计算机网络 第三章 读书笔记

3.1 概述和运输层服务

- 运输层协议为运行在不同主机上的应用进程之间提供了逻辑通信功能。从应用程序的角度看，通过逻辑通信，运行不同进程的主机好像直接相连一样。实际上，这些主机也许位于地球的两侧，通过很多路由器及多种不同类型的链路相连。应用进程使用运输层提供的逻辑通信功能彼此发送报文，而无需考虑承载这些报文的物理基础设施的细节。

- 运输层协议是在端系统中而不是在路由器中实现的。在发送端，运输层将从发送应用程序进程接收到的报文转换成运输层分组，用因特网术语来讲该分组称为运输层报文段。实现的方法是将应用报文划分为较小的块，并为每块加上一个运输层首部以生成运输层报文段。然后，在发送端系统中，运输层将这些报文段传递给网络层，网路层将其封装成网络层分组（即数据报）并向目的地发送。注意到下列事实是重要的：网络路由器仅作用于该数据报的网络层字段；即它们不检查封装在该数据报的运输层报文段的字段。在接收端，网络层从数据报中提取运输层报文段，并将该报文段向上交给运输层。运输层则处理接收到的报文段，使该报文段中的数据为接收应用进程使用。

- 网络应用程序可以使用多种的运输层协议。例如，因特网有两种协议，即 TCP 和 UDP。每种协议都能为调用的应用程序提供一组不同的运输层服务。

3.2 多路复用与多路分解

- 主机怎样将一个到达的运输层报文段定向到适当的套接字（socket）？为此目的，每个运输层报文段中具有几个字段。在接收端，运输层检查这些字段，标识出接收套接字，进而将报文段定向到该套接字。将运输层报文段中的数据交付到正确的套接字的工作称为多路分解（demuliplexing）。在源主机从不同套接字中收集数据块，并为每个数据块封装上首部信息（这将在以后用于分解）从而生成报文段，然后将报文段传递到网络层，所有这些工作称为多路复用（muliplexing）。

3.3 无连接运输： UDP

- UDP 检验和提供了差错检测功能。这就是说，检验和用于确定当 UDP 报文段从源到达目的地移动时，其中的比特是否发生了改变（例如，由于链路中的噪声干扰或者存储在路由器中时引人问题）。发送方的 UDP 对报文段中的所有 16 比特字的和进行反码运算，求和时遇到的任何溢出都被回卷。得到的结果被放在 UDP 报文段中的检验和字段。

- 因为假定 IP 是可以运行在任何第二层协议之上的，运输层提供差错检测作为一种保险措施是非常有用的。虽然 UDP 提供差错检测，但它对差错恢复无能为力。UDP 的某种实现只是丢弃受损的报文段；其他实现是将受损的报文段交给应用程序并给出警告。

3.4 可靠数据传输原理

- TCP 是在不可靠的（IP）端到端网络层之上实现可靠数据传输协议。

- Rdt3.0 是一个功能正确的协议，但并非人人都对它的性能满意，特别是在今天的高速网络中更是如此。rdt 3. 0 性能问题的核心在于它是一个停等协议。

- 解决这种特殊的性能问题的一个简单方法是：不使用停等方式运行，允许发送方发送多个分组而无需等待确认，如果发送方可以在等待确认之前发送 3 个报文，其利用率也基本上提高 3 倍。因为许多从发送方向接收方输送的分组可以被看成是填充到一条流水线中，故这种技术被称为流水线（pipelining）。

- 所需序号范围和对缓冲的要求取决于数据传输协议如何处理丢失、损坏及延时过大的分组。解决流水线的差错恢复有两种基本方法是：回退 N 步（Go- Back-N, GBN）和选择重传（Selective Repeat, SR）。

- 在回退 N 步（GBN）协议中，允许发送方发送多个分组（当有多个分组可用时）而不需等待确认，但它也受限于在流水线中未确认的分组数不能超过某个最大允许数 N。

- 那些已被发送但还未被确认的分组的许可序号范围可以被看成是一个在序号范围内长度为 N 的窗口。随着协议的运行，该窗口在序号空间向前滑动。因此，N 常被称为窗口长度（window size），GBN 协议也常被称为滑动窗口协议（sliding- window protocol）。我们为什么先要限制这些被发送的、未被确认的分组的数目为 N 呢？为什么不允许这些分组为无限制的数目呢？流量控制是对发送方施加限制的原因之一。这是 TCP 拥塞控制时分析另一个原因。

- 选择重传（SR）协议通过让发送方仅重传那些它怀疑在接收方出错（即丢失或受损）的分组而避免了不必要的重传。这种个别的、按需的重传要求接收方逐个地确认正确接收的分组。再次用窗口长度 N 来限制流水线中未完成、未被确认的分组数。然而，与 GBN 不同的是，发送方已经收到了对窗口中某些分组的 ACK。图 3-23 显示了 SR 发送方看到的序号空间。图 3-24 详细描述了 SR 发送方所采取的动作。

- 检验和：用于检测在一个传输分组中的比特错误

- 定时器：用于超时/重传一个分组，可能因为该分组(或其 ACK)在信道中丢失了。由于当一个分组延时但未丢失（过早超时），或当一个分组已被接收方收到但从接收方到发送方的 ACK 丢失时，可能产生超时事件，所以接收方可能会收到一个分组的多个冗余副本

