《通信协议软件设计》课程作业一

1. **请说明你对计算机网络协议的理解：为什么需要计算机网络协议？举一个协议例子说明该计算机网络协议的产生的原因、以及该协议所体现的算法（分布式计算算法）。**

为了在计算机网络中传递数据跟够有序、稳定、无误、使双方都能理解，且能够应对各种突发的不正常情况，信息交流的参与者必须都按照定义好的统一的一套用于交流的规则来进行通讯。计算机网络协议就是为了保证上述过程而建立的规则、标准或约定。

举例：TCP协议

早期的网络连接是不可靠的，存在着传输信息丢失的问题，且互联网络的不同部分可能有不同的拓扑结构，为数据的准确传输带来了障碍。TCP传输控制协议（Transmission Control Protocol）就是为了在不可靠的互联网上提供可靠的端到端字节流而设计的一个链路层的协议。它能动态地适应互联网的特性，具备面对各种故障时的健壮性。

TCP使用多种算法来起到控制流量、保证数据传输的有序，主要包括超时重传，动态窗口，顺序保障、三次握手、四次挥手等。这里简单介绍一下三次握手：

第1次握手：发送方主机发送SYN消息包，其中seq=0。此消息包表示请求连接。

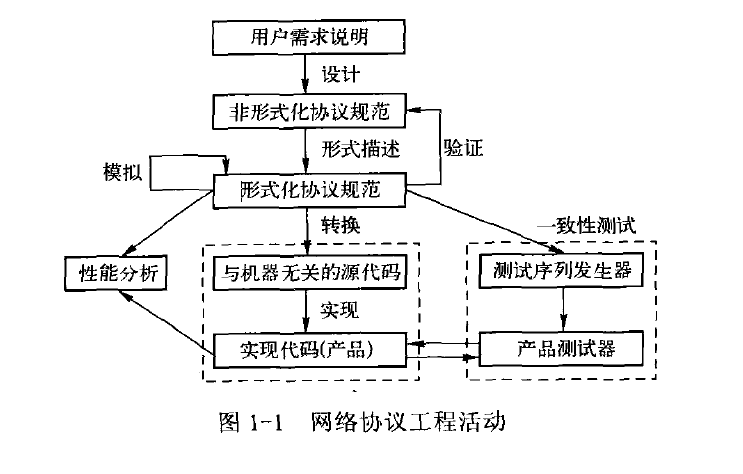
第2次握手：接收方主机发送SYN，ACK响应包，其中seq=0，ack=1（发送方seq+1）。此消息包表示接到了连接请求，确认连接。

第3次握手：发送方发送ACK消息包，其中seq=1（挨着序号0），ack=1（接收方seq+1）.此消息表示接到了响应请求，连接建立。

1. **IPv4协议是一个网络层协议，一个主机（Host）的IPv4协议软件可以将该主机需要发送的数据以IPv4分组的形式发送给不同的接收方主机，在发出的IPv4分组中需要填写该IPv4分组的接收方主机的IPv4地址。问：主机的IPv4协议软件怎样获知到一个IPv4分组的接收方主机的IPv4地址，并将该接收方的IPv4地址填入到此IPv4分组头（IP Header）中？**

发送方主机的IPv4协议软件获得的接受方主机的IPv4地址是由上一层，也就是应用层传输过来的。当应用层发送一个数据时，它会查询DNS缓存或是使用DNS请求（DNS服务起的ip地址是知道的）来获取目标地址的IP地址。当它把数据传入网络层时，也会把该地址传入。这样IPv4协议软件就知道了接受方主机的IP地址，并把它放入IPv4的分组头中。

1. **请说明教学参考书《网络协议工程》图1-1“网络协议工程活动”中的各阶段的工作内容。**



在协议工程中，网络协议的开发过程包括协议的设计、描述、验证、实现、性能分析和测试等几个阶段，

**设计阶段：**

本阶段将需要根据用户的需求来构造出符合要求的对协议的非形式化的规范和描述。它主要包括了协议环境分析(用户需求、协议运行环境的特性、工作模式)、协议的功能设计、协议组织形式的确定、协议元素的构造、协议文本的编制等几项工作。

最后其产出是对协议的非形式化描述文本（用自然语言和图表的形式表述）。将在后续阶段进行形式化描述。

**描述阶段：**

描述阶段接受上一阶段的对协议的非形式化的描述文本，对其使用形式描述技术（例如形式描述语言）进行规范化地表述，以便：为开发者提供一种分析的方法﹔作为对开发结果验证的基础;为设计人员和应用人员提供交流途径;作为开发文档能在将来再开发时使用。

