清华大学

综合论文训练

题目：一种内容中心网络全新实现机制的研究

系别：自动化系

专业：自动化

姓名：马戈

指导教师：曹军威

辅导教师：陈震

2013年6月 12日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

（涉密的学位论文在解密后应遵守此规定）

签　名：_____导师签名：_____日　期：_____

中文摘要

互联网的使用正在从 host-to-host 模式向内容分发的模式转变，如视频内容占据了互联网传输流量的大部分。互联网服务提供商、内容提供商和其它第三方已经部署了大量的 CDN 来改善用户体验。然而，作为一个内容分发解决方案，CDN 依然面临着巨大挑战，如复杂的控制层、彼此间缺乏协调、域名解析可靠性等。

内容中心网络 CCN 作为一个从设计上改变网络架构的体系，内在设计了内容分发功能，希望能够彻底解决该问题。在本文中，搭建了 CDN 和 CCN 基于云计算的测试平台，完整模拟 CDN 与 CCN 的工作原理与实例，从多角度、多方面，全面系统地比较了它们在内容传输上的差异，得到了各自的优势与改进、融合方向。

当前的 CCN 原型系统是建立在 TCP/IP 协议的覆盖网之上。本文设计了一个基于以太网帧协议、不依赖传统 TCP/IP 协议架构、完全依靠命名数据路由的 CCN 网络。为了验证正确性，搭建了物理测试 CCN 网络，通过实验来对比该网络和原来的 CCN 覆盖网络的性能差异。

关键词：内容分发网络；内容中心网络；互联网体系结构；未来网络；性能评估.

ABSTRACT

Internet usage is shifting from host-to-host model to content dissemination model, e.g., video content delivery accounts for the majority of Internet traffic. ISPs, content providers and other third parties have already deployed CDNs to improve user experience. However, as an ad-hoc solution to the content dissemination problem, CDNs still face big challenges, such as complicated control plane.

With a fundamental change in network architecture, CCN (content centric networking) incorporates content delivery functions in its network layer, and appears to be a most promising solution to this problem. In compared with CDN, we try to explore the advantages and disadvantages of CCN architecture. Hence, in this thesis, we construct comparable CCN and CDN test beds separately and systematically compare the content delivery of CCN and CDN.

Current CCN reference design is based on TCP/IP overlay. Pure CCN needs not the support of TCP/IP protocol suite. In this thesis, a CCN implementation we call CCN underlay, is designed without TCP/IP support. CCN underlay is over Ethernet framing protocol directly. A test bed over raw Ethernet frame is constructed and the size of the Maximum Transmission Unit (MTU) is modified to load CCNx data packet. And experiments are conducted to test the performance of CCN underlay compared with CCN overlay.

Keywords：Content Delivery Network; Content Centric Network; Future Network; Internet Architecture；Performance Evaluation.

目录

第1章 引言

1.1 背景及概况

互联网是人类20世纪的一个伟大产物，自诞生以来，取得了空前的成功。它已成为整个社会的基础关键设施，改变影响到人们生活、工作、生产、经济、军事等各个方面，其重要性毋庸置疑。然而，随着互联网的飞速发展，当初设计互联网作为通信的初衷已经改变，当前互联网的工作模式已经无法很好地适应新的背景和许多特殊需求。再加上互联网底层体系结构的限制，使得很多应用变的无效。

考虑当前互联网上的主要信息流量，有近 90%都是信息的发布和订阅，这是当初设计互联网时没有预想到的[1]。由于互联网是端对端的通信模式，它以主机为中心，基于端点的地址进行寻址和数据包的传输，所以任何信息的获取都必须要定位到某个确定的物理主机。因此，实现在互联网上的发布订阅系统，效率都低，每次访问内容，都要间接映射到内容所在的设备，比如，“双十一”期间，淘宝网在数小时内会有上亿的访问量，如果没有内容分发设计，用户都去访问总部的服务器，势必会造成网络的拥塞。

其次是移动性差，因为IP在互联网上具有双重语意，既表示一个主机，同时又指明了物理位置，所以非常不利于移动，无法满足网络移动性的需要。加上现在移动设备的高速发展，有相当一部分人都用手机、笔记本等移动设备上网，这也更突显了网络移动性差的问题。

第三个问题是安全性问题,在设计之初,安全问题根本没有被互联网所考虑，以至于互联网后来设计、增加了许多安全机制。这也是安全性问题一直对当前互联网产生不利影响的原因，同时很多重要的应用也因此被制约了。

为了解决这些问题，P2P(Peer-to-Peer)、CDN（Content Delivery Network）[3]等体系结构应运而生，uTorrent[14]等覆盖网都是其典型代表。但是，这些体系结构虽然都改善了内容获取和分发的效率，但是却没有从根本上解决该问题。因为他们仍是建立在TCP/IP体系结构[11]之上的，而TCP/IP是存在数据传输的瓶颈。比如，高清视频点播凸显了这一瓶颈，拿 P2P 系统作说明，因为 Peer 只能从很小一部分 Peer 中获取数据块，简而言之就是在 Peer 下载同样的内容和网络拓扑时只拥有很有限的信息。此外，当前 IP 网络还存在不易移动（现在移

动设备越来越多），路由表的可扩展性，网络可靠性，地址不足等问题。所以，一场由“Where”到“What”的根本性转变“创新”也被提出[19]，内容中心网络（Content Center Network，CCN）[2]就是其中的主要代表。CCN 采用内容数据的名字路由，通过路由器来缓存内容，从而更适合内容分发。

1.2 内容分发网络（Content Delivery Network, CDN）

内容分发网络（CDN）是一种典型的覆盖网络，在了解 CDN 之前，需要先了解覆盖网相关概念。

当前，覆盖网被广泛部署在互联网上，能够为绝大部分应用提供基础设施；它采用与现有互联网中相对部分不同的、有竞争性的方式来尽可能地转发、传递和处理数据；而且它完全可以独立于第三方下，在一个有组织协调性良好的方式下运作。

