# 2. 平台设计与实现

## 2.1 平台模型

### 2.1.1 模型概述

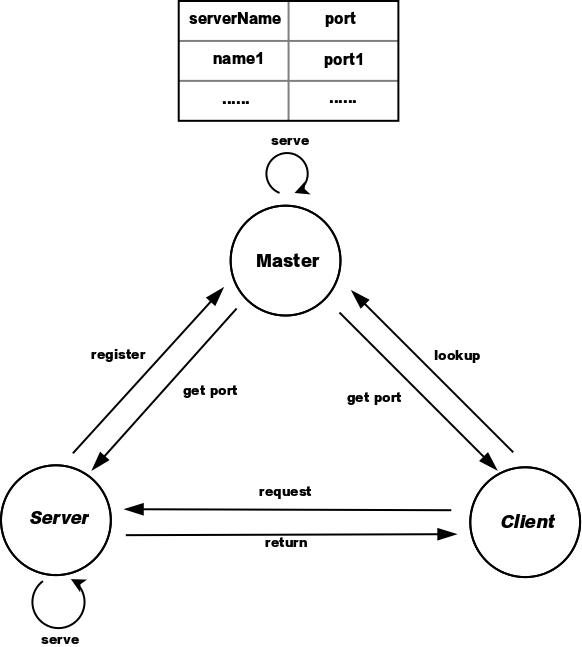
本软件平台使用Master-Server-Client多层次结构。

Master作为中心节点，拥有一个始终固定的通信标识符。Master负责记录和管理所有Server的通信标识符，需要在软件平台使用过程中始终保持在运行状态，

Server提供各类服务，可以在需要时开启，并在Master处进行注册，获得一个唯一的通信标识符。Server开启后被动地等待来自于Client的服务请求，进行处理并返回结果。

Client用于实现具体应用的业务逻辑，主动向Server请求服务。当需要某种服务时，先从Master处查询到对应Server的通信标识符，之后向Server请求服务。

架构模型示意图如下



### 2.1.2 Server

Server用于提供特定的服务，并根据服务的数目和所需的参数，定义通信消息的结构。

每种Server拥有一个唯一且固定的名称，并利用该名称，在Master处进行注册。Master将Server名称与为其分配的通信标识符进行绑定，并记录在表中，以备Client查询。

Server获得通信标识符后，即可开启服务，等待来自于Client的服务请求。

为了使Client能够正确地请求服务，Server需要定义一个自己特定的消息结构，指明为了向自己请求服务，需要填写的申请内容。

当Server收到请求服务时，读取请求服务消息，并根据消息的具体内容执行预定义的操作，进而返回特定的结果给Client。之后，Server继续等待下一个服务请求的到来。

### 2.1.3 Master

Master是软件平台的中心节点，记录和维护Server的注册信息。

事实上，Master可以看做一种特殊的Server，为除自己之外的其他所有Server提供注册服务，并为Client提供查询指定名称Server的服务。

Master拥有一个输入消息类型定义和一个输出消息类型定义。当请求服务者为Server时，该Server需要填写注册信息，包括表明此次请求是来自于Server的标志，以及自己的Server名称。当Master接收到来自于Server的注册服务请求时，会将该Server的名称与一个唯一的通信标识符绑定，进行保存，并返回该通信标识符。此时，输出消息包含的是为该Server分配的通信标识符。

如果向Master请求服务的是Client，则需要在输入消息中写明此次请求来自于Client，并指出之后需要的服务的Server名称。Master接收到该请求后，在注册信息表中查找该Client所需要的Server的通信标识符，并返回给Client。此时，输出消息包含的是Client查询的Server的通信标识符。

### 2.1.4 Client

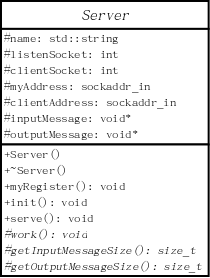
Client主要用于实现具体应用程序的业务逻辑，在需要时向Server请求服务。

当Client需要某个Server提供的服务时，首先利用该Server的名称，向Master请求查询服务，获得Server的通信标识符。之后，Client可以直接与Server进行通信，而不再需要Master的干预。Client遵守Server所定义的消息规范，向Server发送消息请求服务，并接收返回结果。

## 2.2 架构实现

本软件平台采用多进程运行模式，利用socket进程间通信机制进行消息传递。

### 2.2.1 Server类



Server类是所有服务提供类的抽象基类，一个服务器进程只需要创建一个Server派生类的对象，并执行Server::serve()操作开启服务即可。

Server派生类的对象被创建时，利用Server::myRegister()在Master类处进行注册。Master类为该Server派生类对象分配一个唯一的socket描述符，并将其作为通信标识符，与Server派生类对象的名称进行绑定，记录下来。

之后，该对象可以调用Server::init()进行初始化，然后执行Server::serve()操作，利用socket进行进程间通信，等待来自于Client的服务请求，进行相应的处理并返回一定的结果。

为了提供具体的某种服务，开发者需要继承Server类，在构造函数中为Server派生类命名并调用注册函数Server::myRegister()，并实现其中的纯虚函数getInputMessageSize()，getOutputMessageSize()和work()。

为了向Client说明如何向Server派生类对象请求服务，开发者需要为Server派生类定义输入消息结构体和输出消息结构体，用以标明如何向Server派生类对象请求某个服务，该服务所需要的参数，以及服务返回消息的数据结构。

