## BSmart客户端文档

1. 架构设计

简单的描述主要框架内容，详细模块介绍到类图设计与顺序图设计等进行深度的讲解。

* 1. BSmart的Android客户端整体设计如下图1。主要分为应用层Application(与Android平台相关)，领域业务层domain(与Bsmart有关的业务逻辑)和技术服务层service(提供基础服务的调用，比如网络请求，即时通信等)。
  2. 应用层：
     1. view包：主要提供自定义控件（view的子类）与应用的界面布局
     2. activity包：主要包含每个界面的控制器，属于MVC设计模式中的C的扮演者，Controller,主要负责对视图view的操控与向领域层domain请求相应的操作。
     3. service包：主要为实现即时通信多进程的架构部署，将为即时通信服务单独开设一个后台服务进程，通过android平台提供的AIDL，binder通信机制进行进程间的通信。Service进程主要负责监听用户的信息发送与接收事件，通过binder与前台进程进行信息通信。
     4. broadcast包：主要响应一些消息提醒，比如通过监听开机事件后启动后台服务进程和当后台进程接收到服务器推送的消息时以notification的形式通知用户。
  3. 领域层
     1. context包：主要负责管理业务层的逻辑操作，包含一个基类BsmartContext，具备描述整个应用的上下文信息，比如存储路径，数据加载方式等，和统一向子类提供底层服务模块的接口service包。利用工厂模式结合单例类模式来保证服务模块的单一对象。其中BsmartContext子类包括InfoProvider负责用户基本信息的管理，PayManager负责支付功能的处理，IMManager负责即时通信模块的处理，IMSession与IManager配合负责每个对话的管理。LoginManager负责登录注册与登录状态的管理，DynamicManager动态消息模块的管理等。
     2. remote包：封装了服务器提供的API，以实现类似RPC的操作，是业务层透明化的调用服务器接口。其中依赖服务service包的http子包。其中内部利用线程池来实现多线程的网络请求。
     3. entity包：拥有业务层所需要的实体对象类，负责数据的存储，序列号与反序列，所有实体类都继承自Entity基类，基类利用java的反射机制实现json的序列化与反序列化操作，与ContentValue的数据存储从而大大减轻程序员手动序列化json或数据库存储操作。
  4. 服务层
     1. http包：主要抽象化网络请求接口，封装了java提供的url请求库，以更好的为上层提供简单的网络请求操作。
     2. database包：抽象化数据库操作接口，封装android平台提供的sqlite数据库操作。
     3. im包：抽象化即时通信接口，封装smack第三方即时通信库的以更好利用观察者模式，向领域层消息的多路广播。
     4. utils包：提供一些全局的操作程序。
     5. gps包：抽象化定位接口，封装百度的gps库。
     6. entity包：为实体类的基类Entity,EntityGroup,负责实体对象数据的序列化与反序列化
     7. netstate包：抽象化网络状态接口，封装android平台提供的网络状态监测服务，以观察者模式，向上层实时的通知网络状态。
     8. security包：负责数据的加密解密。
     9. threadpool包：负责线程池的配置与调度与线程的提供。
  5. 总体架构设计理念
     1. 总体使用MVC模式，将视图层与业务层很好的隔离，利用android平台提供的activity充当中间者。以实现代码高度低耦合。
     2. 将基础服务抽象并封装与业务层隔离，更好的进行模块化设计，减轻业务层的代码复杂度与冗余度，从而实现高内聚效果。
     3. 将业务层实现模块划分，每个模块充当对应的上下文，类似于操作系统进程的概念和android平台activity与context的设计理念，来更好的实现业务层的功能分工来负责相应视图层的请求。在应用开启后可以有顺序与协调的加载每个模块，比如缓存数据的加载，即时通信服务的开启，登录状态的管理等等。
     4. application包依赖domain包，domain包依赖底层service包。