CDN 的全名是 Content Delivery Network，中文为内容分发网络。它最初设计思路是是绕过互联网上那些高峰通道、偏远线路，即尽量避开影响数据传输高效可靠传输的瓶颈和环节，从而达到数据传输的高效性和及时性。通过在广泛分布的各节点处配置代理服务器，进而组成当前互联网架构之上的一层虚拟化的智能网络，CDN 网络能够根据各节点的网络流量、连接情况、负载状况，以及用户到源站点的距离、响应时间等多方面信息，将用户的请求迅速定位到距离用户最近的服务节点上。这样的好处在于帮助用户从最近服务节点获取请求内容，这大大改善了互联网网络拥塞的状况，降低了用户访问 Web 的响应时间。

作为新的网络构建方式,CDN 可以在传统的 IP 网上发布丰富优化的覆盖网络；在广义的范畴来说，CDN 是高质量、高效率、高秩序的网络服务模式的典型代表。CDN 的广泛覆盖解决了互联网上的一个根本性的挑战：在有效分发和获取内容的同时,降低客户端主机所经历的时延。概括而言，CDN 网络的核心是分布式存储、内容管理、负载均衡和网络请求的重定向四个部分，而内容管理和 CDN 网络的流量管理是其重中之重。通过用户就近请求和节点服务器的负载判断，CDN 确保能为请求的用户提供高效高质量的服务。总的来说，内容服务是基于节点缓存的，也被称为代理缓存，它位于 CDN 网络的边缘节点，它离用户仅有"一跳"的距离。这样的架构设计让 CDN 服务提供商可以代表他们所服务的内容供应商，向客户端用户提供尽可能好的体验。

1.2.1 CDN 类别

CDN 一般分成三种不同的类别：商业性的，合作性的，基于 P2P 的覆盖网。

商业性的 CDN 服务在我们生活中屡见不鲜，例如我们预定飞机票以及一些静态方面的服务，微软补丁就是其中的一个例子。在中国，商业性的 CDN 规模最大的为 Akamai，据悉 10-15％的网站都有它的影子，而且在全球范围内，拥有 10,000 以上的互联网供应商。

合作性的 CDN，比较常见的有 CoralCDN 以及 OpenCDN，合作的目的是为了提供无商业性的利益好处。由于商业性弱，经济来源多为捐赠的设备和技术，所以他们的性能有限，技术含量低。但是在另一个方面，他们用户更加广泛，更加普及，有更大的潜能。他们服务的方式可以举出下面例子阐释。网站发布者（例如 http://www.x.com ） 在 URL 后加 nyud.net:8090 ， 之后网址变成 http://www.x.com.nyud.net:8090，此时网站内容便缓存在 CoralCDN 上面，就称“Coralize”出一个网址。

第三种是基于 P2P 的覆盖网，相比于合作 CDN，他多出了搜索功能，而且网站里面内容可以维护版权，同时相比于合作性的 CDN 网络节点是内容的缓从，他的内容节点可以同时充当客户端和服务器的作用。在这里举 BitTorrent 作为一例，BitTorrent 现在逐渐为人们所知，它是一个分发网站，最初是 Linux 的分发渠道之一。在 BitTorrent 网站上，发布方用对等 CDN 技术来减少成本。然而对于不适用与于这种降低成本的网站内容，对等的内容便可以相应代替，减少成本，与此同时，侵犯了版权。

1.2.2 CDN 特点与不足

对 CDN 的架构原理进行分析，我们发现 CDN 具有以下特点与不足之处：

1，实现了本地缓存加速。改良了 Web 站点（特别是含有大量文字或图片的站点）的访问速度，降低了响应时间，从而大大增强了 Web 站点的稳定性；

2，实现了镜像服务。有效减轻了同运营商之间沟通问题的不良后果，从而提高了不同运营商的合作网络的速度，这样即使用户处于不同运营商管理的网络中，依然可以快速高效的访问；

3，实现了访问加速。客户端用户根据负载均衡技术和某些策略，自动地选择代理缓存服务器，选择最近或者最快的缓存服务器，提高了远程访问的速度；

4，实现了带宽优化。通过自动生成服务器的远程镜像缓从服务器，用户远程访问的时候便可以，直接从缓从服务器上读取，缩短了远程访问的时候的带宽、可以讲网络流量分流、从而分担了 WEB 服务器负荷；

5，提高了安全性。由于 CDN 节点服务器覆盖范围广，所以各个节点之间可以相互补充，相互保护，从而减轻了收到黑客袭击的受损程度，也保护网站不受到来自于 DDoS 的攻击，从而提高了服务水平。

对于目前国内的网站来说，CDN 已经达到了绝大多数的覆盖。然而 CDN 的可靠性如何？研究表明，CDN 的结构仍然多节点，而与此带来的好处便是，CDN 的某一个节点出现问题的时候，其他的节点便可以相应取而代之，防止了整个网点的崩溃。CDN 的另一个巨大优点便在于可以在不涉及服务器，新增宽带等成本，不涉及服务器的镜像成本，不涉及维护人员支出的基础上来完成覆盖全国的网络设置。

另一方面，实时性不能保证是 CDN 的一个重大问题，由于源站点和代理缓存二者的信息会出现时差，不能及时同步，给用户带来不好的使用体验，这个是将来需要不断改良的地方。

1.2.3 CDN 的重大意义

作为目前覆盖率极高，使用率也很高的网络，CDN 的存在具有以下重大的意义：

首先，CDN 促进了 Internet 行业的分化细分。通过边缘化网站内容，可以帮助用户就近访问，进而提高访问速度，稳定性，提高服务质量，优化用户体验，维护网络秩序，细分网络行业分类；

其次，它体现了管道智能化。CDN 的分发和缓存机制，保证了边缘节点可以存储热门访问内容，同时通过有效调度，平衡节点之间的压力，有效分担压力过大的节点负荷。

第三，促进了第三服务业的发展。通过提供更多内容传递的增值业务，例如用户行为统计分析，网络监督管理，网络调节等，将增加与内容传递有关的岗位，促进第三方服务发展。

1.3 内容中心网络（Content Centric Network, CCN）

CCN 就是将网络中的一切都作为内容数据来处理，网络从主机互联变成了内

容数据互联，其核心是内容数据，并通过对内容数据的命名来标识每一个内容数据。从整体来说，有名字的内容数据流动组成了网络系统，它能将每一个内容数据都区别开来，但具体的内容数据意义网络系统却无法解析，这需要依靠生产者和消费者的上层的应用来解析。各种内容的流动运行驱动了整个网络和终端，而网络的作用就是所有内容数据的流动控制和缓存管理，并能快速准确地将正确的内容发给客户端。用户或者应用只关注内容数据本身，而与该内容数据块的其他属性没有关系。

