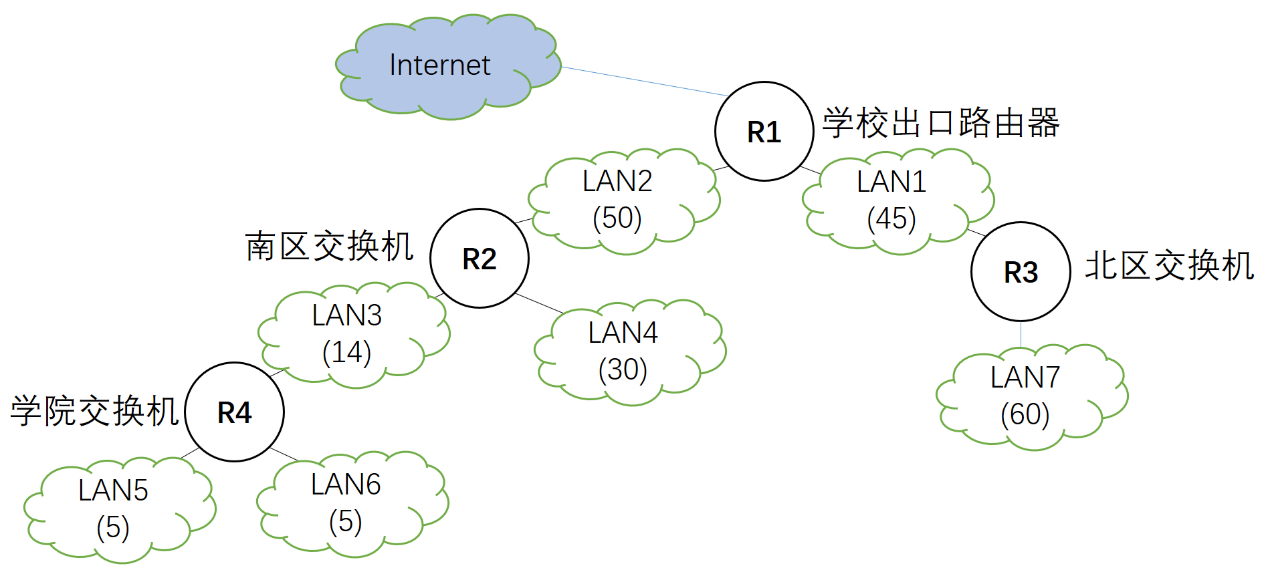
**作业1：子网划分**



首先我们要划分的网段是192.168.1/24，是C类网络划分出来的私网部分，可以支持254台主机。

根据各个网络的主机数量，我们从大到小分配子网，原则是分配的子网是能容纳下所有主机的最小子网。首先我们寻找主机数量在63-126之间的，发现没有。

寻找31-62之间的，发现有三个网络，分别是LAN7，LAN2，LAN1，那么我们直接将192.168.1/24分成四个子网，也就是掩码位数为26，都可以最大容纳2^6 – 2=62台主机。网络地址分别是：192.168.1.0/26（LAN7），192.168.1.64/26（LAN2），192.168.1.128/26（LAN1），192.168.1.192/26。

这样我们还剩下子网192.168.1.192/26，寻找主机数在15-30之间的，只有LAN4，那么我们将192.168.1.192/26再划分为两个掩码位数为27的子网，都可以最大容纳2^5 – 2=30台主机：192.168.1.192/27（LAN4），192.168.1.224/27。

这样我们还剩下192.168.1.224/27可以分配，寻找主机数在7-14之间的，只有LAN3，那么我们将192.168.1.224/27再划分为两个掩码位数为28的子网，都可以最大容纳2^4 – 2=14台主机：192.168.1.224/28（LAN3），192.168.1.240/28。

这样我们还剩下192.168.1.240/28可以分配，LAN5和LAN6都是只有5个主机，所以我们可以将192.168.1.240/28再划分为两个掩码位数为29的子网，都可以最大容纳2^3 – 2=6台主机：192.168.1.240/29（LAN5），192.168.1.248/29（LAN6）。

