**第五章 组网功能**

**组网设计方案**

如图5-1所示，我们的组网方案包括水声节点、移动节点、浮子节点、汇聚节点和系统管理分析中心，可同时支持水声通信点对点测试、水声组网、水声定位测试。系统管理分析中心是一台PC机，负责测试系统的控制、数据管理和分析，通过网线跟汇聚节点进行点对点有线通信。

系统汇聚节点和浮子节点具有相同的硬件电路结构，软件功能不同。系统汇聚节点、浮子节点负责透明转发水声节点与系统管理分析中心之间的命令、数据信息和状态信息。系统汇聚节点和浮子节点构成了验证平台的监控网络层，两者之间通过远程WIFI通信。移动节点拟采用水下无人航行器或无人船（悬吊水声节点）。水声节点支持软件可编程、逻辑可编程，通道可配置。



图 5-1 组网方案

**网络协议栈架构**

如图5-2所示，水声通信节点的网络协议栈基于五层协议体系而设计，与传统协议栈不同，本协议栈中的每一层可以存在多种协议，在运行时再通过配置文件选择具体运行的协议。而且为协议之间的跨层通信提供了简单易用的接口。主要由六个部件组成：

(1) 协议(Protocol)：协议模块，包含某种协议的具体实现，由用户编写。

(2) 调度器(Scheduler)：负责跨层信息和I/O 事件的调度。

(3) 服务接入点(Service Access Point, SAP)：与协议和调度器相连，用于实现协议间跨层通信。

(4) 驱动(Driver)：用于驱动通信机物理层收发，使得数据能够发送到真实信道中，或者驱动信道仿真器以进行仿真实验。

(5) 设备接入点(Device Access Ponit, DAP)：与SAP角色类似，为驱动和真正设备之间提供接口。

(6) 设备：指通信机的物理层实现，包括收发前端、物理层通信处理。

协议栈中提供串口、Socket等多种设备接口，用户通过添加新的驱动使用相应的设备实现了协议栈驱动AquaSeNT Modem与声学所通信机。同时系统实现了一个简单的模拟信道，拟引入WOSS(BELLHOP)。

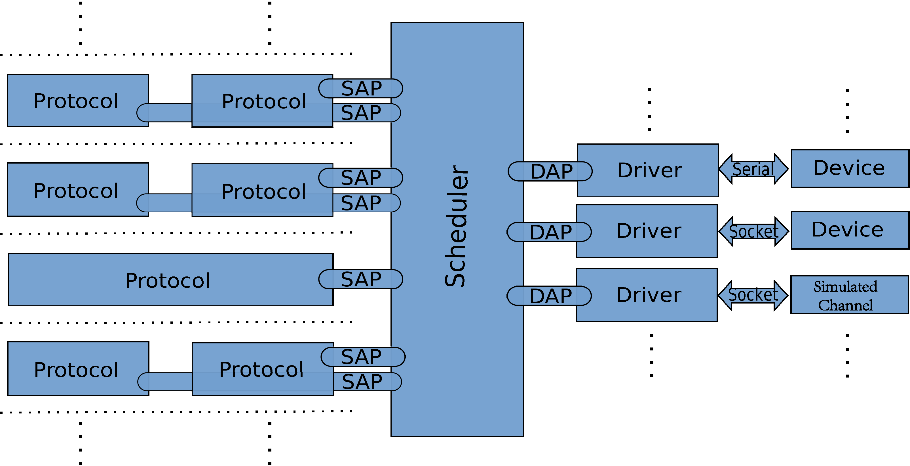


图5-2协议栈的总体架构

整个协议栈由事件驱动，当有IO 事件或者跨层通信事件发生时，调度器就会调用相应的模块执行操作。整个流程就如图5-3所示。本协议栈的总体设计主要参考了NS2的扩展NS-Miracle。整个设计和实现都比较简单明了，所需的系统资源也不多，甚至能够运行在一些嵌入式环境中。协议栈中使用层次状态机(Hierarchical State Machine, HSM) 的方式编写协议，用户能够很简单地将协议逻辑转化为代码，提高编程效率。每个协议都是以模块的形式存在，增加了使用时的灵活性，而且提供了易用的跨层通信接口，每种协议都可以很简单地与其它协议进行通信。协议栈中提供串口、Socket 等多种设备接口，用户只需添加简单添加新的驱动就能使用相应的设备。协议栈的设计与实现详见相应文档。