本阶段的产出是应该是用形式描述技术描述的协议，其严密、无二义性，。将在后续阶段转化为可执行的程序。

验证阶段

本阶段主要对协议本身的逻辑正确性进行校验。它处于协议描述之后和协议实现之前，是对已经形式化描述的协议进行检测以纠正协议的错误和缺陷。

其需要对协议进行语法和语义方面进行验证，具体内容主要包括：可达性分析、死锁和活锁检测、协议的有界性和完整性检查、协议的动作序列检查、通道溢出检查等。

在协议验证阶段通过后，协议开发就可以进入实现阶段了。

**实现阶段**

本阶段是将已经通过了验证的协议规范转化到可执行的协议目标代码。在本阶段中，需要自动生成（利用翻译程序）的与机器无关的源代码，或是手动编写与机器有关的实现代码。

该阶段产出是实现规范协议内容的源码程序文件。

**测试阶段**

本阶段对已经实现了的协议程序进行多项测试，来保证实现的协议程序是正确的和符合要求的。

一般会在本阶段进行协议一致性测试，来保证该协议实现与其他同样通过一致性测试的协议实现可在同一个全局系统中很好地协调工作。还会进行对协议实现的评价，核对新实现的性质。

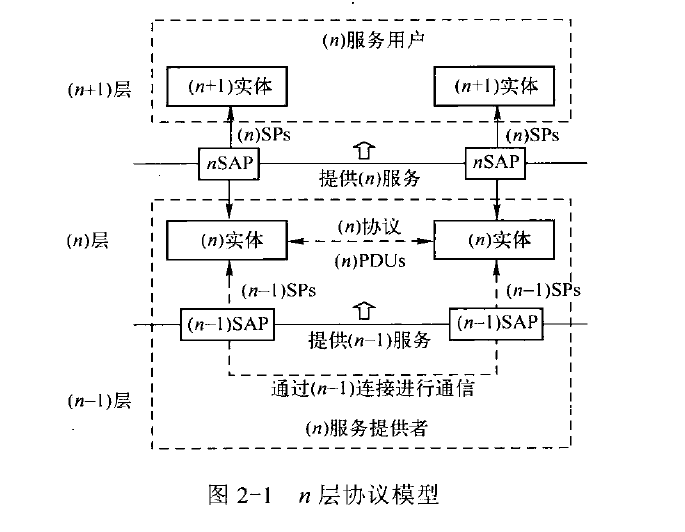
**性能分析阶段**

本阶段是将协议实现进行实质地性能分析以改善协议机制，提高执行效率。其测试的内容主要包括吞吐晕和时延等一些性质。

1. **阅读教学参考书《网络协议工程》2.2节，说明协议的通信环境的含义，并说明设计一个协议时，为什么需要对所设计的协议进行协议环境分析？**

n层中任何两个协议实体通过(n- 1)SAP所形成的数据逻辑通路称为(n-1)层通道，如下图所示。而用户要求、(n-1) 层通道的性质及n 层协议运行时的操作系统和硬件环境构成了n 层协议的通信环境。

通信环境是协议设计要优先考虑的问题。只有充分了解和定义了协议运行的通信环境才能准确地设计出协议的其他元素来。也只有这样，设计出来的协议才能在功能上和性能上满足用户的需求。



1. **分析ABP（Alternating Bit Protocol）协议的环境，包括ABP协议提供的服务、ABP协议底层通道特性、ABP协议机制、ABP协议过程。**

**提供的服务：**

ABP为一个数据链路层协议，是一种停-等的单向传输机制，它提供了带确认的无连接数据传输服务。

**底层通道特性：**

ABP协议不是完全可靠的，会存在丢包的风险。同时，由于它有确认的过程，所以其能保证交给上层的数据包能够按原发送顺序。

**ABP协议机制：**

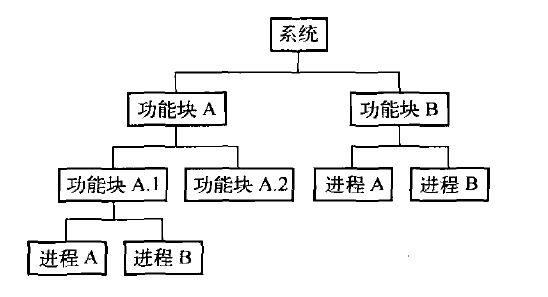
ABP使用ACK（用来表示确认收到消息）和一个附加位（用来标示已发送消息序列中的消息）来提供消息的发送确认。每个从发送方A到接收方B的消息包含一个数据部分和一个字节序列号码,值是0或1。而B有两个确认码:ACK0，ACK1。只有当受到B符合预期的响应后，A才会发送下一条消息，在这之前处于等待状态。

**ABP的协议过程为：**

开始时，消息从发送方发往接收方，并进入等待状态（设定定时器），在发送的消息里除了发送的数据外还有一个1位的序列号（第一次发送的D0置为0）。当接收方接收到消息后，它会将该位的序号反转来告知发送方下一条应该发送的消息（比如受到0，则回复1，表示下一次发送为D1）。发送方在受到响应后回复相应的消息。