- 序号：用于为从发送方流向接收方的数据分组按顺序编号。所接收分组的序号间的空隙可使接收方检测出丢失的分组。具有相同序号的分组可使接收方检测出一个分组的冗余副本

- 确认：接收方用于告诉发送方一个分组或一组分组已被正确地接收到了。确认报文通常携带着被确认的分组或多个分组的序号。确认可以是逐个的或累积的，这取决于协议

- 否定确认：接收方用于告诉发送方某个分组未被正确地接收。否定确认报文通常携带着未被正确接收的分组的序号

- 窗口、流水线：发送方也许被限制仅发送那些序号落在一个指定范围内的分组。通过允许一次发送多个分组但未被确认，发送方的利用率可在停等操作模式的基础上得到增加。我们很快将会看到，窗口长度可根据接收方接收和缓存报文的能力、网络中的拥塞程度或两者情况来进行设置

3.5 面向连接的运输： TCP

- TCP 被称为是面向连接的（connection- oriented），这是因为在一个应用进程可以开始向另一个应用进程发送数据之前，这两个进程必须先相互“握手”，即它们必须相互发送某些预备报文段，以建立确保数据传输的参数。作为 TCP 连接建立的一部分，连接的双方都将初始化与 TCP 连接相关的许多 TCP 状态变量

- TCP 连接提供的是全双工服务（full-duplex service)

- TCP 在 IP 不可靠的尽力而为服务之上创建了一种可靠数据传输服务（reliable data transfer service）。TCP 的可靠数据传输服务确保一个进程从其接收缓存中读出的数据流是无损坏、无间隔、非冗余和按序的数据流；即该字节流与连接的另一方端系统发送出的字节流是完全相同。

- TCP 通过重传引起超时的报文段来响应超时事件。然后 TCP 重启定时器。

- TCP 发送方必须处理的第三个主要事件是一个来自接收方的确认报文段（ACK）的到达（更确切地说是一个包含了有效 ACK 字段值的报文段）。当该事件发生时，TCP 将 ACK 的值 y 与它的变量 SendBase 进行比较。

- 如果当前有未被确认的报文段，TCP 还要重新启动定时器。

- 当该 TCP 连接收到正确、按序的字节后，它就将数据放人接收缓存。相关联的应用进程会从该缓存中读取数据，但不必是数据刚一到达就立即读取。事实上，接收方应用也许正忙于其他任务，甚至要过很长时间后才去读取该数据。如果某应用程序读取数据时相对缓慢，而发送方发送得太多、太快，发送的数据就会很容易地使该连接的接收缓存溢出。

- TCP 为它的应用程序提供了流量控制服务（flow- control service）以消除发送方使接收方缓存溢出的可能性。流量控制因此是一个速度匹配服务，即发送方的发送速率与接

收方应用程序的读取速率相匹配。前面提到过，TCP 发送方也可能因为 IP 网络的拥塞而被遏制；这种形式的发送方的控制被称为拥塞控制（congestion control）

- TCP 通过让发送方维护一个称为接收窗口（receive window）的变量来提供流量控制。

3.6 拥塞控制原理

- 分组重传因此作为网络拥塞的征兆（某个特定的运输层报文段的丢失）来对待，但是却无法处理导致网络拥塞的原因，因为有太多的源想以过高的速率发送数据。为了处理网络拥塞原因，需要一些机制以在面临网络拥塞时遏制发送方。

- 当一个分组沿一条路径被丢弃时，每个上游路由器用于转发该分组到丢弃该分组而使用的传输容量最终被浪费掉了

- 在最为宽泛的级别上，我们可根据网络层是否为运输层拥塞控制提供了显式帮助，来区分拥塞控制方法。

- 端到端拥塞控制：在端到端拥塞控制方法中，网络层没有为运输层拥塞控制提供显式支持。即使网络中存在拥塞，端系统也必须通过对网络行为的观察（如分组丢失与时延）来推断之。

- 网络辅助的拥塞控制：在网络辅助的拥塞控制中，网络层构件（即路由器）向发送方提供关于网络中拥塞状态的显式反馈信息。这种反馈可以简单地用一个比特来指示链路中的拥塞情况。

- ATMABR 拥塞控制是种基于速率的方法。即发送方明确地计算出它所能发送的最大速率，并据此对自己进行相应的调整。

3.7 TCP 拥塞控制

- TCP 所采用的方法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向连接发送流量的速率。如果一个 TCP 发送方感知从它到目的地之间的路径上没什么拥塞，则 TCP 发送方增加其发送速率；如果发送方感知沿着该路径有拥塞，则发送方就会降低其发送速率。

- TCP 发送方是如何限制向其连接发送流量？TCP 连接的每一端都是由一个接收缓存、一个发送缓存和几个变量（LastByteRead、rwnd 等）组成。运行在发送方的 TCP 拥塞控制机制跟踪一个额外的变量，即拥塞窗口（congestion window）。拥塞窗口表示为 cwnd，它对一个 TCP 发送方能向网络中发送流量的速率进行了限制。限制了发送方中未被确认的数据量，因此间接地限制了发送方的发送速率。

- TCP 拥塞控制算法（TCP con-gestion control algorithm）：该算法包括 3 个主要部分：1、慢启动；2、拥塞避免；3、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 ewnd 长度的方式。快速恢复是推荐部分，对 TCP 发送方并非是必需的。