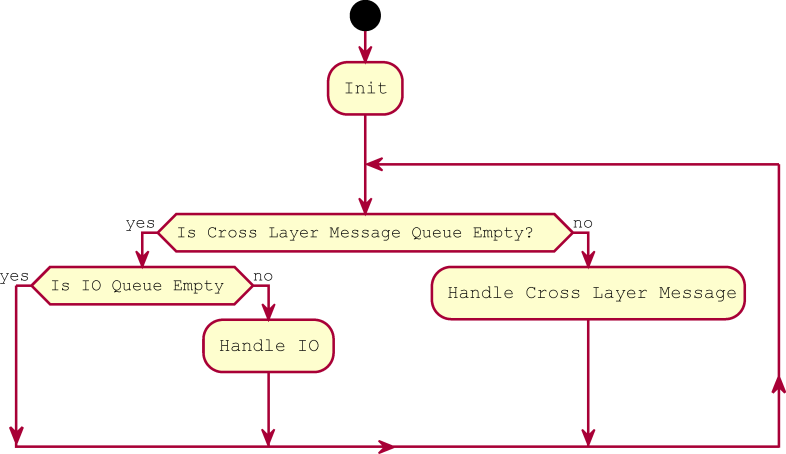


图5-3 协议栈事件处理流程图

**基于层次状态机的协议设计**

协议是协议栈的主体部分，协议栈必须有真正的协议才能运行起来。本协议栈为用户编写协议提供了一个灵活的环境以及很多有意义的帮助，其中最大的帮助是为用户提供了一个层次状态机框架，使得用户只需要把精力集中在协议逻辑，而无需考虑协议的状态转移。

状态机是一种用于表示有限个状态及这些状态间的转移和动作的离散数学模型。状态机在网络协议的设计中起着重大的作用，因为网络协议本身就是一个状态机。如果我们把协议划分出一些状态，把协议内部的原语定义为输入，协议中的处理定义为动作，协议就被抽象为状态、输入、动作、状态转移和处理逻辑的集合，加上初始状态，网络协议就符合状态机的定义。由于状态机可以将问题整体的分解成各个部分状态及跳转，直观地对系统进行建模，所以它不仅被用于理论研究过程当中，而且被广泛用于操作系统和网络协议栈的程序设计中。

在传统的状态机中，必须为每种可能的参数组合都建立一个状态，这会导致大量的状态和转移，就算是在一个很简单的系统中。层次式状态机，又叫状态图(Statechart)，能够减少这种复杂性和增加状态图的可读性。层次状态机最大的特点就是引入了层次式状态，与面向对象编程语言中的继承类似，一个状态可以继承另外一个状态，如果不对某种输入的动作进行重载，则该状态会执行与父状态一样的动作。可以通过把不同状态对某些输入的动作抽象出来，构造成父状态，对系统进行了逻辑上和实现上的简化，降低系统的复杂度。如图5-4为水声ALOHA协议的层次状态机图，其中有的TOP状态是继承自根状态，里面包括一些最基本的处理；具体实现时包括：空闲状态IDLE、ACK等待WaitACK、退避Backoff三个主要状态，跟VBF路由协议跨层处理时，还有WaitVbf状态。状态之间在一些条件或者事件的驱动下可以相互转换。