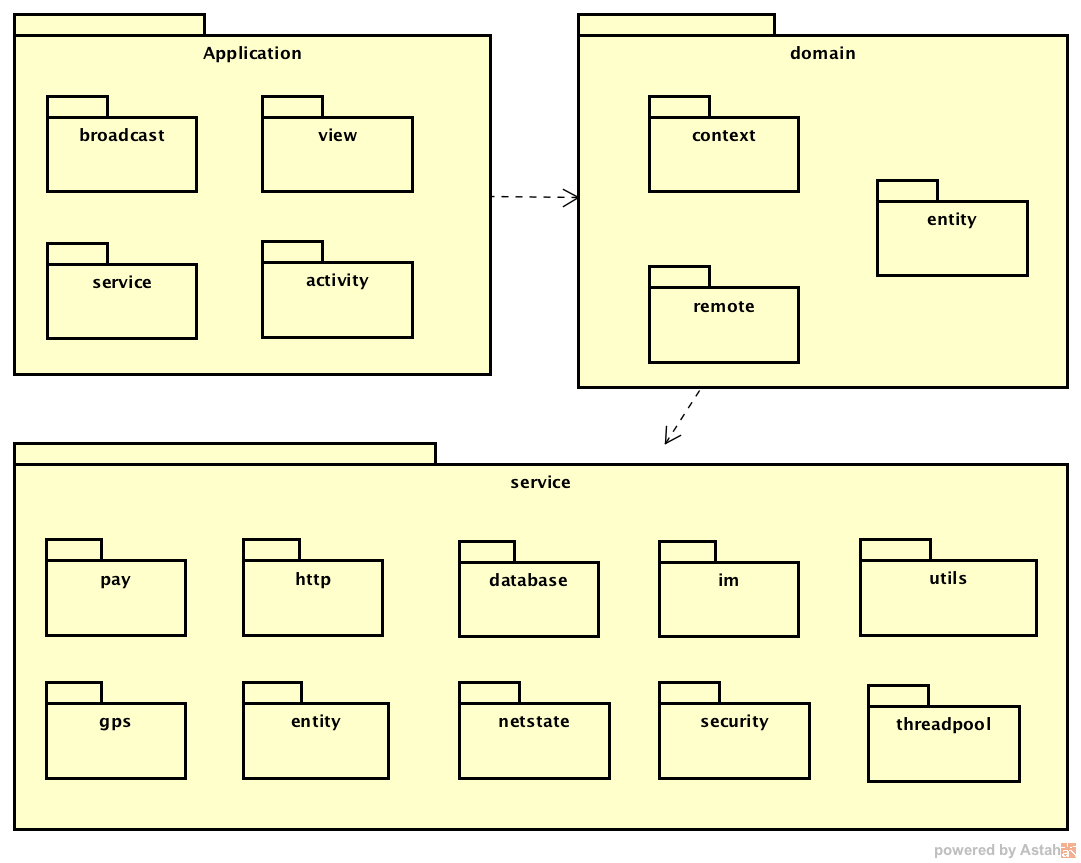


图 1 包图

1. 类图设计
2. 模块设计

对每一个层选择具有代表性的子模块进行详细介绍。

* 1. 服务层：
     1. Http模块：
        1. remote包属于领域层模块
           1. RemoteServer包含服务器为客户端提供的所有接口，类图中只展示了登录和上传头像接口，通过业务层调用RemoteServer对象的函数接口，进而调用HttpManager相应的请求函数，HttpManager通过RSAEncrypt进行数据加密后向底层服务层的HttpRequest进行真正的http网络请求，中间涉及到数据的分发与数据的缓存（利用FIFO的形式缓存最近的请求，避免重复的进行网络请求）。其中HttpListener为异步进行时提供回调到业务层的作用。HttpResponse为请求返回数据的格式描述。
        2. Http包为服务层模块
           1. HttpRequest为http网络请求的核心模块，实现了文件下载上传接口与http协议的GET与POST方法。
           2. Response为响应数据的描述
           3. Request为请求的数据格式描述
        3. Security包为安全加密模块
           1. 利用非对称RSA加密对数据进行加密。对数据进行Base64编码与解码。
        4. 图3为一个用户登录的时序图。
           1. 从RemoteService发起，调用login()接口，
           2. RemoteService将请求异步提交给HttpManager，
           3. HttpManager利用线程池提供的线程通过加密模块接口encrypt()进行数据的加密。
           4. HttpManager将加密过的数据后的请求提交给HttpRequest进行真正的服务器Post请求。
           5. 服务器响应请求，HttpRequest将响应的数据封装为Response响应对象后返回到HttpManager。
           6. HttpManager将通过监听接口HttpListener回到业务层。