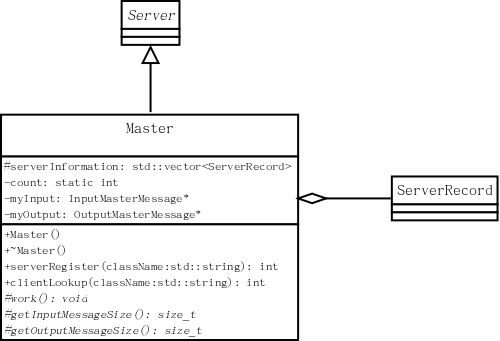
开发者需要实现Server类中的纯虚函数getInputMessageSize()和getOutputMessageSize()，返回输入消息结构体的大小和输出消息结构体的大小，为软件平台内置的通信功能提供必要的支持。

此外，开发者需要实现Server类中定义的纯虚函数work()，在其中编写具体的服务代码。在Server::work()函数中，需要从输入消息结构体对象所在的inputMessage缓冲区中读取服务请求消息，针对不同的消息内容，执行不同的任务，并将结果写入输出消息结构体对象所在的缓冲区outputMessage中。

Server::serve()函数执行后，等待Client请求服务。当收到服务请求时，执行开发者实现的Server::work()操作，并通过socket将结果返回给Client。之后，继续执行Server::serve()操作，等待下一个服务请求，重复上述操作。

### 2.2.2 Master类



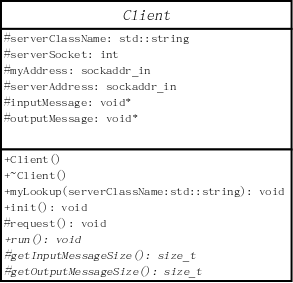


Master类作为一种特殊的Server类，继承自Server类，并实现了纯虚函数getInputMessageSize()，getOutputMessageSize()和work()。

Master类对象运行在一个单独的进程，在使用软件平台的整个过程中始终保持运行，并拥有一个众所周知且唯一的socket描述符作为通信标识符。

Master类实际上就是一个提供具体服务的Server派生类，工作机制与Server基本相同。serve()函数执行之后，等待来自于Server派生类对象的注册服务请求，或者是来自于Client派生类对象的查询服务请求。work()从inputMessage缓冲区中读取请求消息。根据InputMasterMessage::type判断此次请求的发起者，并提供相应的服务，再将结果写入outputMessage缓冲区。之后，由serve()中的操作通过socket将结果返回给请求者。

### 2.2.3 Client类



Client类是所有实现具体应用程序业务逻辑代码的类的抽象基类。

每个Client派生类对象运行在一个独立的进程中，利用需要服务的Server名称，通过周知的Master类对象的socket描述符，向Master查询Server的socket描述符。之后，Client进程可以直接通过socket与Server进程进行消息传递，请求服务并获得结果，而不再需要Master的干预。

开发者需要继承Client类，并实现getInputMessageSize()，getOutputMessageSize()以及run()三个纯虚函数。

向Server派生类对象请求服务时，必须遵守该Server派生类所定义的通信消息规范。getInputMessageSize()和getOutputMessageSize()分别返回提供该Server派生类所定义的输出消息结构体大小和输出消息结构体大小。

run()函数中，可以实现业务逻辑。需要某个Server服务时，调用Client::myLookup()向Master查询该Server派生类对象的socket描述符，之后执行run()操作。在run()函数中，根据Server派生类所定义的消息结构体，向outputMessage缓冲区写入消息内容。

之后，调用Client::request()函数发送消息，并接收返回结果。返回结果会保存在inputMessage缓冲区中。

可以看出，当请求服务的Server派生类确定之后，该Server派生类的输入消息，实际上就是Client派生类此时的输出消息。该Server派生类的输出消息则为Client派生类此时的输入消息。

当Client需要不同的服务时，随着Client::myLookup()请求的Server名称不同，应该对inputMessage和outputMessage的类型做相应的调整。

## 2.3 模块划分

在软件平台的基础上，我们开发了多种Server，提供了语音输入、语音输出、华南理工大学图书馆查询、摄像头、Qrobot机器人等服务，并利用一个包含了业务逻辑的Client，利用各个Server提供的服务，组成了一个完整的应用程序。

**语音输入**

语音输入模块的主要功能由SpeechToText类提供。SpeechToText类利用音频支持库ALSA，从音频输入设备获取语音，并对其进行识别，从而生成对应的文本。

**语音输出**

语音输出模块的主要功能由TextToSpeech类提供。TextToSpeech类同样利用音频支持库ALSA，接收字符串作为输入，利用音频输出设备进行播放。

**图书馆查询**

图书馆查询模块的主要功能由Serverlibrary类提供。Serverlibrary类提供了查询用户个人信息、查询图书等功能。

**摄像头**

摄像头模块的主要功能由ServerCamera类提供。ServerCamera通过OpenCV库，

**Qrobot机器人**

Qrobot机器人模块的主要功能由ServerQrobot类提供。ServerQrobot类调用QrobotController类的操作，实现对机器人的间接控制。

**主客户端**

MainClient类提供了主客户端模块的功能。MainClient通过调用语音输入、语音输出、图书馆查询、摄像头、Qrobot机器人等模块提供的服务，为用户提供了一个完整的应用。

## 2.4 开发环境

编程语言： C++

Python

操作系统： Ubuntu 14.04 LTS

Ubuntu 13.10

编辑器： gVim

gedit

编译器： g++

版本管理工具： Git

版本库地址： https://github.com/Isilendil/socketPlatform

## 2.5 依赖库

图像处理支持库 OpenCV

音频支持库 ALSA

Qrobot支持库

Python HTML/XML处理模块 BeautifulSoup

python c++ 支持库