摘要

TCP/IP，Internet 世界中必不可少的协议包，是由美国国防先进技术研究计划署（DARPA）于 15 年前开发实现的，已经被广泛地应用于国防和经济领域。如今我们可以轻易地找到大量研究这个协议如何工作的论文，但是我们很难从这些论文中弄清楚这个协议为什么一步步地变成了现在的样子。比如说，Internet 协议是为了能够提供无连接或报文模式服务而设计的，但是其中详细的设计动机已经被人们大大的误解了。这片论文就是尝试着发掘这些最终决定了 Internet 协议结构的最初的一些想法。

1.简介

过去的 15 年里，美国国防先进技术研究计划署开发出了一系列用于包交换网络的协议，其中就有 IP 和 TCP。而这两个协议现已成为美国国防部内网的标准，同时也已在经济领域广泛应用。不仅如此，它们还影响了其它相关协议的设计，最重要的就是 ISO 的无连接架构协议。

尽管国防部这个协议用得挺好，但是现在很难搞明白当初是何因素促成了这种良好的设计。

事实上，从第一个协议诞生，一直发展到现在的标准，这整个过程中处处散发着设计哲学的气息，时时都晃动着设计哲学的影子。比如说，第一篇论文里并没有特别强调报文和无连接服务，但是它们逐渐成为了这个协议的关键特征。还有一个例子，我们现在感觉在 IP 层和 TCP 层之间加上层次结构是最基本的设计常识了，但是你相信吗，最初的设计并没有考虑到这些。在反复的实践和测试中，这些更“合理”的设计才逐渐被加入到协议中。

Internet 的体系结构还在发展中。一些新的发展思路可能会和现有的设计规范相冲突，但是我们应该认识到，对以往的 Internet 设计哲学的良好理解总是可以给现在的工作提供有意义的借鉴的。ISO 的无连接协议里面也是经常可以看到 Internet 设计哲学的影子，所以好好了解一下这种设计哲学对那些在 ISO 组织的工作人员来说应该是有帮助的。

这篇论文罗列了 Internet 体系结构设计的最初的一些目标，然后讨论了这些目标和协议中重要特征之间的关系。

2.目标之基础层次

DARPA-Internet 体系结构设计的高级目标就是想开发出一种实用的技术，使现存的多种网络环境之间能够自由沟通。下面一些具体的例子可以让大家更了解这个目标。

Internet 的组件是各种各样的网络，而这些不同的网络可以相互联通以便提供更多高级的服

务。最初的目标就是想把当初的 ARPANET 和 ARPA 电台网连接起来，让后者的用户可以享受前者提供的服务。当时设计人员就已经假定会有各种各样其它的网络共存，虽然那时局域网都还没有出现。

为了实现网络互联的目的，我们可以设计一个所有网络的综合体，也就是巨大而复杂的“多媒体”网络——融合度越高，越好使。那也就是说，要想让这个 Internet 真正地好用，我们必须将所有现存的网络系统全部融合在一起。更可怕的是，网络本身还存在着管理域的问题。所以说，要实现这样一种方式的互联似乎有些过于雄心壮志了——野心太大了，不现实。

现实中我们选择了包交换技术用于适应多元化的网络环境。当然，像电路交换似乎也可以考虑，但是实际上它的应用，比如远程登入，其实也还是依靠报文交换技术实现的，网络互联用的是网络交换，都属于包交换。也就是说，包交换应该成为 Internet 体系结构的一个基础构件。

要实现这个基本的目标，最后要考虑的就是实现互联的技术细节了。而之前 DARPA 的另一个项目——ARPANET，已经实现了一种很好的存储转送包交换技术，那么我们就需要考虑更高级的细节问题：实现一种基于 Internet 层次的包交换，而这种层次上的连接我们称之为网关。

好了，从这些细节我们已经可以看到 Internet 的结构雏形了：在一个很大的基于包交换技术的环境中，大量各种各样、乱七八糟的网络通过使用一种称之为网关的包交换处理机实现了互联，在这种网关里面执行的是一种存储转送包交换算法。