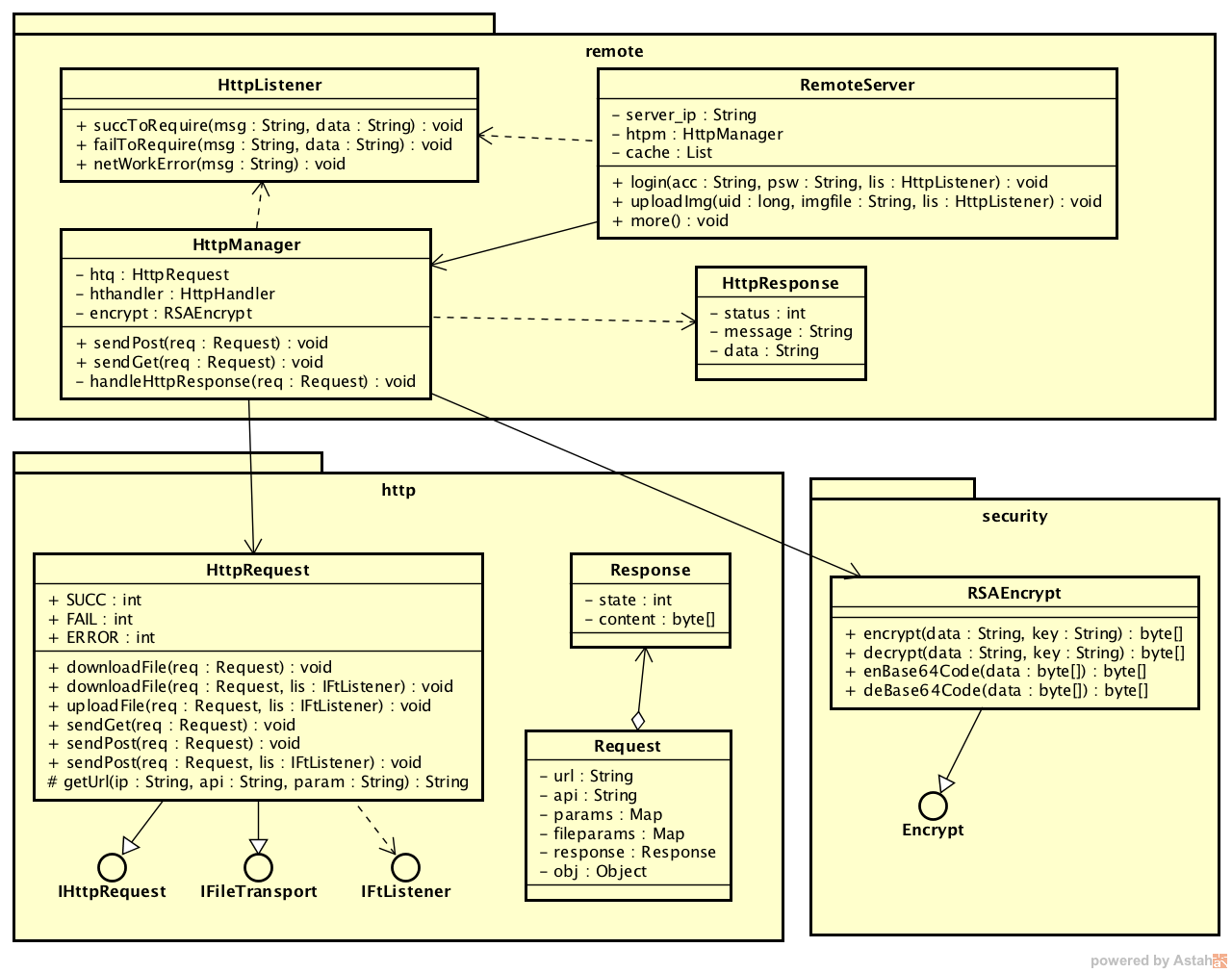


图 2 Http模块

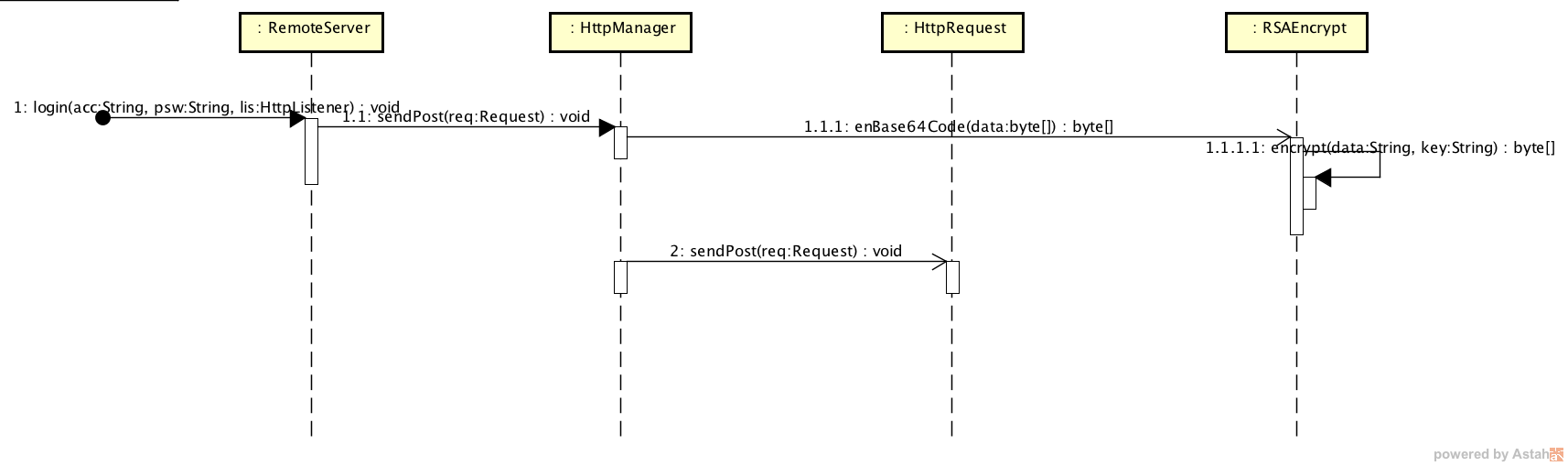


图 3 http请求时序图

* + 1. IM模块:（多进程介绍在后文）
       1. 此项目的通信模块的底层实现是依赖第三方开源库smack+openfire实现，模块IM包抽象封装了此库，为上层领域层提供更好的通信接口，其中声明了IMCallback回调接口，IIMService统一抽象通信接口，IMListener监听信息回调接口。在IMService类中设置了全局广播，系统消息通知，私人会话机制。
       2. 在领域层的Context包中，包含IMManager,IMSession模块，主要作用为在业务层中实现更好的管理会话，IMManager负责管理每个会话过程与系统全局消息广播，IMSession负责管理具体每一个会话，中间过程包括数据加密解密的发送与接收，会话记录的本地缓存等。
       3. 用户发送信息的顺序图为图5.
          1. 从IManager出发，调用genIMSession接口创建一个会话Session.
          2. 通过第一步创建的会话进行信息的通信，IMSession通过sendMessage()接口，现将数据通过RSAEncrypt进行加密后，向底层IMService服务模块提交请求。
          3. IMService将指定的请求向服务器发送。