1. **SDL语言中的系统、功能块、进程之间的关系？**

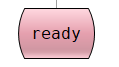
在SDL 中，将规范(specification) 所描述的对象称为一个系统，系统内划分成了各个功能块，功能块的主要目的是描述系统结构，利用功能块，可以分层描述一个大的系统，不同层次对应着系统的不同抽象级别。系统中的每一个功能块由一组进程或一组子功能块组成。系统和功能块之间，以及功能块与功能块之间是由通道相连接的。三者的关系如下图所示：



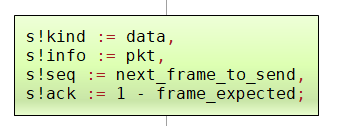
系统的动态行为是通过进程来体现的。

1. **SDL语言中的进程状态机图有哪些组成部分？**

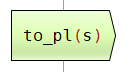
状态图框



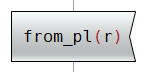
任务块框



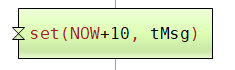
消息输出框



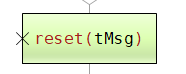
消息输入框



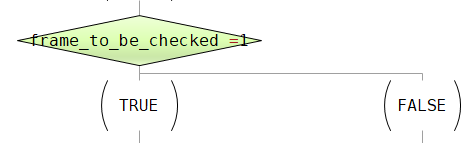
定时器设置框



定时器取消框



判断框

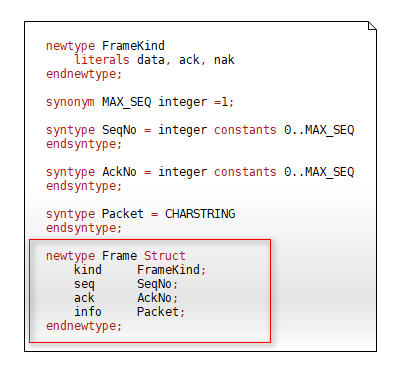


连接点



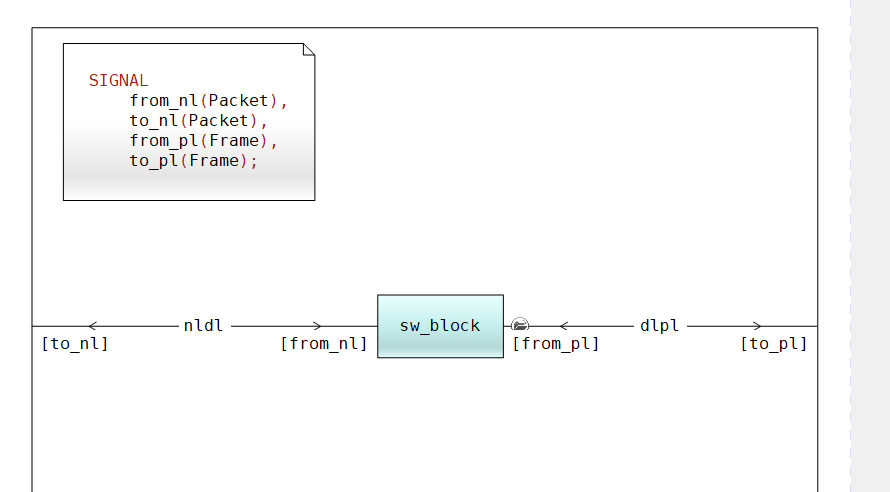
1. **下载、安装Pragmadev Studio软件，阅读该软件的Tutorial文档，了解Pragmadev软件的功能，学习Pragmadev的使用。对所提供的Pragmadev工程例子（Stop-Wait-Protocol），说明在该示例工程中，用SDL语言定义的帧结构中有哪几个字段？该示例是怎样描述了该协议实体与上层协议实体的接口？是怎样描述了该协议实体与下一层协议实体的接口？请描述该示例中，SDL进程“sw\_process”在状态“ready”下接收到信号“from\_pl”后的状态迁移过程。**