总结一下刚才我们的划分结果：

LAN7： IP范围：192.168.1.1/26 - 192.168.1.62/26

网络地址：192.168.1.0/26

广播地址：192.168.1.63/26

LAN2： IP范围：192.168.1.65/26 - 192.168.1.126/26

网络地址：192.168.1.64/26

广播地址：192.168.1.127/26

LAN1： IP范围：192.168.1.129/26 - 192.168.1.190/26

网络地址：192.168.1.128/26

广播地址：192.168.1.191/26

LAN4： IP范围：192.168.1.193/27 - 192.168.1.222/27

网络地址：192.168.1.192/27

广播地址：192.168.1.223/27

LAN3： IP范围：192.168.1.225/28 - 192.168.1.238/28

网络地址：192.168.1.224/28

广播地址：192.168.1.239/28

LAN5： IP范围：192.168.1.241/29 - 192.168.1.246/29

网络地址：192.168.1.240/29

广播地址：192.168.1.247/29

LAN6： IP范围：192.168.1.249/29 - 192.168.1.254/29

网络地址：192.168.1.248/29

广播地址：192.168.1.255/29

**作业2：**

我们以TCP为例，总结一下其对可靠数据传输中差错控制、流量控制、拥塞控制的实现原理。TCP为了弥补IP协议尽力而为服务的不可靠性，使用了可靠数据传输技术。

其中的差错恢复机制结合了GBN（go back n）和SR（selective repeat）思想。主要使用的技术我们可以分为超时重传、快速重传。

我们要明确TCP在什么时候会触发这些重传机制：发送端的数据到达接收端之后，接收端和发送一个 ACK 应答信息，表示已经收到了数据包。但是如果发生了数据丢失，比如发送端的数据丢失了，接收端根本没接收到，或者接收端接收到了，应答信息在中间丢了，都会触发重传机制来确保数据发送接收成功。

首先我们总结一下超时重传：

顾名思义，超时重传就是通过使用一个重传定时器设定超时时间（TimeoutInterval/RTO），当发送数据之后，这个时间内没有接收到应答信息，发送端就认为数据丢失了，就会重新发送丢失的数据包。但是问题在于，如何去设置超时时间，超时时间设置长了可能包都丢了有一会儿了才发现丢包了，如果设置过短，就会出现数据还没丢，发送端就判断数据丢失了触发了重传，这样就浪费了资源。

我们的TCP超时时间使用EstimatedRTT和DevRTT计算得出，保证这个超过时间能够根据网络环境的变化而变化。EstimatedRTT是由每一次接收到ACK计算出的SampleRTT和旧的EstimatedRTT加权相加计算得出，DevRTT就是RTT偏差，用每次的SampleRTT与EstimatedRTT的差值与旧的DevRTT加权相加计算得出。最后TimeoutInterval = EstimatedRTT + 4\*DevRTT，给出了一个偏差范围（我这里描述思想，个人认为比公式要关键）。

另外，大多数TCP实现中还使用了超时间隔加倍的方法，避免较差的网络环境中频繁的超时重传。首先发送方发送数据包并启动定时器。如果定时器超时，发送方进行超时重传，并将 RTO 加倍。如果新的 RTO 被超过，发送方再次进行超时重传，并再次将 RTO 加倍。重复上述过程，这样，RTO在每次重传后都会呈指数型增长，直到接收到 ACK 或者达到一个最大重传次数，RTO才开始由我们上文描述的公式决定。

接下来总结一下快速重传技术：它不等待指定的时间，而是根据接收到发送的 ACK 应答信息来判断重传时机。接收方会注意冗余ACK：如果发送端连续收到三个相同的 ACK（表明接收端接收到后续的数据包，但是一个中间的数据包丢失），它会立即重传丢失的数据包，而不必等待定时器超时。快速重传弥补了超时重传机制中可能超时时间过长的问题。

我们刚才说TCP是回退 N 步（Go-Back-N）和选择重传（Selective Repeat）机制的结合体，通过刚才的两种技术我们看到了GBN的特性：只使用一个定时器和累计确认，一个Ack代表这个ACK号之前的包全部都收到了。那么SR体现在哪里？TCP在返回的 ACK 报文中加上 SACK ，作用是让接收端接收到数据包之后返回一个明确收到某一个数据包的应答报文。这样，接收方可以缓存乱序到达的数据包，并通过 SACK 通知发送方，这样发送方就知道哪些是不需要重传的，这使得数据传输更加高效。

流量控制：（flow-control）

为什么需要流量控制？TCP连接中的每一个主机都有接收缓存，TCP接收到的数据存入缓存中，应用层的程序会根据实际情况来读取。但是如果应用程序读取数据的速度相对较慢，而发送方发送的太多、太快，会导致缓冲区溢出。