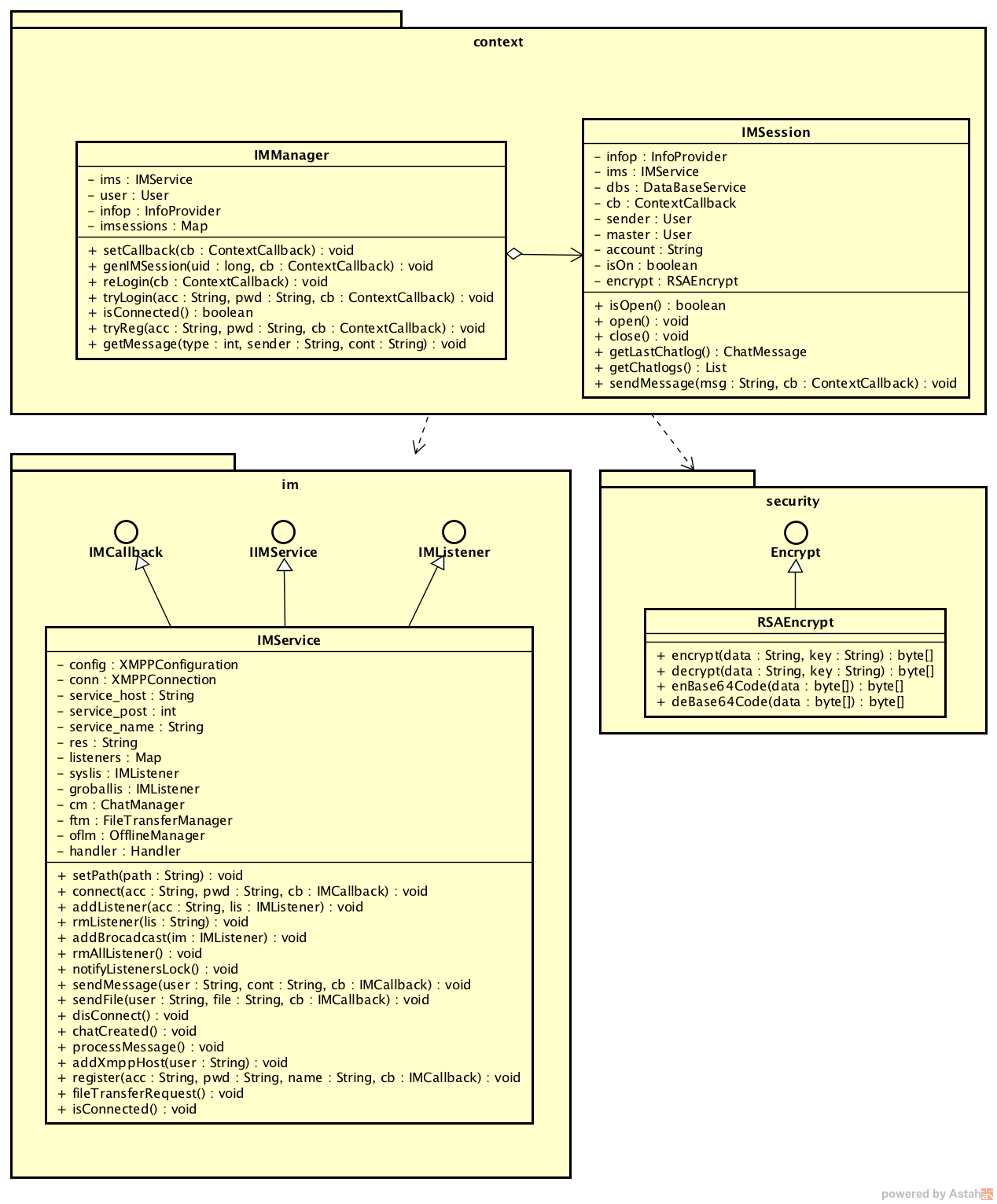


图 4 通信模块

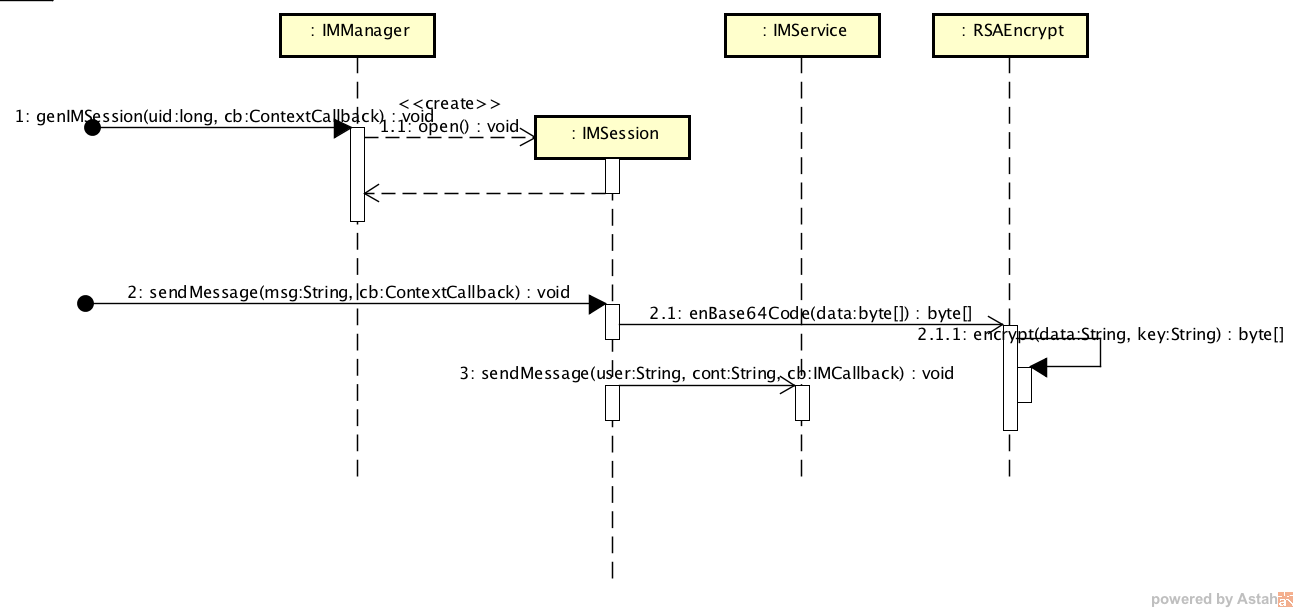


图 5 通信模块顺序图

* + 1. 应用层的通信前台与后台的多进程设计
       1. 多进程设计采用C/S架构模式，基于Android平台提供的Binder机制实现的，利用AIDL为客户进程Client与服务进程Server分别定义两套接口即：IMServiceProxy和IMServiceStub。
       2. 下图6中，左边为服务进程，右边为客户进程，利用Android平台提供Service组件，在客户进程中创建一个客户服务MsgClient以接收服务进程发送的信息，服务端创建一个MsgServer服务以响应客户进程发出的请求，比如发送消息或文件等。
       3. 图7为顺序图。
          1. 从IMServiceProxy的startServer入口启动服务进程的MsgServer,
          2. MsgServer启动客户端服务MsgClient，至此客户端进程与服务端进程建立起连接。
          3. 从IMServiceProxy的sendMessage()入口从客户端向服务端进程发起发送消息请求。