图 1.1 显示了 CCN 和 TCP/IP 的架构比较[16]，可以发现 CCN 和当今 TCP/IP 网络的体系结构在外型上很相似，都是中间细、两头宽的沙漏模型，不同的是 CCN 用 Content chunks（数据块）替代了 IP 包。从网络的角度看，CCN 不再是对物理主机的命名，而是对内容数据的命名。另外，它在网络中采用了分布式缓存机制，将经过的信息数据都缓存起来，从而加快了其他用户访问相同信息数据的响应速度，分流了源节点的访问用户量，同时网络中的流量可以得到有效地减少。

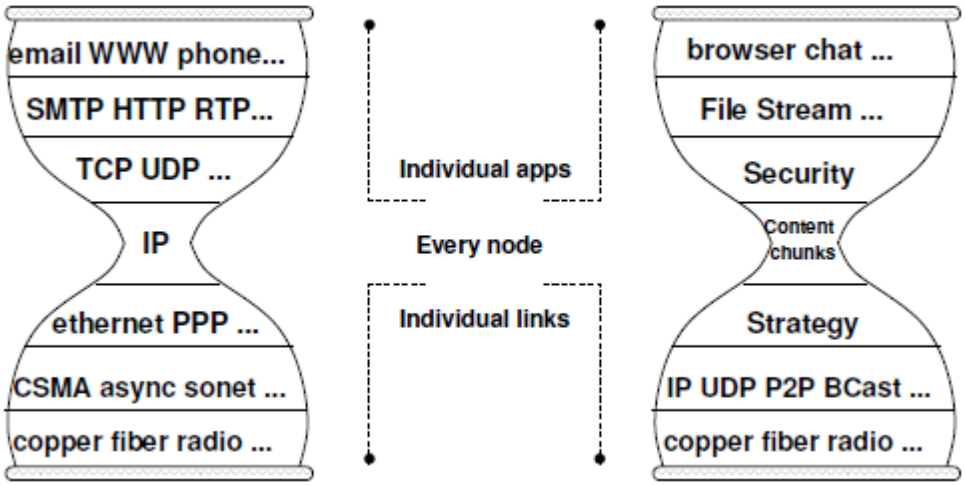

图 1.1 TCP/IP 和 CCN 架构

1.3.1 CCN 设计原则

1，沿用沙漏模型。实践证明这种“沙漏”的结构促进了互联网的快速增长，其原因是它允许下层和上层不断革新变化而又不改变其核心，所以在 CCN 的设计中也保留了这一结构；

2，继承端到端原则。端到端原则的思想是均衡系统的性能和代价花费，几乎所有分层系统在设计时都考虑该原则，因此 CCN 的体系结构也继承了该原则；

3，重视安全性。互联网在设计之初是没有考虑安全性的，所以之后的设计增加了许多安全机制，这种修补的方式对互联网产生了很多的负面影响，而 CCN 则是要把安全性直接贯穿到体系结构的设计之中；

4，流量均衡。流量均衡是一个网络稳定运行的必要前提。网络本身的设计并没有考虑流量的调节，所以这只能依靠端的传输协议；

5，保留路由和转发层的分离。这种结构的保留将允许转发技术，路由技术同时发展；

6，中立体系结构。简化用户选择。

1.3.2 CCN 的优势和不足

CCN 作为彻底改变了 TCP/IP 的体系结构，具有很多的优势：

1，更安全。前面已经说过，CCN 是把安全性直接贯穿到体系结构的设计之中，而它自带的数据签名为未来互联网提供了必要的安全机制，比如数据的完整性校验和来源校验，当客户端接收到它自己请求回来的数据后，它可以通过检验签名信息，进而知道它收到的是否是真实发布者提供的信息内容。更重要的是，CCN 不需要考虑数据通道的安全问题，因为在 CCN 的网络下，数据的传输是没有固定的通道可循，因此，客户端可以从有数据的缓存中任意获取。另外，CCN 能够抵御很多 DDOS 攻击；

2，高性能。CCN 相对于当前的 IP 网络具有很多明显的优势，比如它既能支持内容的分发，又能允许多播功能。另外，在动态内容、点对点通信等方面，CCN 也具有比 IP 网络更高的灵活性、安全性和鲁棒性；

3，均衡流量。一个稳定的网络，理论上应该具备流量自调节的功能，但互联网在最初设计之时并没有全面考虑，忽略了网络中的流量拥塞和控制问题，导致当前网络几乎崩溃。而 CCN 具有天生的流量均衡机制，在数据传输时，可以根据链路状况、响应时间等因素，综合考虑选择传输策略，进而调节整个网络的流量。

4，绿色环保。作为当前学术界和工业界的热点研究，在性能提高的同时，更要权衡利用能量率。尤其是如今的互联网每时每刻都消耗着人类巨大的能量，高效的能量利用率对服务提供商甚至对整个人类都是有益的。

5，简化部署。当前互联网上的众多应用都需要依托于复杂的中间插件，这就需要将应用的信息和相应的 IP 地址之间映射起来，而这些应用的部署和开发关系将会在 CCN 网络中大大简化；

作为还没有研究彻底的新新科技，CCN 的缺点也有一些，但是这是事物发展

初期的必经之路，我们需要做的，就是不断发现问题，不断直面问题，不断改善。CCN 的缺点主要有以下几点：

1，需要带有快速更新的精确字符串匹配。CCN 转发层中，带有快速更新的精确字符串匹配由 PIT 或者 CS 查询实现。最坏情况下，每个包都需要更新，而现在的研究没有支持这一方面的基础；

2，可变长度和无界限命名的最长前缀匹配（LPM）问题[17]。尽管 LPM 在 IP 查询和 URL 过滤中被广泛研究，但现有的 IP 查询方法并不适用于 CCN。CCN 包拥有可变长而且无界限的命名，而 IP 包只有固定 32bit 长的地址。大多数提出的 IP LPM 方案在处理长键值是会变慢，且消耗大量内存资源；