TCP使用接收窗口（receive window）进行流量控制，以确保发送方不会发送超过接收方可以处理的数据量。接收窗口的大小动态变化，以反映接收方的可用缓存空间。具体来说，接收窗口大小（rwnd）是根据接收方的缓存状态实时调整的。一下定义一些变量来表示流量控制的方法

RcvBuffer：主机B为该连接分配的接收缓存的总大小。

LastByteRead：主机B上的应用进程从缓存中读取的数据流的最后一个字节的编号。

LastByteRcvd：从网络中到达并且放入主机B接收缓存中的数据流的最后一个字节的编号。LastByteRcvd - LastByteRead表示接收缓存中当前已接收但尚未被应用程序读取的字节数。

RcvBuffer -（LastByteRcvd – LastByteRead）表示接收缓存中剩余的可用空间，这就是接收窗口的大小。

随着数据的接收和应用程序读取数据，这些值会动态变化，从而调整接收窗口的大小。接收窗口告诉发送方还有多少空间可以用于存储接收的数据，确保发送方不会发送超过接收方能够处理的数据量。接收窗口大小是通过TCP头中的窗口字段（win）通告给发送方的。发送方根据这个值调整发送数据的速率，以实现有效的流量控制。

拥塞控制：

在TCP中，丢包一般是当网络变得拥挤是由于路由器缓存溢出引起的。分组重传因此作为网络拥塞的征兆。TCP采用的方式是让每一个发送方根据所感知到的网络拥塞程度来限制其能像连接发送流量的速率。如果TCP感知路径上没有拥塞，则会增加发送速率。如果感知到路径上有拥塞，则会减少发送速率。

具体来说，TCP的拥塞控制算法包括3个主要部分：慢启动，拥塞避免，快速恢复。

慢启动和拥塞避免是TCP的强制部分，两者的差异在于对收到的ACK做出反应时增加cwnd长度的方式：一个是指数增长，一个是线性增长。

慢启动：

每经过一个RTT（收到一个ACK），cwnd（拥塞窗口，收到ACK前能发送的字节数）变为之前的两倍。发送方开始时发送initcwnd个报文段。TCP发送速率起始慢，但在慢启动阶段以指数增长。有三种情况结束（或者重启）慢启动：

1. 如果存在一个由超时指示的丢包事件，TCP发送方将cwnd设置为1并**重新开始慢启动**过程。同时还会将第二个状态变量的值ssthresh（慢启动阈值）设置为cwnd/2，即当检测到拥塞时将ssthresh置为拥塞窗口值的一半
2. 当检测到拥塞时ssthresh（慢启动阈值）设置为cwnd的值一半，当到达或超过ssthresh的值时，继续使用cwnd翻番可能会造成拥塞。因此当cwnd的值为ssthresh时，结束慢启动并且TCP**转移到拥塞避免模式**。
3. 如果检测到3个冗余ACK，这时TCP执行快速重传，并**进入快速恢复状态**。

拥塞避免

cwnd的值大约是上次遇到拥塞的值的一半，距离再次拥塞的阈值并不遥远。所以TCP不能采用翻倍策略。因此TCP在**每个RTT**只将cwnd的值增加一个MSS。

例如：如果MSS是1460字节并且cwnd是14600字节，则在一个RTT内发送十个报文段。每个到达ACK增加1/10MSS的拥塞窗口长度，因此在收到对所有10个报文段的确认后，拥塞窗口的值将增加一个MSS。

何时结束拥塞避免呢？**当发生拥塞的时候（超时或者收到重复ack），此时ssthresh需要更新为cwnd的一半，**但是不小于两个MSS。然后：

1. 如果是超时引起的拥塞，则再加一个动作：cwnd被置为initcwnd，开始慢启动。
2. 如果检测到冗余ACK现象，进入快速恢复。

快速恢复：

进入快速恢复状态的同时，TCP也进行了快速重传。

首先知道ssthres已经被被设置为当前cwnd的一半。然后cwn被设置为ssthresh的值加上3个MSS（因为已知收到了3个冗余ACK）。每收到一个额外的冗余ACK，cwnd增加一个MSS。这段时间其实就是一直在等待重传segment的ACK。

当丢失报文段的ACK（TCP在进入快速恢复时进行了快速重传）到达时，TCP将cwnd设置为ssthresh，并进入拥塞避免状态。

当一直没等到丢失报文段的ACK，也就是超时了，ssthresh设为cwnd的一半，cwnd重置，进入慢启动。

**作业3：**

首先谈一谈网络工程管理的一些核心概念：

网络工程管理是指在设计、部署、运营和维护网络系统时，应用各种管理方法和技术，以确保网络系统的高效运行、可靠性和安全性。其核心概念包括规划与设计、部署与实施、监控与维护、安全管理和资源管理。