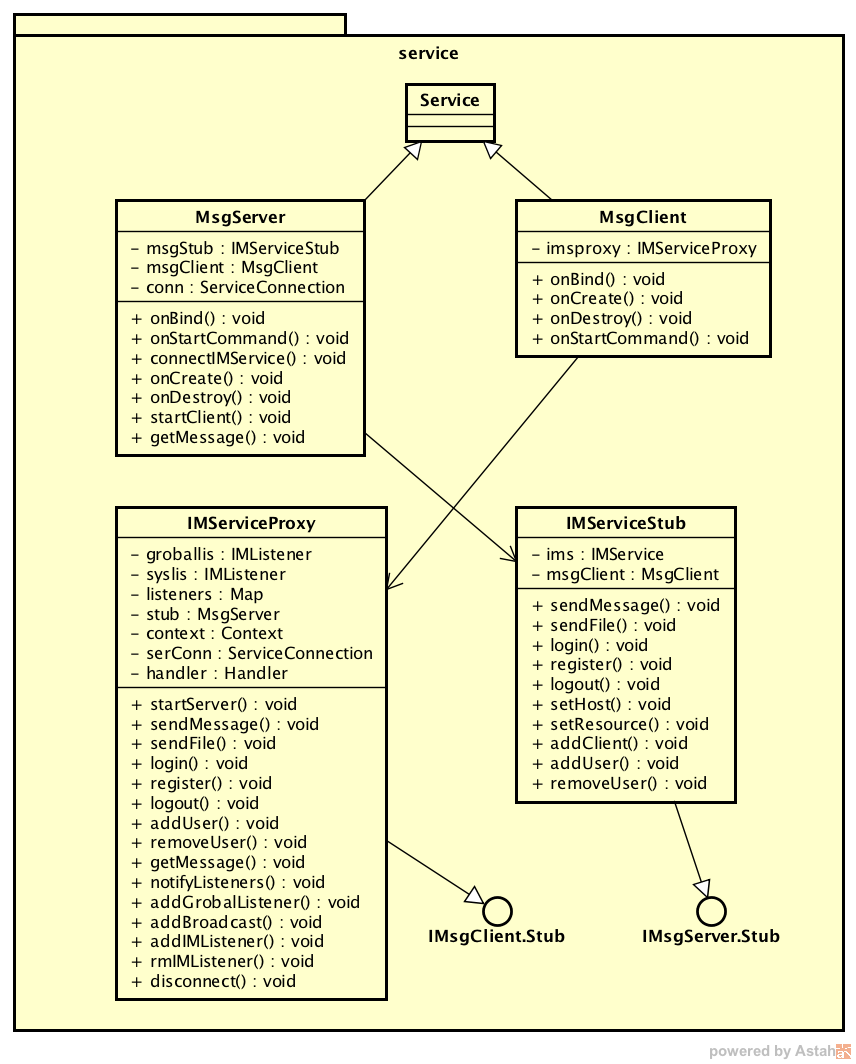


图 6 通信模块多进程设计

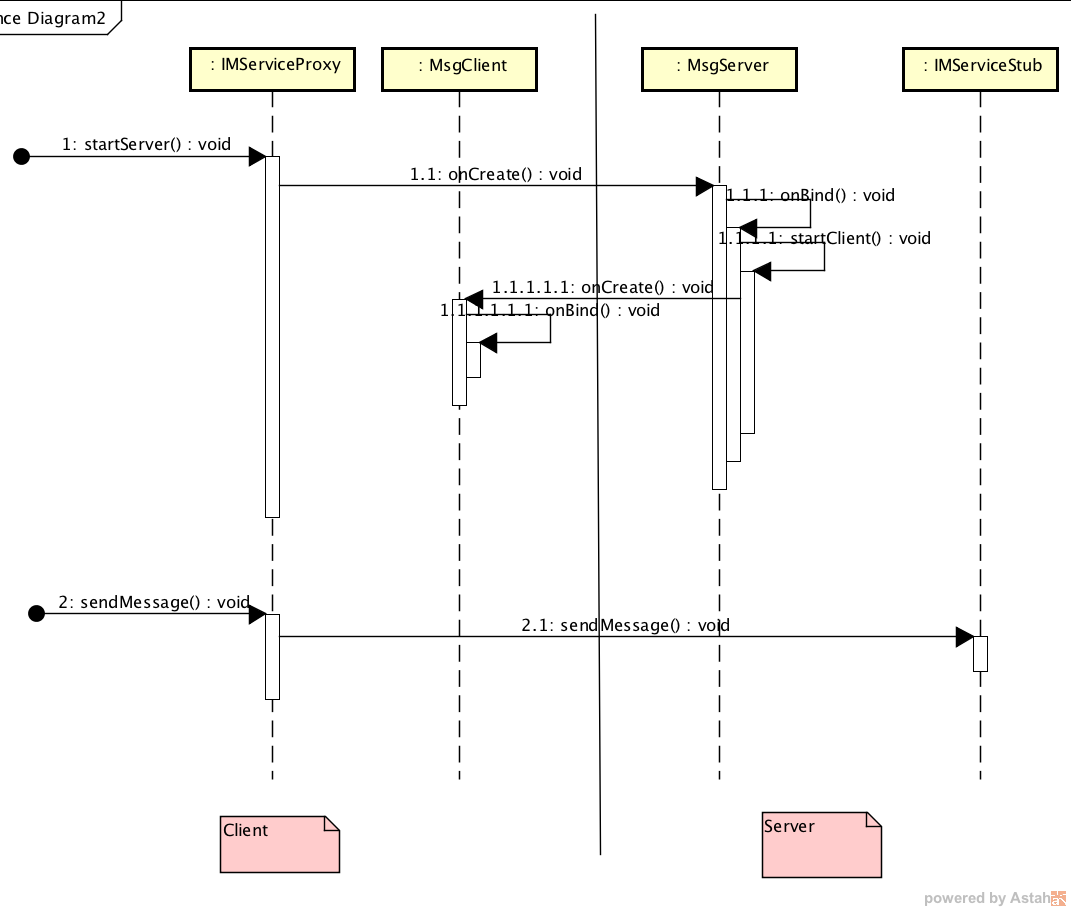


图 7 多进程间通信图

* 1. 领域层：
     1. Context模块设计：
     2. LoginManager模块：
     3. 其他还有PayManager,InfoProvider,IMManager等
     4. 下图8中展示了BSmartContext的设计，BsmartContext包含了业务层所需要基础服务模块，子类通过继承父类可以快速的获取响应的服务对象，比如登录时需要网络请求可以获取Http，IM服务对象，需要本地缓存时可以获取数据库对象等。其中ServiceManager采用了工厂模式与单例类模式的设计方法，对基础服务模块进行有效的管理。
     5. 图9为应用启动时，有android入口Application调用BsmartContext进而初始化基础服务类与业务层的各个上下文。然后在通过登录界面进入LoginManager,调用trylogin()过程中，LoginManager需要从基类对象获取基础服务对象比如http,imservice,databaseservice等进行服务器登录请求与本地数据加载。