帧定义图如下：

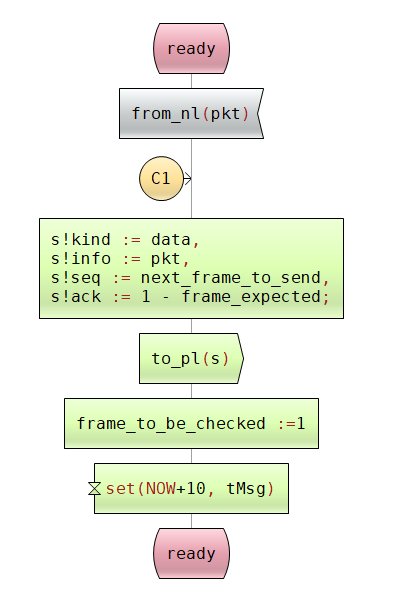
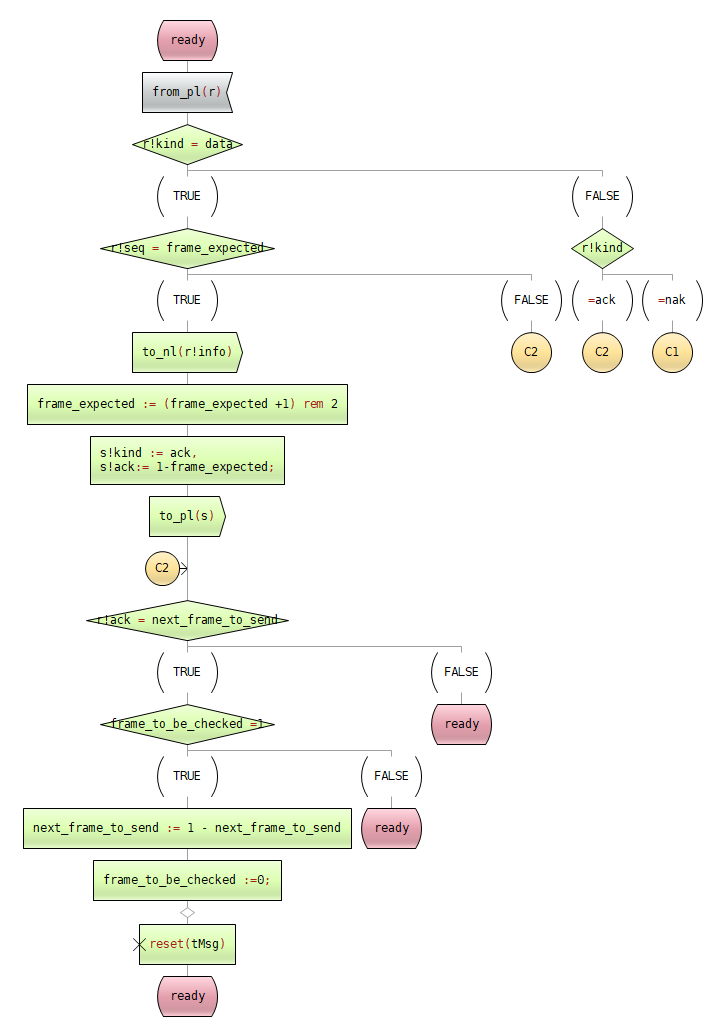


图中Frame为定义的数据帧。其中

* 字段kind是FrameKind类型（自定义的literals类型变量，类似于C语言中的枚举），代表数据帧的类型（数据data，响应ack，错误应答nak）；
* 字段seq是SeqNo类型（限制了范围的整型），代表数据帧的序列号；
* 字段ack是AckNo类型（限制了范围的整型），代表应答的序列号；
* 字段info是Packet类型（字符串）代表帧数据。



图中sw\_block代表了协议实体，其与上层协议实体（网络层的）之间通过信道nldl连接：接受from\_nl(Packet)信号，发送to\_nl(Packet)信号。其与下层协议实体（物理层的）之间通过信道dlpl连接：接受from\_pl(Frame)信号，发送to\_pl(Frame)信号。



在ready状态时，接受from\_pl(r)信号（r是携带的Frame类型数据）后：

* 首先会对消息的类型进行判断。
  + 如果不为data类型则进一步判断类型，
    - 是ack类型（此时处理程序为发送方）就通过C2连接点跳转到图中的C2处，
      * 判断消息的ack与接下来需要发送的消息帧序号是否相等，
        + 如果不相等视为接收方的回应消息错误，忽视之并继续等待。
        + 如果消息的ack与接下来需要发送的消息帧序号相等，则对错误性进行检查（这里简单使用了一个值进行判等表示），

如果不为1视为接收方的消息存在错误，忽视之并继续等待其重发。

如果为1，则将需要发送的消息帧序号变反，用于判断的检查值清零，关闭定时器，进入ready状态等待网络层发过来新的数据包。

* + - 如果数据包类型是nak（此时处理程序为发送方）
      * 说明上一次发送的包接收方接到了，但存在问题，则通过C1连接点跳转到图中的C1处，进行重发操作
  + 如果消息类型为data（此时处理程序为接收方），说明接受的是发送方发来的数据包。则判断是否为期望的数据包
    - 如果不是（这种情况下只可能是因为上一次发送的响应消息发送方没有收到。然后发送方超时重传，此时接收方接收的不符合预期）。则应该重新发送（这一段示例有些问题）。
    - 如果是，则取出数据发送给网络层，然后构造响应包并发送，重置信息（上述C2的步骤）。