3.目标之第二层次

关于更高一层的目标，前面已经提到，那就是“有效”。当然前面并没有提供有效连接的具体定义，下面列出了一组评测 Internet 体系结构够不够有效的目标：

1.Internet 连接可以保持下去，即使网络或者网关出了问题；

2.Internet 必须可以支持大量的网络互联服务；

3.Internet 体系结构必须能容纳各种各样的网络；

4.Internet 体系结构必须可以胜任各种资源的分散式管理；

5.Internet 体系结构必须可以做到投入合理；

6.Internet 体系结构必须可以比较方便地成为主机的一部分；

7.Internet 体系结构里面使用的资源必须是可控、容易被理解的。

这些目标看起来不过是所有特征网络的集合。但是我们一定要明白这些特征排列顺序的重要性。顺序不同，网络可能也就不同了。比如说，对于军用网络，生存性就会被排在第一位，而易解读性则应该放在最后。在战场上，我们更关心的是如何尽快将所能搜集到的所有信息以最可靠的方式传到目的地，而不是关注于种种细节的含义。起初大家并没有太留意易解读性，后来才逐渐重视的。对于为经济应用而设计的体系而言，易解读性应该放在第一位。

同样地，降低成本也是我们明确的目标，但是排在胜任分散式管理和容纳所有网络之后。而在其它协议里，包括一些很出名的商用体系，比如一个基于多媒体快速传输的网络而言，它就需要很昂贵的设备投入，并且与其它网络的互联性能也很差，局域网就是一例。读者可能很仔细地考虑了上面的目标清单，并且发现这并不是一个真正的“母版”清单，而只是一组对 Internet 体系产生了很大影响的优先考虑的设计目标。下面几部分讨论这个清单和实际中 Internet 特征之间的关系。

4.绝境逢生

在 Internet 中最重要的目标就是要持续地提供互联服务，即使网络和网关都出现了错误。具体说，有两个网络正在通过 Internet 互联，一些错误导致 Internet 发生暂时性中断，然后 Internet 可以自己悄无声息地自动重新连接修复错误，继续提供服务，而不用我们人为地去在更高层次进行操作来手工修复 Internet。更准确地说，传输层协议并不总向客户端报告发送方和接受方可能失去同步性这一事实，它总是假定这个同步性是一定可以保障实现的，除了确实无路可走了。对于传输层的最高层而言，只有一个最终的无连接的错误报告，而发生在底层的各种各样的传输错误体系根本不会让我们知道。

为了达到上面的目标，我们必须保存好通信的状态信息。典型的状态信息应该包括传送包数，应答数，还有流量控制信息。协议低层如果丢失了这些信息的话，他们就搞不清楚数据是否丢失了，应用层还得应付同步性缺失的问题。本协议中就不容许丢失这些重要的状态信息，必须要保存好。

有些网络体系是把这些状态信息保存在网络包交换节点里的。在这种情况下，我们就需要把信息复制。考虑到复制的分布式的特性，而保障复制可靠性的算法又很难实现，所以几乎所有这种网络体系都不提供应付失败的机制。另一种方案呢，就是把状态信息收集起来保存在网络终端，当然这些终端本身也使用网络服务。我把这种方式称为“各扫门前雪”（均担）——一个终端实体消失了，那么它自己保存的状态信息也可以丢失，对别的实体、整个 Internet 并无太大影响。传输层同步性信息则保存在主机中，主机连接到整个网络并且使用整个网络上的服务。

“各扫门前雪”模式比“复制”模式有两大优势：第一，前者不怕出错（任何错误不致影响到整个网络），而后者容错性就差多了；第二，前者在技术上也更容易实现。

“各扫门前雪”模式引出了两个推论：第一，无论中间包交换节点还是网关，都不能携带包含持续互联的实质性信息，换句话说，他们是无状态信息的包交换，因此有时候这种网络设

计被称为“报文”网络，他们自己带着寻址信息，并不管中间怎么走；第二，在这个体系中主机的作用更主要，因为它不像有些网络本身就可以保障数据传输安全。如果主机可以搞定数据传输中的各种问题，那么应用层就不需要再去费心这些事了。

尽管连接的持续性是最重要的目标，但是还有一个很高的目标就是互联现存所有的网络。多功能的网络体系设计可能会让网络互联更加健壮。比如说，Internet 不太相信网络会报告自身发生的错误，而是亲历亲为，从 Internet 层次发现控制错误。

