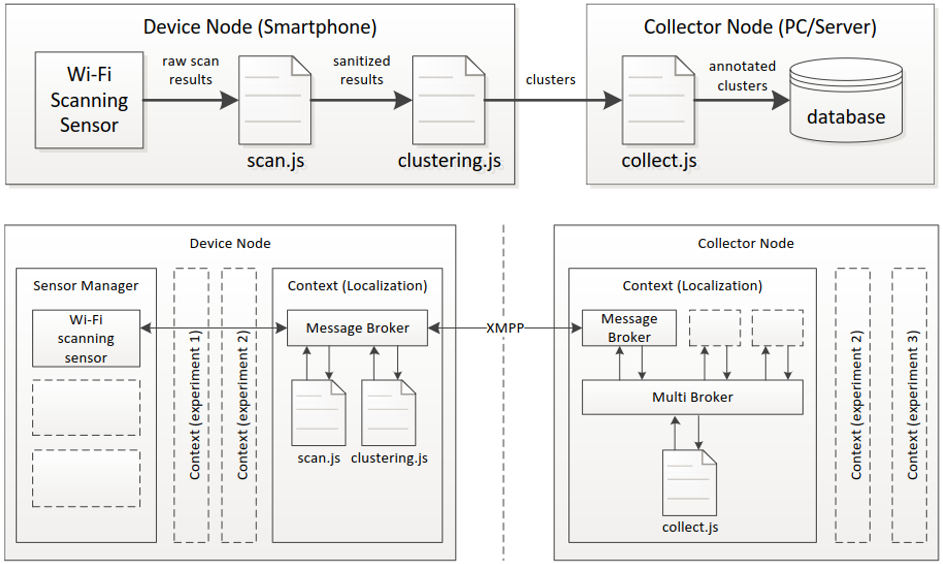
当前典型的也是应用最多的移动感知中间件的组织架构如左图，类似于中心式客户端-服务器体系架构，移动感知中间件分别位于手机端以及中心服务器。例如PRISM和AnonySense。移动端的中间件主要负责数据的采集和处理，服务器端中间件则负责数据的进一步处理、整合并将数据以Web接口方式展示出或者供开发人员调用。但这种方案，对于部署大规模实验是不可行的。因为科研人员与被测手机本身就有着一种多对多的关系。那么全部的数据都要经过中心服务器，然后由服务器到真正的科研人员手中。这将增大能耗、带宽成本以及隐私泄露的风险，也会造成服务器的处理负担。其实无论是科研人员还是移动端被测人员的设备都有不错的处理性能，因此本文采用了右图的架构模式。该架构中存在三中角色，**设备所有者**向系统提供计算和感测资源；**研究人员**在通过部署实验来消耗这些资源；

**测试台的管理员**，也就是中心服务器决定哪些设备分配给哪些研究人员，类似于一个broker或者DNS服务器的角色。感觉，完全可以和DNS服务器合并= =。这样，就可以跳过broker直接进行数据传输。

Broker可以被看成消息转发器。Broker可能也负责一些控制和管理操作。它能够定位服务端的位置，若发生异常，能够将异常捕获传给Client。Broker需要提供注册服务的接口给Server。如果请求来自其他的Broker，本地的Broker需要转发请求并最终将结果或异常回应给相应的远程Broker。Broker提供的服务和name service非常相像（如DNS、LDAP）。

至于如何开发具体的实验应用，通常有两种方法，一种是Runtime-conﬁgurable systems ，另一种是domain-speciﬁc languages。首先，通过PhoneGap工具整合GPS、加速度传感器等感知功能到相应的接口，这个接口可移随着传感器的增多而扩展，并采用发布订阅机制供上层调用。包括发布订阅，总共封装了11个API，开发者可以利用这11个API开发相应的实验应用脚本。这些实验应用脚本采用的是javascript语言。