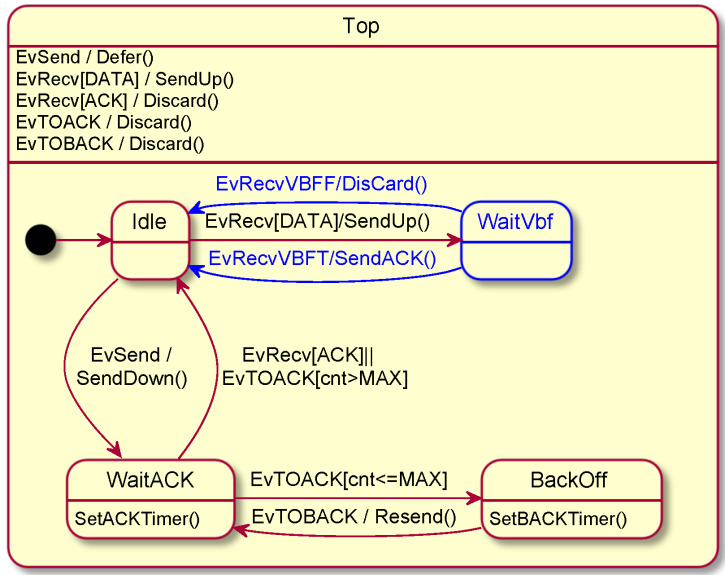


图5-4 水声Aloha协议的层次状态机图(其中蓝色为跨层处理)

如图5-5所示，基于层次状态机的协议**设计**主要有两个步骤：(1)将协议按层次状态图进行组织；(2)对协议中的状态、事件、动作、转换等事项进行准确的定义。



图5-5基于层次状态机的协议设计步骤

下面以图5-4所示的水声Aloha协议为例，介绍基于层次状态机的协议**实现**步骤：

(1) 定义协议类

struct AlohaHeader // Define the header

{ uint8\_t type ;// packet type

uint8\_t src ;// source address

uint8\_t dst ;// destination address};

// Declare the protocol class

class Aloha : public mod :: Module <Aloha , CURRENT\_LAYER ,CURRENT\_PROTOCOL >

(2) 宣称事件

struct Top;

struct Idle ;

struct WaitAck ;

struct BackOff ;

struct WaitVbf ;

(3) 宣称状态

using msg :: MsgSendDataReq ; // EvSendReq

using msg :: MsgRecvDataNtf ; // EvRecv

using msg :: MsgTimeOut ; // EvTOBACK or EvTOACK

(4) 定义状态并宣称处理函数(以WaitAck为例)

struct WaitAck : hsm :: State < WaitAck , Top > {

HSM\_TRANSIT\_DEFER ( MsgRecvDataNtf , Idle ,

&Aloha ::ReceiveAck , & Aloha :: IsAckPacket );}

struct WaitAck : hsm :: State < WaitAck , Top >

{

typedef hsm\_vector <MsgRecvDataNtf , MsgTimeOut>reactions ;

HSM\_TRANSIT\_DEFER ( MsgRecvDataNtf , Idle ,

&Aloha:: ReceiveAck ,& Aloha :: IsAckPacket );

HSM\_TRANSIT\_TRANSIT ( MsgTimeOut ,

BackOff , &Aloha :: HandleTimeouMsg,

& Aloha::HandleTimeoutAck, Idle );

};

(5) 定义动作函数(以SendUp为例)

void Aloha :: SendUp ( const Ptr < MsgRecvDataNtf > & ntf)

{ // Log system is also implemented

LOG ( INFO ) << " Aloha send up data ";

ntf -> packet . MoveUp < AlohaHeader >();

SendNtf (ntf , UPPER\_LAYER , UPPER\_PROTOCOL );

}

上述过程只有最后一步稍微复杂一些，需要自己根据协议的设计来撰写代码外，其余步骤相当简单甚至可以自动完成。

基于层次状态机的协议设计与传统的协议设计方法的比较，各自的特点如图5-6所示。



图5-6 基于层次状态机的协议设计与传统的协议设计方法的比较

**协议栈与通信机的驱动（陈仲恒）**

主要

**网络性能分析指标与实现（毛姗姗、王俊杰、陈仲恒）**

主要

**组网演示界面（郑杰文）**

主要

**水声组网测试（杰文恒姗）**

多节点多上位机水下组网测试



图5-X 组网测试场景

多节点有线局域网汇聚到单上位机组网测试



图5-X 有线组网测试配置

图5-X 组网测试场景（放水池图片）

测试结果

异网段

同网段

多节点无线局域网汇聚到单上位机组网测试(同网段)

主要

图5-X 无线组网测试配置



图5-X 组网测试场景