5.还要支持各种各样的服务

Internet 体系第二个目标就是要为各种各样的网络服务提供支持。不同服务存在着巨大的差异，在速度、延迟和可靠性方面都有不同的要求。传统的服务类型是面向双向可靠数据传输的，这种服务又被称为“虚电路”服务，一般应用于远程登录和文件传输，这是在 Internet 中用传输控制协议（TCP）提供的最重要的服务。人们很早就认识到这种服务要面对不同的需求，因为远程登录要求延迟不能太长，但是并不苛求带宽；而文件传输则对带宽（数据吞吐量）要求很高，对延迟要求不高。TCP 就要同时提供这两者不同的服务。最初人们感觉 TCP 就是要提高所有类型的服务，但是后来大家弄清楚到底有多少服务类型时，谁都知道，用一个协议来支持那么多的服务是很不行的。

第一个 TCP 之外的实例是用来支持 XNET 的。以下原因使得 TCP 并不适用于 XNET：第一，调试器协议不需要多么可靠。这可能听起来挺古怪。不过我们设想一下：在一个高负荷和充满错误的恶劣条件下我们却要求可靠的连接，这是不太现实的，而也许恰恰这个时候我们需要调试器——很显然，我们这时候就是要想办法建立起连接，可以将数据传出去，而不能再那么死板地要求字节流一定要按顺序传输了。第二，假如 TCP 可以包容大量不同的客户段，那么它必定是相当的复杂，而在调试环境中引入这种复杂度是不明智的，要知道，在调试环境可能连其它系统必需的最基本的服务（比如计时器）都没有。所以 XNET 被设计成一种运行在 Internet 提供报文服务层上端的系统。

还有一种不符合 TCP 胃口的服务是数字语音实时传输（用于命令控制系统中的电话会议方面），它不要求传输多么可靠，而是要求传输得尽量及时和平滑。应用层把语言数字化，打包并像其它数据一样传出去，但是要求接收方可以接收到正常顺序的数据流并可以还原成正确的语音。假如没有收到应该收到的数据，那么就不能实时地再现语音了。而通过调查，人们惊讶地发现传输延迟大部分是因为对传输可靠性要求太高了！典型的可靠传输是这样的，一旦有数据包传输发生错误，它就要求重发，直到正确地接收到这个包才会继续接收其它后续的数据包。这样就会导致传输延迟比正常时多出很多倍，当然也就中断了语音的再现。事实上，偶尔丢失了一个数据包并不要紧，我们可以用空白代替它，这在语音还原时不会造成听者的理解障碍。即使听者没听清（此时可能网络环境不好，接连发生了较多的包丢失），我们此时还有高级纠错措施；再说了，听者也可以让说话者重复一遍嘛。

因此在相当早的时候，人们就认识到有很多传输服务需要支持，体系必须容忍同时发生的传输对可靠性、延迟、带宽有着不同的要求。

这个目标导致了 TCP 和 IP 的产生，从原来的一个层变成现在的两个层。TCP 提高可靠的顺序的数据流传输服务，而 IP 则提供一个基本的隔离区，把各种不同的服务都隔离在 IP 层之下。这种隔离区里面用的是支持持续连接的报文传递。此时报文传递“很高效”，但是可靠性不能得到保障，怎么办呢？有办法！我们可以另外建立一个可靠的服务（通过更高层次上的确认重传机制实现）；还有一种方法，牺牲物理层原始延迟换取可靠性。用户数据报协议（UDP）提供基本的 Internet 数据报文服务。

协议不希望下层网络自己就可以支持多种服务，因为这与使用现存网络的目标相违背（现有网络应该独立于其它同层网络）。实际上，我们希望用主机和网关里的算法通过熟记报文的方式来支持多种服务。比如说（尽管现在我们一般不那么做），我们把延迟可控的、不要求可靠性的数据报文放在传输队列的队首，除非它们的生命周期已经终止（此时就将它们丢弃）；同时把高可靠性数据包放在后面，当然不会把它们丢弃，不管它们在网中已经有多长了。

在没有下层网络支持的情况下实现对多种服务的支持这一设想，比原来设想得难得多。最棘手的问题是，很多为特殊用途设计的简单网络根本支持不了其它的服务。一个最普遍的情况是，很多网络是基本可靠数据传输而设计的，它们会把延迟当作数据可靠性的有机组成部分，不管此时是否要求可靠性。比如像 X.25 这种网络，就是为了可靠传输设计的，想回避它的这一特性？门都没有！所以呢，尽管 Internet 在 X.25 上工作得挺好，但是并不能完成那些有其它要求的服务。像其它有些本身支持数据报服务的网络可能更灵活，但是这种网络比较少见（特别是长途上下文？）。