3，大流量维护。流量维护是实现 PIT 的另一种方式。在硬件设计中，实现指针结构和动态 PIT 效率很低。取而代之，我们选择维护（名字，输入接口 ID，输出接口 ID）的数据对而实现转发层的功能。流量维护和 IP 网络中基于流量的监控很相似，不同的是 CCN 包需要处理可变长的命名，而不是固定长度的数组。这也导致了流量表会变得很大，的维护工作变得很有挑战。

1.3.3 CCN 的工作机制

CCN网络中，通信是由数据的请求者发起，数据进行的是块级传输。CCN网络中有两个包类型：Interest Packet和Data Packet，见图1.2。其中请求者广播 Interest 包，请求内容，监听节点如果有该内容，则响应Data包。Interest包和Data包和位置无关，对于一个相同广播媒介，如果有一个兴趣包请求，那么其他对这一内容感兴趣的请求者可以共享该请求。

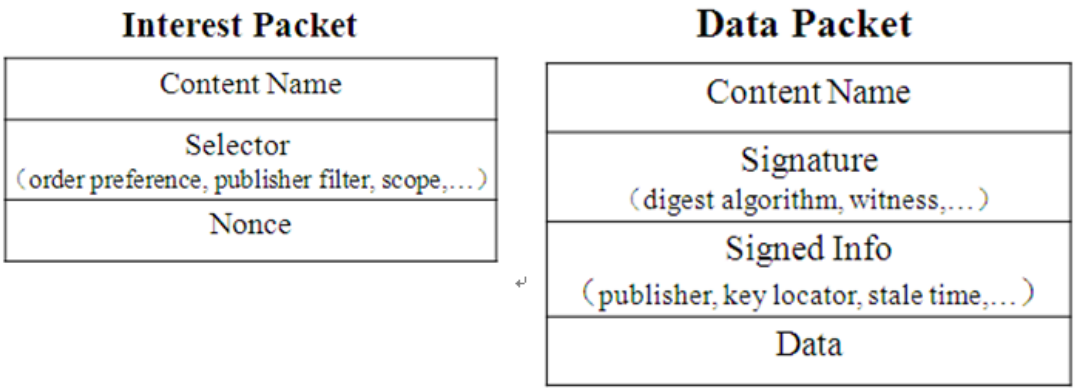

图 1.2 CCN包类型

如果Interest包的名字前缀是data包的名字命名的前缀，则该数据包满足该

请求。总所周知，内容的命名十分复杂，动态生成的不同内容名字，有效保证了不存在该请求内容的节点服务器接收到该兴趣包，而随即发生的内容内容，这同现在的动态 Web（同时支持静态和动态内容）一样。

当一个兴趣包或者数据包到达一个接口，首先需要对名字进行最长前缀来相应查找，然后需要通过以下3个数据结构来分发：FIB（Forwarding Information Table）[17]、CS（Content Store）[18]、PIT（Pending Information Table）。其中，FIB表类似IP的FIB表，但不同是它不限于一个出口，可以配备一套出口。CS为了尽可能地提供数据给其他请求者，可以设置缓存的转发策略为尽可能长时间地缓存那些最新、需求最多的数据包。PIT 表则记录了已经转发的兴趣包，当响应请求的数据包借助PIT表的某条目转发后，该条目应被删除，以保证数据包能顺利传输到请求者那里。图1.3显示了兴趣包的处理过程。

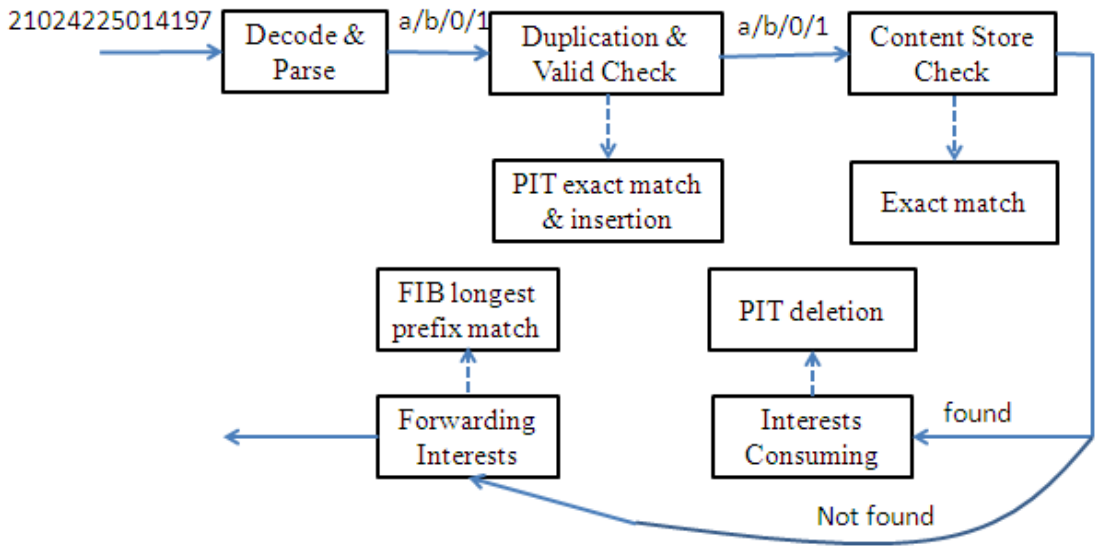

图 1.3 Interest包处理过程

CCN网络中简化了数据包处理过程，当数据包到达时，首先对数据包的内容命名字段进行最长前缀匹配，先在CS中匹配，如果存在，则丢弃兴趣包；再在 PIT 表中匹配条目，如果仍然存在，则发送到请求者那里，同时缓存在CS中，如果没有匹配，则丢弃该数据包。图1.4展示了数据包的处理过程。