在规划与设计阶段，网络工程师需要进行详细的需求分析，了解用户和业务的需求，设计网络拓扑结构，选择适合的网络技术和设备，并制定IP地址分配方案。部署与实施阶段是将设计好的网络架构付诸实践，确保网络能够按计划运行。配置网络设备，进行物理线路的铺设和连接，完成系统集成和功能测试。监控与维护阶段确保网络系统的持续高效运行。使用网络监控工具实时监控网络设备和链路的运行状态，及时处理网络故障，进行性能优化，生成网络运行报告，评估网络性能和稳定性，提出改进建议。安全管理阶段确保网络系统免受未经授权的访问和攻击，保护数据的完整性和隐私性。比如实施访问控制和身份验证，使用加密技术保护数据传输和存储的安全性。资源管理阶段有效利用网络资源，优化网络性能和成本。

网络分层模型，不管是OSI七层模型还是我们学的因特网五层模型，都是为了实现各个层次功能的分离，就像是我们软件理论中的降低耦合度的操作一样，就是为了便于维护和拓展。我们网络模型中的每个层次都有自己的功能，将来自上一层的数据封装上自己的操作，这样一层层传递，最后从物理层发出的时候已经实现了所有功能。相邻的层次之间都有特定的接口，比如我们应用层和运输层之间有套接字接口，TCP套接字和UDP套接字在多路分解的时候还有所不同，一个有完整的源和目的ip以及端口，一个只有目的ip和端口（扯远了）。每个层次都工作着多种协议，比如应用层有HTTP、SMTP等，运输层有TCP、UDP等，网络层更不用说，除了数据平面的IP协议，还有控制平面中iBGP和eBGP可用的多种路由协议，这些协议适用于不同的需求，有不同的特性，并无好坏之分。

我们站在上帝视角，总览一遍一个网络服务建立的流程，就拿最常用的邮件传输服务举例：

问题定义：

在不同网络环境中实现可靠的电子邮件传输和管理，需要解决邮件的发送、接收和存储问题，同时确保邮件传输的可靠性、安全性和高效性。

需求分析：

用户需要能够通过各种邮件客户端包括网页发送和接收电子邮件。确保邮件在传输过程中不会丢失或损坏。防止邮件被未授权的第三方截获或篡改。系统需要支持多种邮件客户端和服务器，实现不同设备和平台之间的互操作性。

可行性分析：

采用分层模型设计电子邮件系统，可以简化系统设计和维护。分层模型将各个网络功能模块化，使得每层可以独立开发和优化，同时利用现有的标准化协议（如SMTP、IMAP、POP3）进行邮件传输和存储，提高系统的可扩展性和兼容性。

技术方案：

我们就选择SMTP（Simple Mail Transfer Protocol）作为我们的应用层协议，由于我们的邮件传输最重要的是需要可靠性和安全性，我们可以使用TCP协议来提供可靠的、面向连接的传输服务，确保邮件数据包在传输过程中不会丢失、重复或失序。TCP通过三次握手建立连接，并使用确认机制和重传机制保障数据传输的完整性和正确性，还有流量控制和拥塞控制来减少重传的发生，提高效率。网络层我们首先当然要有IP协议，负责标记IP地址。然后我们还应该使用完整的路由协议来引导分组的转发，比如说自治域内的OSPF协议，域间的BGP协议，实现基于不同的算法。在数据链路层，就负责数据帧的传输、错误检测和纠正（CRC算法）。常用的技术包括以太网（Ethernet）、Wi-Fi等，确保数据在局域网（LAN）内的可靠传输。至于物理层，就是定义了各种传输设备或者介质的规范，我们这里不多阐述。

我们以自顶向下的方式将整个互联网的各个层次（浅薄地）学习了一遍，大体上也知道了我们的数据是如何指哪打哪，准确的发送到我们想要的位置。最关键的是，这些技术对于用户来说是基本不可见的，甚至各个层次之间的实现也是相互不可见的，这一点难能可贵，是网络发展了几十年的结果。我们从网络工程管理的角度，分析了一个网络服务系统的实现过程从设计者的角度来宏观地看待五个网络层次分别发挥着什么样的作用，其实是对学习很有帮助的。