6.还要搞定各种五花八门的网络

Internet 体系的成功有一个很重要的因素，它合理地集成利用了大量的网络技术，包括军事、经济领域的。Internet 体系在这一方面做得非常成功，它调配着各种各样的网络，包括远程网（ARPANET 和各种各样的 X.25 网）、局域网（以太网、环形网等）、广播卫星网（传输速度为 64 千比特每秒的 DARPA 大西洋卫星网和覆盖全美速度为 3 兆比特每秒的 DARPA 广谱频试验卫星网）、信息包无线电网（DARPA 信息包无线电网、英国试验信息包无线电网以及业余爱好者开发的信息包无线电网）、各种连续信号连接（从每秒 1200 次到 1T 次异步连接）还有各种其它特殊设备（比如其它网络更高层如 IBM's HASP 提供的交通服务）。

Internet 体系假定网络起码要提供的功能组，包括网络可以传输包或报文。数据包大小应该合理，最小在 100 字节左右，还能以一个合理的可靠度来传输数据。对于非点对点连接来说，合理的寻址方式也是很重要的。

当然还有大量的服务是网络本身所不支持的，比如说可靠或有序传输，网络级广播和多点传送，数据包传输排序，多种服务支持以及失败、速度或者延迟等信息内部获取。如果我们想使用这些服务，那么就需要让这个网络融入 Internet 中，这有两种实现方式，要么这个网络自己直接向 Internet 提供这些服务，要么在网络终端应用软件层面上模拟这些功能。这显然不太爽，因为我们要为每一个网络甚至每一个主机重新设计实现这些服务。对这些服务进行移植，拿在 TCP 中进行可靠传输来说，首先必须要重新设计一次，然后在每个主机上执行

一次。之后在一个新网上执行就容易多了。

7.还有其他目标

前面已经讨论过的三个目标对整个体系的设计起着至关重要的作用。其余的目标不是那么重要，所以可能就没有严格地满足或者没有很彻底地设计。满足分布式管理这一目标在某些方面很好地被满足了。比如说，Internet 中有很多网关，它们并不是被同一个机构管理，而不同机构会有不同的路由算法，这时就必须要求它们能够正确地交换路由表，不管他们之间是否相互信任；事实上在各个网关中存在着各种各样的算法。也就是说，网关的管理者可能和该网关连接的网络的管理者根本是两家组织。

另外呢，当今 Internet 有很多重要的难题是跟缺少分布式管理的工具有关的，特别是在路由这块儿。现在 Internet 路由仍受限于资源使用的规定，而人们对此只能采取很有限的措施并且要求人工设置路由表。这是一种错误的倾向况且也不是很有效。最近几年 Internet 体系最重要的进步很可能是对资源管理能力的改善，而这些资源是由成倍增加的各种关系错综复杂的管理者所提供的。

很清楚的一点是，在某些环境下 Internet 体系在控制资源成本方面做得还不尽人意。现在数据报头还是太长（典型的有 40 字节），如果包很小，那代价就会变得很大（假设包信息有 1 个字节，而必须要带一个 40 字节的头部，代价显然是很大的）。事实上，对于任何一种协议来说，提出基于提高效率的传输方案都是非常难的。当然了，在大量文件传输时，数据包有 1000 字节，此时 40 字节的头部就显得微不足道了，仅占总开销的 4%。另一个低效的罪魁祸首就是遗失包的重发。既然 Internet 不能保证遗失包在网络层的有效复原，那么从 Internet 另一端重发丢失的包就很必要了。也就是说，这个数据包要重新路由、重新历经千山万险才有可能成功（仅仅是可能）。这就是一种折中的方案，由终点提供服务，这样做网络层的代码是简单了，但是真个的效率却降低了。当然了，如果重发率足够低（比如低于 1%），那么这么消耗还是可以容忍的。对于融入这个协议的网络而言，有一个粗糙的评判标准：1%的重发率还是可以接受的，10%的话则就需要将提高可靠性考虑加入到网络中了（如果这项服务是必需的话）。