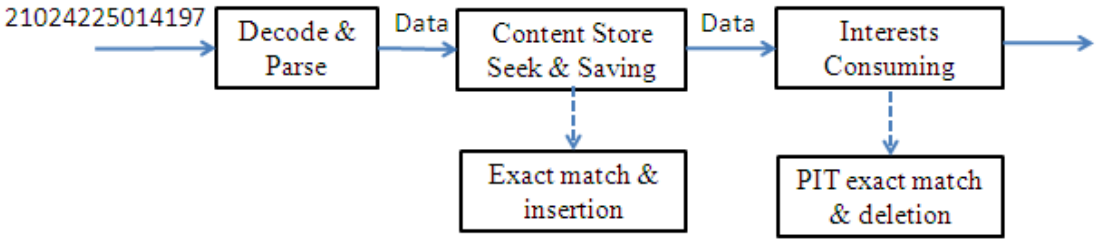

图 1.4 Data包处理过程

1.4 本文工作简介

面临Internet的种种问题，当前国内外主要有“改良”和“创新”两种思路。CDN体系结构作为“改良派”中的佼佼者，现如今已发展的技术相当成熟、应用非常广泛；而CCN作为“创新派”中最热的一股热潮，旨在从根本上解决互联网问题，也得了越来越多的人的支持与参与[13][20]。

本文主要比较CDN和CCN两种体系结构，通过自己搭建的CloudStack开源云计算平台，在上面分别搭建了CDN与CCN网络的试验平台，对比二者在结构、实现复杂度、传输速率等性能、发展应用前景，以及拓扑结构、策略层等方面的优缺点，从而得到各自优势与改进方向。接着设计、实现了一种基于以太网帧协议、不依赖传统TCP/IP协议架构、完全依靠命名数据路由的CCN网络，称之为CCN Underlay网络；把当前建立在TCP/IP协议之上的CCN网络称为CCN Overlay网络。并通过实验验证CCN underlay网络性能优于原来的CCN Overlay网络。

第2章 相关技术背景

为了能更好地控制、管理和部署实验平台，我们研究了采用云计算技术的 CloudStack 开源云平台来实现实验平台搭建，以虚拟化技术模拟搭建了 CDN 与 CCN 网络。接着，我们继续调研了 CDN 的核心关键技术，包括 Apache、Squid、nginx，BIND 等等，为构建 CDN 网络奠定基础；最后我们分析 CCNx 的源码实现，并部署了 CCN 网络，为从多角度、多方面，全面系统地比较了 CDN 和 CCN 的实验做好了准备

2.1 CloudStack 平台

CloudStack 是一个综合，并具有很高拓展性的完全开源的云计算平台。CloudStack 可以方便快捷地让用户在现有的网络基础上建立自己管理的私有云服务端，并能够在协调服务器、存储、网络资源等方面给予用户很大的帮助，从而使用这可以高效地管理云平台。

2.1.1 CLoudStack 的基础架构

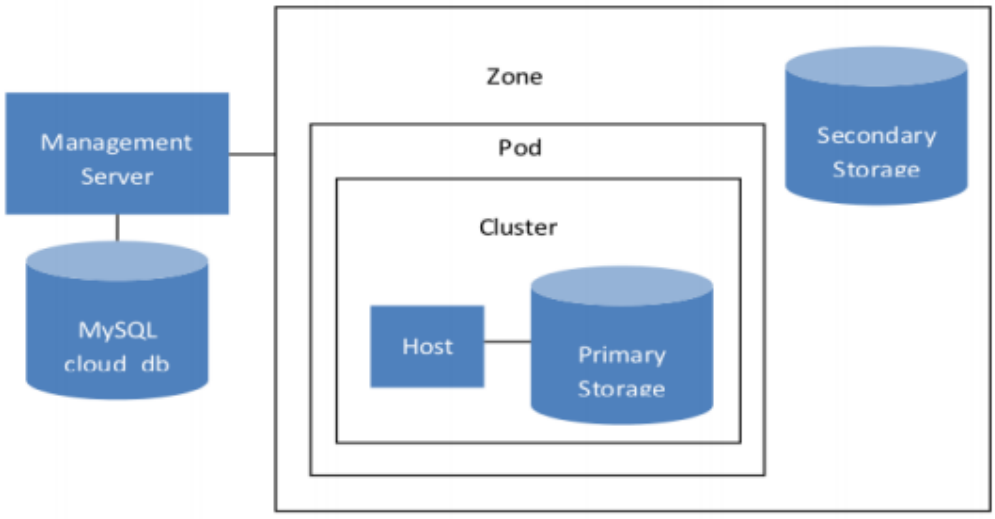

图 2.1 CloudStack基础架构

从图 2.1 可以看出，CloudStack 的最底层是 Host，即主机；多个 Host 构成 Cluster，即集群；多个 Cluster 组成一个 Pod，即一个子网；多个 Pod 则组成一个 Zone，即一个局域网；当然它还需要 MySQL 的数据库管理用户信息，需要 Management Server 管理整个平台，以及一些存储设备。

CloudStack 基础设施的具体概念如下。

Host（主机）：是 CloudStack 中部署的最小组织单元。一个 Host 是一台独立的计算机，提供资源设备来运行客户端的虚拟机，虚拟机软件可以是 KVM、Vmware 等等，但对终端用户并不可见；

Cluster（集群）：提供一种集合 Host 的方式，在集群中，每个主机共享同种硬件设备，运行相同的管理程序；而且在同一集群中，主机可以在不终端服务的同时，完成到另一个主机的动态迁移；

Pod：包括至少一个带有主机的集群，同一 Pod 中的所有主机共享同一子网；而且 Pod 对客户端的用户并不可见；

Zone：是 CloudStack 中部署的最大组织单元，包括至少一个 Pod，构成了一个局域网；既有主存储服务器，也有辅助存储服务器；而且 Zone 对客户端的用户可见；

Primary Storage（主存储）：可以与集群通信，为集群内部的所有主机提供存储卷；

Secondary Storage（辅助存储）：可与与 Zone 通信，上面存储了 Templates（模版）、ISO 镜像、snapshots（磁盘卷快照）等信息。

2.1.2 CLoudStack 的优势

1，可视化界面。CloudStack 优势之一就是它有了可视化界面，大大简化了部署、管理工作，基本上所有工作都可以只通过鼠标点击来完成。图 2.2 就显示了 CloudStack 部署 Vmware 集群的界面。