PhoneGap是一个用基于HTML，CSS和JavaScript的，创建移动跨平台移动应用程序的快速开发平台。它使开发者能够利用IOS，Android，Palm，Symbian,WP7,WP8,Bada和Blackberry智能手机的核心功能——包括地理定位，加速器，联系人，声音和振动等，此外PhoneGap拥有丰富的插件，可以调用。



collector node(研究者)与device node（实验者）的结构如上图，其中context对应一个实验的沙盒，context中的message broker负责各个脚本之间的，与远程的message broker，以及与sensor manager的通信，通信的方式也是基于发布订阅机制。device node每一个context只存在一个massage broker，而collector node可以存在多个，其中每一个对应一个不同的实验体，然后经由一个multi broker进行整合，进一步与远程的应用脚本进行通信。另外，device node之间是永远不会彼此通信的。

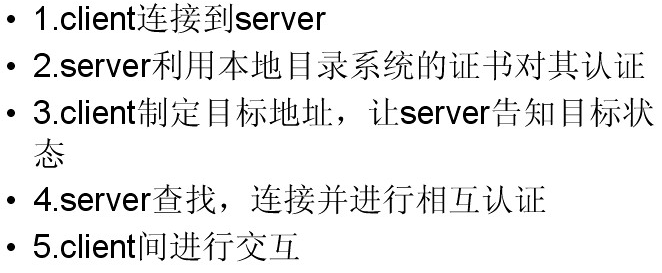
Message broker之间的协议采用MPTT协议，这是一种基于XML的即时通讯协议。

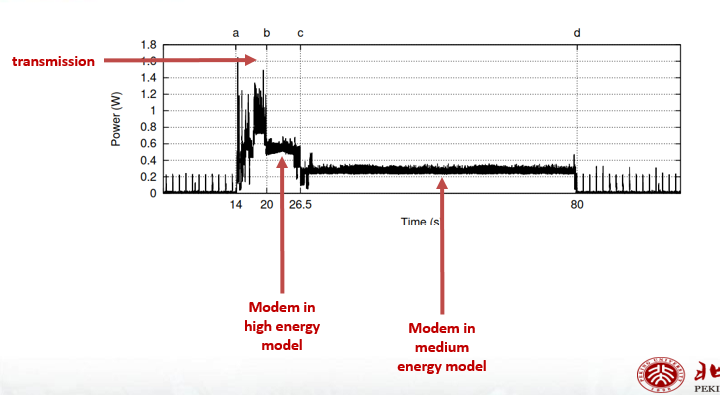
主要优点如下：

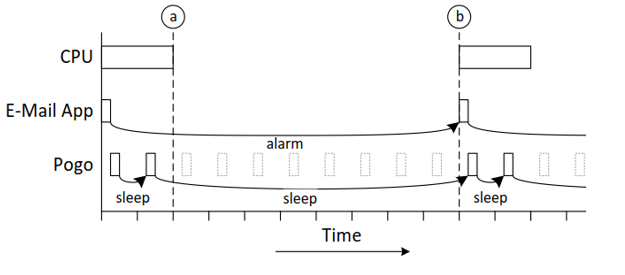
XMPP 协议是公开的，并不属于任何的机构和个人，而是属于整个社区，这一点从根本上保证了其开放性。

在XMPP 中，由于即时消息和到场信息都是基于XML 的结构化信息，因此具有良好的语义完整性和扩展性。

最重要的一点，XMPP的工作原理正好贴合broker的工作需求。







这一部分就是用来减少发送数据的电量消耗，通过在其他周期性数据发送应用的尾部能量开始时，发送自己的数据，从而起到降低电量消耗的作用。

注意，这里用到的方法不是利用alarm唤醒系统，因为以秒为单位的唤醒间距对cpu造成的开销是巨大的。而是利用java的thead.sleep()函数，当处理器处于睡眠模式时，管理睡眠行为的定时器也被冻结，这意味着线程将仅在CPU被某个其他进程唤醒之后才继续执行。 我们使用它来检测什么时候CPU被另一个应用程序唤醒，如果是一个想要进行数据传输的后台服务就会触发中间件的数据传输。

**Paper 2：SenSocial: A Middleware for Integrating Online Social Networks and Mobile** Sensing Data Streams

Abhinav Mehrotra @Middleware 2014

明天再整理