将主机接入 Internet 中的代价可能要比接入其它协议要大很多，因为像应答、重传等服务要在主机上执行，而不是在 Internet 上。让不熟悉协议的程序员做这种工作会非常困难，因为人们试图把用于传输的协议移植到顺序执行的处理机中，并且只需执行一次，不用对不同主机重新来一次。这时候我们就需要这么一种支持前置协议的主机，它执行的效果和原始传输协议的效果是一样的。随着协议本身的发展，大家对此的担心也少了很多，因为无论对于个人电脑还是那些计算资源有限的机器，都可以很好的执行我们的协议。

采取这种“主机驻留协议”方式产生了一个后果：当执行不是那么顺畅时，不仅会伤害到网络，还会伤害到主机。起初的试验时只是用了少量可控的主机应用，所以这个问题还可以容忍；但是随着 Internet 的迅速壮大，这个问题变得日益严重。这么看来，当初是为了追求鲁棒性才引入了“各扫门前雪”的想法，又通过主机驻留的算法实现了这一想法，而现在正是这种算法在主机出现运行异常时会导致鲁棒性的丧失。

最后的一个目标是易解读。事实上，由 Cerf 和 Kahn 执笔的第一篇论文把易解读作为了协议和网关一个非常重要的特征。但是现在 Internet 体系了几乎没有用于保障数据包易解读的工具项。考虑到非军事用户和那么特别关注数据包易读易控性的用户的要求，这个问题现在才被真正地研究。

8.理论的体系与实际的执行

前面讨论过了，Internet 体系就是要提供大量灵活的服务支持。不同的传输协议支持不同的服务，不同的网络可以被融合在一起。从另一个角度看，体系尽量不去限制 Internet 可以提供的服务范围，反过来说，要想理解这些特定的执行我们必须关注运行这些执行的主机和网关，还要理解接入 Internet 的特定的网络，而不是体系本身；我把这些接入 Internet 中的网络、网关和主机称为“实现”。实现可以用它们的规模大小来区分，它可以建立在速度 1200 比特每秒的电话线上，也可以建立在速度超过 1 兆每秒的网络上。很明显，吞吐量（速度）是衡量规模大小的一个标准。相似的，延迟的长短也是一个标准，一个延迟时间以十万分之一秒计，一个以秒计，那么像实时语音这种应用在这两个实现上效果会截然不同。有些网络在网关和路径上设置了大量的冗余，这种网络就比较稳定，因为发生错误时可以重新配置。而有些网络实体为了降低成本，采用单线连接，一旦出了问题网络就会分成两部分。

Internet 体系容忍实体的多样性。但是这会给那些特定实体的设计者很大的难题。一个主要的问题是体系如何给设计者以指导，这些指导牵涉到实现各种各样的服务和你的设计之间的关系。比如说，设计者必须弄清楚以下的问题：对于一定传输速度的服务，下层网络的带宽应该是多少？给出在实体中出现的特定的错误可能，应该在实体中加入何种冗余来抵御这些错误？大部分已有的网络设计帮助对这些问题都效果不好。比如说，协议验证用来检验协议的规范性，而实际上它们根本不会检查那些有关服务的本质细节，它们只依照规范检查很有限的协议逻辑的正确性。即使这些检验工具对规范和执行层次上的检查是比较有效的，那也保障不了真正使用时不出严重的问题。一个典型的例子是，逻辑被证明通过了，却在执行时效率严重低下。对这个问题研究后表明，困难的增加往往不是因为协议本身，而是因为协议执行所依赖的操作系统。我们很难在体系中准确地指出问题具体出在那里，但是我们坚持给出必要的指导。今天我们仍为解决这一问题而努力着。

另一类帮助设计的方式是提供模拟器，这种模拟器带有一个特定的实现和一些在多种载体上容易执行的服务。目前为止还没有一个模拟器包容了所有的网关执行、主机执行和网络执行这些看起来在 Internet 实体中必要的成分。所以说大部分 Internet 实体分析都是摸索着来的，在体系设计领域有句名言：只有一个有足够博学的人才可以将 Internet 分析摸索的足够好。设计者在意的是当前给定条件下是否可以实现要求的服务，而不太在意利用 5%的可有可无的资源。

体系与实际应用之间的关系很是复杂。设计者强烈地感觉到只是考虑逻辑正确性而忽略实际影响因素是大错特错的。但是在他们试图将实际因素考虑到体系内部的时候遇到了很大的困难。这是因为两个原因：一是体系设计目标不是限制实际情况而是适应各种情况，二是实际上可能根本就不会有一种可以形式化描述实际因素的工具，这可能是更根本的原因。