图 2.2 CloudStack 中 Vmware 的部署

2，支持多种虚拟化平台。CloudStack 平台可以支持多种 Xen，KVM，VMware

在内的各种虚拟化平台，可以同时管理不同平台下的集群而互不影响，具体结构见图 2.3。

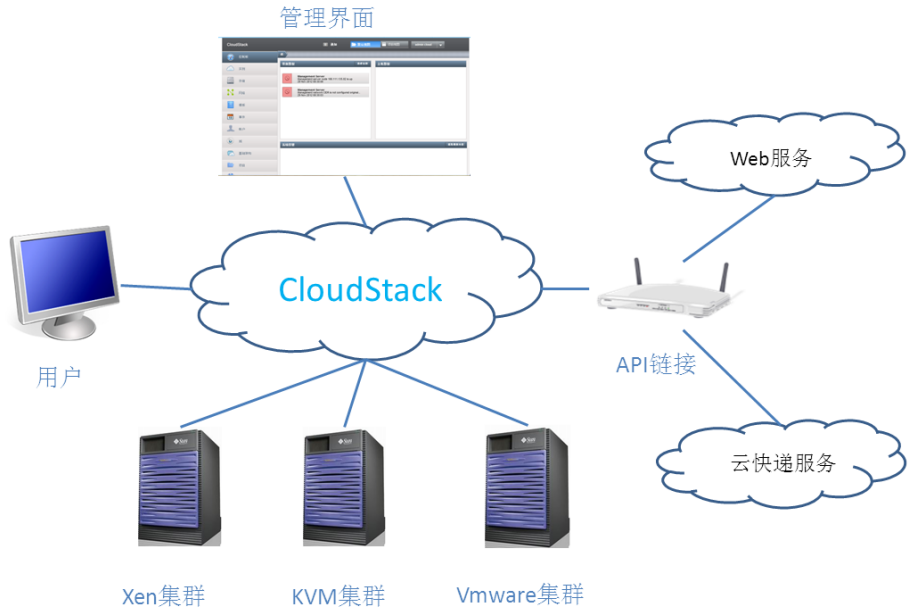

图 2.3 CloudStack 管理多虚拟化平台结构

2.2 Web 源服务器——Apache HTTP 软件

Apache HTTP[9]服务器软件是一个完全开源，具有极高的处理能力、拓展能力和稳定性，而且它的安装与环境配置都相对简单，因此它成为了世界使用排名第一的 Web 服务器软件。

当 Apache 服务器在运行出故障时，一条或者多条记录细节的消息将会被写入错误日志，所以基本都可以通过错误日志来判断问题、解决问题。错误日志的默认位置在/usr/local/apache2/logs 目录下，而且还可以通过 ErrorLog 指令来确认错误日志的确切位置。

Apache HTTP 的配置文件 httpd.conf 保存在/usr/local/apache/etc 目录下，其中需要重点关注的项目如表 2.1 所示，

表 2.1 Apache HTTP 服务器配置文件项目说明

配置项目	说明
ServerRoot	描述 Apache 服务器软件的安装路径，与安装配置时填写的 prefix 参数一致
Listen	描述 HTTP 服务侦听的端口号，默认是 80
Document	描述用于 HTTP 服务的 Web 页面文件的存放位置

Apache HTTP 服务的启动需要执行如下指令，

cd /usr/local/apache/bin/

#./apachectl start

配置和启动完成后，在客户端的浏览器的地址栏中输入 Apache HTTP 服务器的 IP 地址（如 http://166.111.135.90），将能看到如图 2.4 所示的内容，

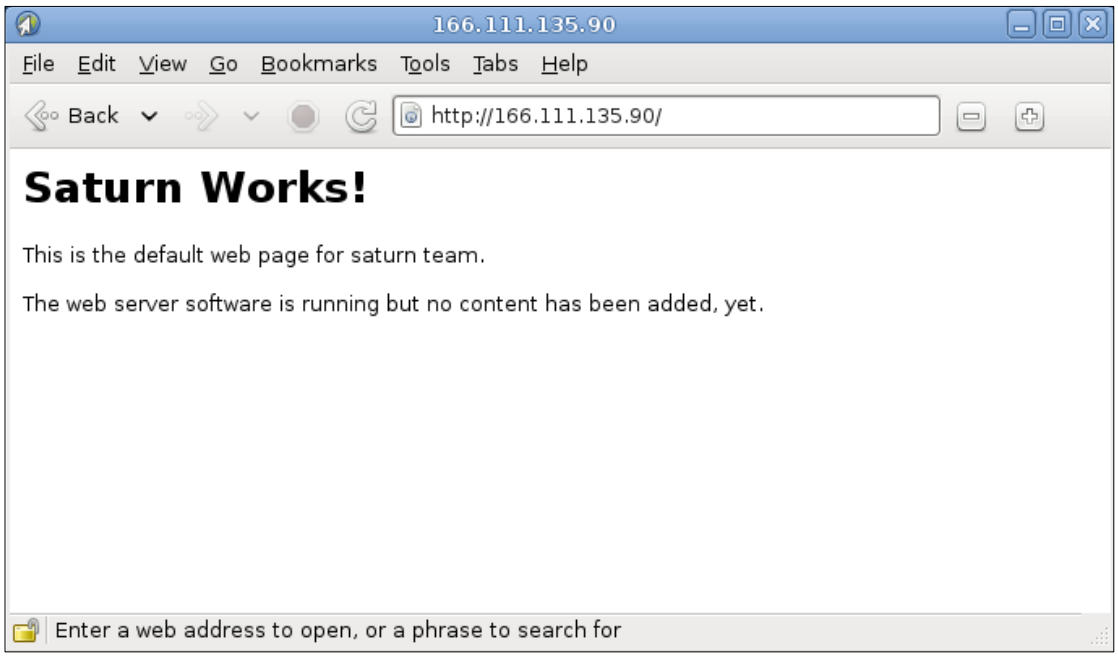

图 2.4 Apache HTTP 服务器安装成功界面

2.3 Web 缓存代理——Squid 软件

Squid[8]是当前应用最为广泛的 Web 代理缓存软件。它的部署方式灵活多样，因此被广泛用于 Web 内容缓存，来提高 Web 服务器的响应速度。各网站通过将 Squid 部署在分布各地、更靠近当地用户的站点上，利用它能存储经过自己站点

的内容数据的特性，即当用户发送一个请求时，如果 Squid 上存有该请求内容，那么用户直接在 Squid 代理缓存站点上获得其所需的内容，而不必再去访问源站点，这大大提高了用户的访问效率，同时源站点的访问压力也得到了缓解。更重要的是 Squid 还能实现服务代理，即可以模仿用户的真实行为，如当 Squid 站点中没有命中用户的请求内容，它能以用户的名义向其上级（一般为源站点）发送请求，并从上级站点获得所需内容再传输给客户端。