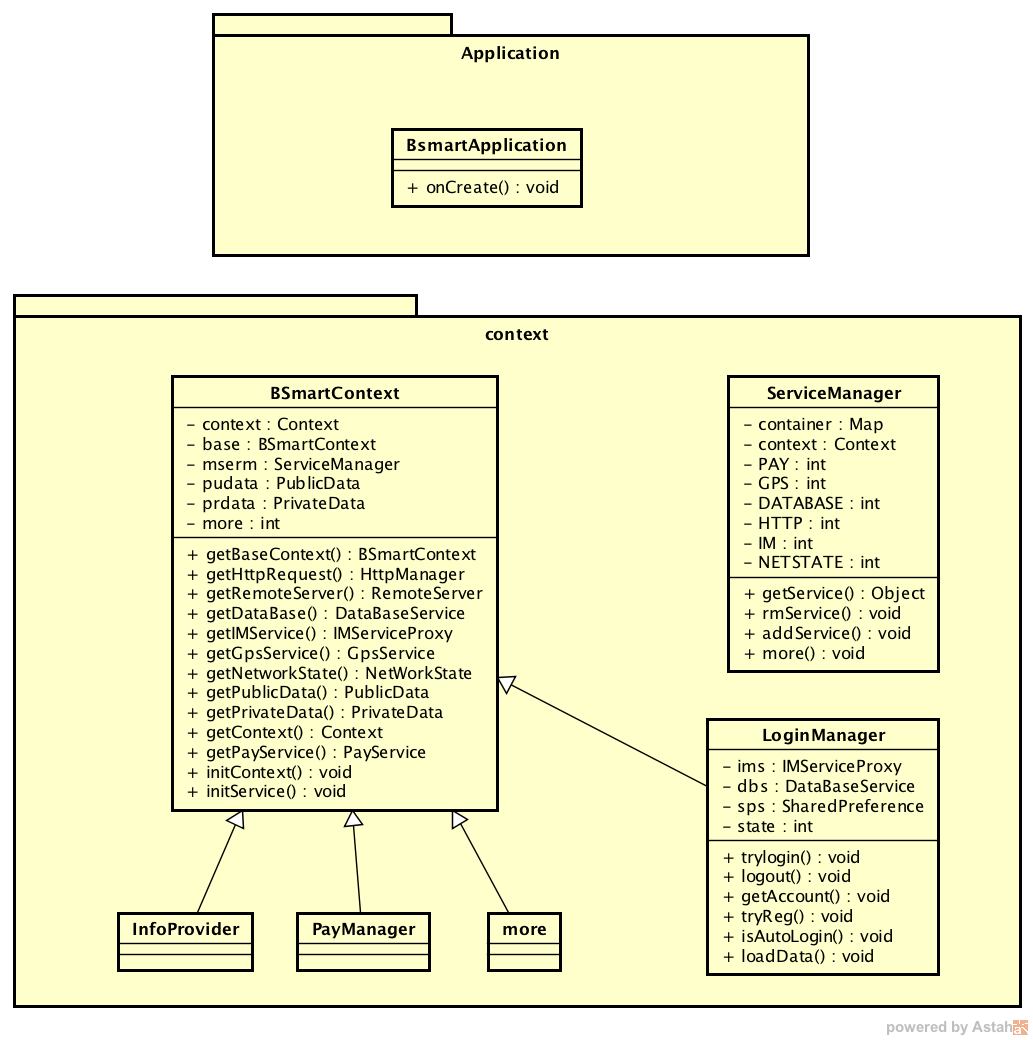


图 8 Context设计

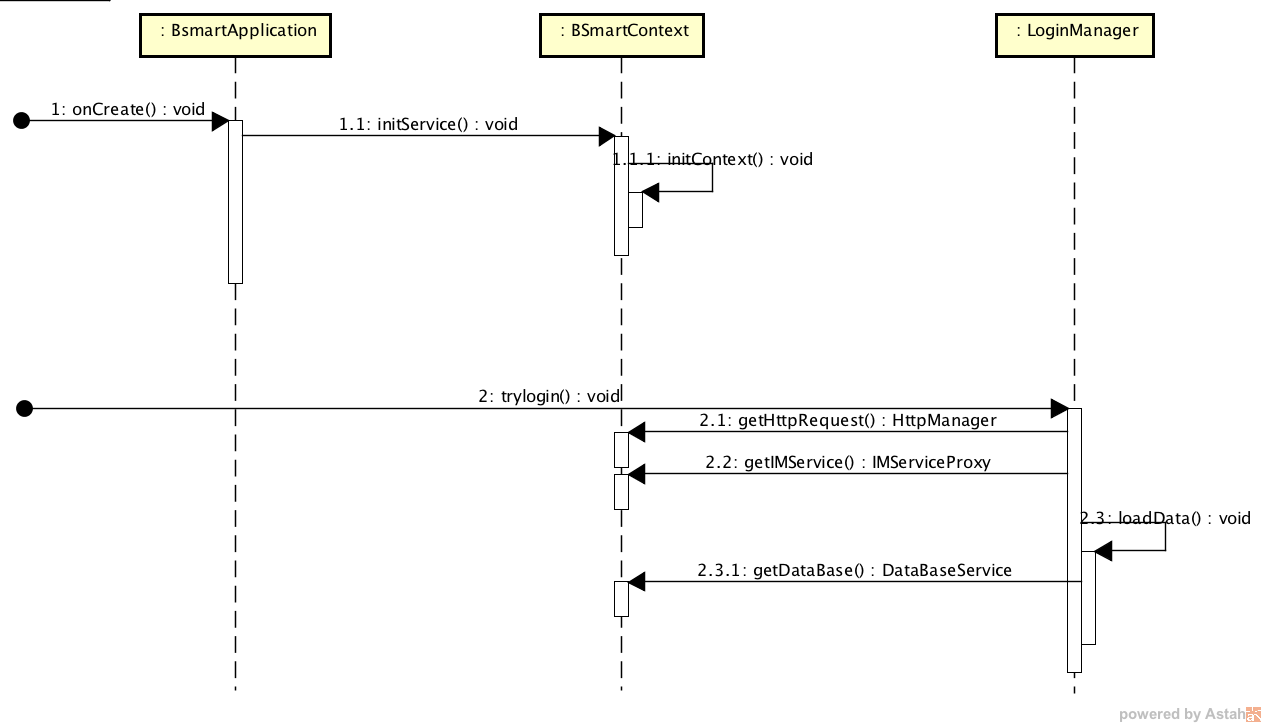


图 9 应用启动时序图

* 1. 应用层：
     1. ListView优化模块
     2. 自定义控件
     3. UI界面

1. 总结
   1. to do
2. 总结
   1. to do