这个越来越严重，因为 Internet 项目组要制作规范化文档，而这些文档要成为军用标准的。对于政府合同大家都清楚一点，政府总是希望承包人要在接受现有标准的基础上工作。如果 Internet 考虑实际因素，那么强制性要求就会加入到规范中。制定那么多强制性要求就会显得很琐碎，比如说强制要求设备必须每秒至少能传输 1000 包。这种强制肯定不能加到 Internet 体系中，那我们就应该制定一个为实际使用指导的标准，来指导他们怎么样设计才能实现要求的服务。我们还不清楚在体系里该怎么样去指导个人来实现这一任务。

9.独立遨游的数据包

Internet 体系一个基本特征就是在底层网络是用数据报来整合数据传输的。此文已经说道在体系里使用数据报技术的重要性。首先，它省去了普通数据包中的保存复制状态信息的开销，这意味着可以修复网络错误而不必要求状态信息；其次呢，数据报屏蔽了各种不同服务本身数据传输方式的差异性；与支持确定服务的虚电路相比，数据报可以提供更基本的服务，而这些服务又可以进一步支持其他更高级各种各样的服务；第三，数据报代表着最基本的网络服务，这种服务支持各种各样的网络实体成功互联。使用数据报是极其成功的决定，因为它让 Internet 帮助实现了它最重要的几个目标。

有那么一种错误的假定，有人认为使用数据报是因为在更高层服务本来就需要数据报服务。换句话说，应用层本身要求一种数据报服务。事实上，极少有这种需要。Internet 中的一些应用，比如说数据、域名的简单查询操作，这时候我们采不可靠的数据报。但是大部分服务有更复杂的传输要求。有些服务要求增强型的可靠性，还有要求延迟的平滑性，总之就是绝大多数比数据报要复杂得多。这里我们必须弄清楚，数据报一种建立隔离区的手段，而非一种具体的服务。

10.TCP

在 TCP 的发展过程中有几个有趣甚至有争议的决定，并且它也是经历过好几个版本之后才成为现在的一个比较稳定的标准。有些设计决定，想窗口管理和端口地址结构，是在 TCP 手册里的使用记录里讨论的。但是我要再次声明，这种决定的动机有时候是不充分的。在这一节里，我尝试着发掘一些后来成为 TCP 一部分的早期论证。这一节讨论的内容肯定不完全；想完全了解 TCP 的发展历史又要专门写一篇和本文一样长的论文才行。

最早 ARPANET 的主机到主机的协议提供的流控制包括比特和包两种方式。这太复杂了，TCP 的设计者也感觉涌一种规范就可以了，于是就对字节传输进行校准，而不是包。流控制和应答也是机遇字节数而非包数。的确，在 TCP 中数据包本身是没什么意义的。这个决定是由好几个考虑推动实现的，有些考虑现在已经没啥意义了，而有些考虑现在的意义比原来设想的重要的多。应答字节的加入源于向字节流中加入控制信息，所以控制和数据都可以得到确认。空白序列的使用被一种特殊控制信息技术所取代了。有时候一些原始想法是很简

单的，实际操作中却会变得很复杂。

选择字节流的第二个原因是为了方便的把包切分成更小的包以适应网上的包尺寸限制。但是这个功能随着 IP 从 TCP 中分离出去了。IP 被迫使用了不同的方式去切分包(因为某种原因)。

选择字节流的第三个原因是为了方便在发送端将小包聚合成一个大包（如果要求数据重发）。当然这种优势是否重要还不得而知。事实证明它很有必要。像 UNIX 这样的系统中它们内部数据传输时经常会只传输一个字节的数据(可能有人从网络的角度出发,认为这么做太傻了,但这就是事实，并且在内容远程登录时这是必需的），在这样的主机上经常会有大量的一个字节的包要传递，以至于慢的主机来不及处理它们，只好丢包然后要求重发。

如果重发时还是原来的小包，那么这种操作可能繁琐到使系统宕机。但是如果把字节聚合成一个大包，那么重发就是更高效，更适合于实际操作。从另一个角度看，字节应答被认为是产生这个问题的罪魁祸首。如果使用基于包的流控制则不会有这种淹没发生。虽然如此，但是它可能带来吞吐量小的严重限制，特别是大量小包被发送时。还有，对于接受方而言，要接收大量的大小未知的包（每个包有多少数据是发送方决定的，接受方并不清楚，可能从 1 字节到 1000 字节），这是很麻烦的。