Squid 主要实现了以下功能。

1，Web 代理。Squid 代理缓存站点位于客户端和源站点之间，因此，用户不是向源站点发送请求，而是通过向 Squid 发送请求，再由 Squid 作为代理，间接访问源站点，Squid 并将取回的内容再转发给用户。

2，内容缓存和加速。Squid 软件实现缓存功能，它不断将从上级站点取得的数据存储到自己本地，并对它们进行有效地维护更新。当用户请求的内容存在 Squid 服务器上并是最新的状态时，Squid 将不再向上级站点发送请求，而直接从本地存储向用户传输所需内容，这大大提高了传输效率。

3，日志。Squid 也具有非常强大的日志功能，记录了服务器进程的运行情况、用户的访问情况、缓存的存储情况等内容，比如，客户端从何处何时获得的何种数据都可以从代理缓存服务器的 access.log 日志找到，具体见图 2.5。

```
1362647375.497      2 166.111.135.133 TCP_REFRESH_UNMODIFIED/200 425 GET http://166.111.135.133/123.txt -
FIRST_UP_PARENT/166.111.135.122 text/plain
1362647619.445      0 166.111.135.133 TCP_MEM_HIT/200 339 GET http://166.111.135.133/123.txt - NONE/- text/plain
```

图 2.5 access.log 日志截图

上图记录了 Squid 代理缓存服务器连续两次被访问（166.111.135.133 为访问者）的相关信息，第一条体现了 Squid 服务器被访问时本地缓存不命中，进而向其上一级节点（即 Web 源站点 166.111.135.122）发出内容获取 123.txt 请求的过程;第二条体现了 Squid 服务器在本地内存中发现了客户端所需内容 123.txt，而直接将其返回给用户的过程。其中，每条记录最前面的数字表示 Squid 服务器被访问时间（从 1970 年 1 月 1 日至访问时所经历的秒数）。

2.4 负载均衡——Nginx 软件

Nginx[10]是一款开源的高性能 HTTP 和反向代理软件，同时它也可以作为

IMAP/POP3/SMTP 的代理服务器。Nginx 具有很高的性能，特别是能处理极高的并发负荷，因此它在全球有着很广泛地应用。

在现实的网站部署中，Nginx 经常被用于处理网路协议栈的负载均衡问题。Nginx 负载均衡的核心思想是当 Nginx 接收到来自客户端的 HTTP 请求后，它将生成一个应用服务器的 Upstream 请求，等到该服务器发回用户的请求数据后，Nginx 再将 Upstream 中的数据传输给客户端。

Nginx 主要包括以下几个特点。

1，调度灵活。Nginx 工作在网络协议栈的第 7 层，支持复杂的正则规则，具有更好的负载均衡调度能力。

2，依懒网络弱。理论上，Nginx 只要能够 ping 通就可以实施负载均衡，而且可以内、外网流量有效地区分开来。

3，支持检测。Nginx 能够根据应用服务器在处理页面时，返回的状态码、超时数据等实时信息，检测服务器是否运行出错，并能及时地将返回错误信息的请求重新转发到其它节点上。

2.5 域名解析服务——BIND 软件

BIND（Berkeley Internet Name Domain）是一款依赖 BSD 许可证的开源 DNS 服务软件，是当今互联网上最常使用的 DNS 软件。

BIND 的关键在于，它不仅可以反映域名解析，域名反解析的过程，还可以储存得到域名解析库。具体过程是，BIND 服务器收到域名查找请求，第一步，它在目前所有的域名数据库，缓存里面查找是否有之前查询历史，如果没有相应信息，就向上反映，查找更多的 DNS 服务器，这样一般便可以得到正确地址，如果仍然没有，系统便会报错。

作为一个研究比较深入，应用成熟的 DNS 服务软件，BIND 能够方便快速的均衡负荷。具体操作是，BIND 服务器，为同一个的主机名配备若干 IP 地址，在执行 DNS 查找操作过程中，BIND 服务器按照 DNS 历史记录的 IP 地址顺序查找，这样可以反馈给不同的解析结构，便可以将用户的查找操作分派到若干服务器，从而避免个别过度负荷，均衡负荷。

2.6 CCNx 项目

CCNx 是由 PARC（Palo Alto Research Center）公司研发的基于内容中心网络思想的一套开源项目，该项目为庞大的 CCN 设计出了一个原型系统，给广大网络研究者一个长期开发与研究的平台。CCNx 是当前 CCN 网络架构中实现最好的模型，所以大量的工作都是基于这个开源项目设计和研究的。在 CCNx 中，每个网络数据内容都有自己的一个“名字”，即用命名代替传统的 IP 地址信息，从而作为在网络中底层的身份标识。在 CCNx 的网络结构中，每个节点都是平等的，不过扮演的角色可不尽相同：可能是数据源、中间节点、客户端等等。CCNx 的核心是 ccnd 进程，它支持了数据包的转发和缓存。组成 ccnd 最重要的三个数据结构分别是：FIB(Forwarding Information Base)、CS(Content Store)和 PIT(Pending Interest Table)。其中，FIB 表用来做路由表，CS 则用作缓存经过的数据包，PIT 表用来匹配、记录那些未处理的兴趣包。