回顾一下，TCP 要是想支持有效的各种各样的服务的话，包和字节方式都得进行校验，就像原来 ARPANET 那样做的一样。

另一个决定是关于 EOL 的，它现在已经被 PSH 取代了。最初使用 EOL 是想将字节流切分成记录，然后把数据以记录的形式放到独立的包中，而这种做法是和重发聚合包的做法相违背，所以它的语法就被削弱了，也就是说它只处理网络或 TCP 缓存去产生的纯应用层的一个或多个数据。这里说“一个或多个”（而不是“就一个”），也就意味着可以进行整合来实现数据压缩的目标。但同时，语法被削弱意味着大量的应用程序要自己把数据流切分成记录。

在这场关于 EOL 语法的革命中，有些可能产生很多负担的特殊情况很少有人谈起。对于基于缓存区机制的主机，TCP 字节流模式会在一种几乎不可能的情况下导致很大的问题。现在假定有一个这样的主机，它要接收一串由各种尺寸的缓存区组成的数据流，当缓存区满或 EOL 出现时一个缓存区将被返回。现在我们考虑这样的情况，数据流发生了乱序，乱到有 EOL 包跑到前面一个错误的位置了，这时候系统会把当前一个缓存区返回（而它可能还没有满)，这导致下一个缓存区里的数据会被放在一个错误的位置。这时侯就要有一个记账本,虽然看起来似乎没有必要。

有个提议用来应付这个问题：在队列里空白处都是 EOL，直到一个双缓冲大小为零的值。换句话说，EOL 可以成为标示字节流来实现缓冲区管理的工具。

这个主意当时没被采纳，大家认为这种情况太特殊了。现在开来，把几种处理队列空白和缓存区管理的算法应该加入到 TCP 中。在当时设计者还没有足够的远见去实现一种通用的方法。

11.结论

Internet 体系已经被各领域广泛应用并且衍生出了好多相似的体系，从这个角度来看它是挺成功的。同时它的成功也产生了一个问题，设计者可能并不会很好的满足实际用户的需要。比如说，账户、资源管理以及分布式管理操作等问题需要多加关注。

尽管报文技术可以满足 Internet 中最重要的几个目标，但是对于后面几个目标效果却不是太好。比如说，报文技术很难处理好资源管理和易读易控的问题。就像前面讨论过的，绝大多数数据报是从源端到目的端的数据包队列，而不是应用层那样的独立单元。但是网关并分不清这些关系，它们只会独立平等对待每个包。报文模式的广泛使用屏蔽了其他包含重要资源信息的层，而这些信息恰恰可以实现后面的几个目标。

这意味着在下一代体系中会有更好的隔离区取代数据报模式。这种隔离区应该可以把所有的数据都看成一样的，不会假定数据包具有某种服务。我用“流”来表明这一特征。而网关应该掌握流的状态，知道这些流的性质，而这些状态信息并不会影响到流对应的服务。这种工作应该通过定期发送消息来保证流与服务正确对应。这样的话，数据流撞车时把状态信息丢弃也不会影响服务的使用。我把这种状态叫做“软状态”，这可以让我们很好实现最重要的健壮性和灵活性，同时也能搞定资源管理和用户管理的问题。这种新一代隔离区选择问题是目前 DARPA-Internet 项目的一个研究方向。

12.感谢，用一个历史性的眼光

不可能向 Internet 项目的所有参与者一一表达谢意；因为在过去 15 年的发展中有数以百计的设计者、操作者、作家和批评家做出了贡献。的确，最值得写篇论文的是项目得以良好进展的管理程序。参与者来自大学、研究室、公司，他们一起实现了这个共同的目标。 TCP 的最初版本是由 Robert Kahn 和 Vinton Cerf 搞出来的，早在 1973 年他们就清楚地认识到，一个合理的协议必须可以很好融合利用现有的网络技术。从在 DARPA 任职时开始，他们就领导这个项目组，一直到 TCP 和 IP 成为美国国防部的标准。

本文作者 70 年代中期加入这个项目组，1981 年接手负责 TCP/IP 相关事务。他感谢他的所有同事，特别要感谢那些朋友，是他们花费时间帮助重新拣起本文中提到的那些被遗失的历史。