图 2.6 和图 2.7 则分别展示了一个 ccnd 内部的体系架构和各个节点之间的通讯。

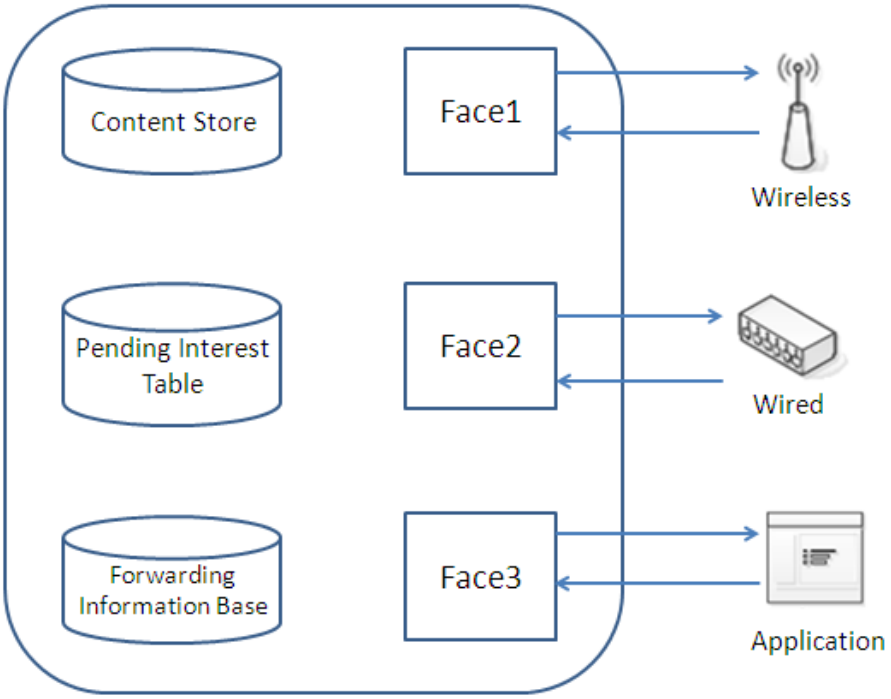

图 2.6 ccnd 的体系结构

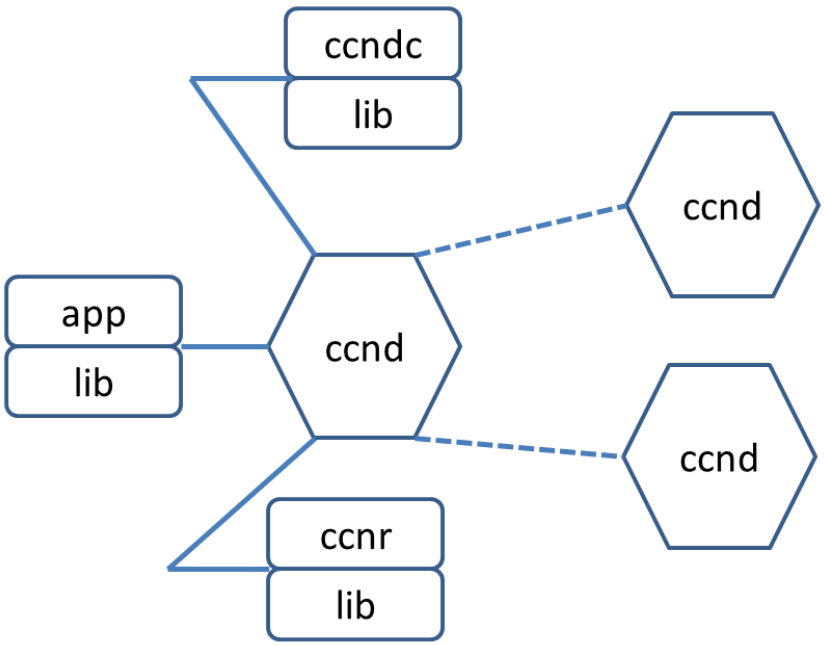

图 2.7 ccnd 间的通讯模型

其中，app 是 CCN 中的应用，ccndc 是 CCN 中的路由控制进程,ccnd 是核心通信进程，ccnr 是它的存储库。

当节点启动 ccnd 后，从该进程中可以清楚地看到收发数据包的过程，图 2.5 显示了发包过程，图 2.8 显示了收包过程。

```
1363079397.277592 ccnd[2837]: debug.4557 dup 9 ccnx:/2000/%FD%05%0F%8FV-%0E/%00%
08%2B%F1/%40%94%D4I%9B%CC%26%F5YD%17V%E1m%20l%99%CC~%9D%D43%BE%C1K%AE%08%009%D7%
E1%94 (4756 bytes)
```

图 2.8 Web 源站点 ccnd 截图

这条记录显示了 Web 源站点在发送数据包，数据包大小为 4756bytes，发送的数据名称为 2000，发送时间为 133079397.277592s（从 1970 年 1 月 1 日至发送时所经历的秒数）。

```
ccn@ccn-virtual-machine:~/test$ time ccngetfile ccnx:/2013new /home/ccn/111
Retrieved content /home/ccn/111 got 6011 bytes.
```

图 2.9 客户端截图

这条记录显示了客户端请求了一个名字为 2013new 的数据，接收了 6011bytes，保存名称为 111。

第3章 实验研究与设计

CDN 作为一个广泛应用在全球的商业系统，其技术与功能已经相对完善；而 CCN 是一个从架构上完全颠覆传统网络的系统，尚处于演化变革中，我们希望对比二者在相同的拓扑结构下的传输性能差异[15]，从而得到各自的优劣势和改进方向。而且，现在 CCN 的源代码实现仍是建立在 IP 协议之上（为了能更好地应用在当前互联网上），我们希望能去掉 IP 的限制，实现数据的无 IP 传输。

3.1 CDN 和 CCN 设计原则的对比

在设计原则上，CDN 和 CCN 有着很多共同点和不同点。当今，二者都是通过覆盖网络实现的。表 3.1 就展示了 CDN 和 CCN 在内容命名解析、缓存、负载均衡等方面的对比情况。

表 3.1 CDN 和 CCN 设计原则上的不同

	CDN	CCN
命名内容解析	域名服务	无（数据块自带）
缓存粒度	对象级或文件级	数据块级(默认 4KB 数据)
负载均衡	循环调度	自平衡
本地缓存	无	有
能否去掉 IP	否	能
实时性支持	无	有

3.2 实验环境

为了进行大规模实验与对照，所有进行比较实验的机器都使用的是虚拟机，所安装的是 Debian Linux 6.0 64 位系统。服务器配置是 HP Z210 工作站，i5 处理器，16G 内存和 2TB 的硬盘，安装的虚拟化软件是 VMWare ESXi vSphere 5.0。

3.3 CDN 实验平台设计

本文设计的 CDN 实验平台[6]含有 6 台机器，包括 1 台 Web 源服务器、2 台代理缓存服务器、1 台负载均衡服务器、1 台 DNS 服务器和 1 台客户端，其拓扑结

构如下图 3.1 所示。

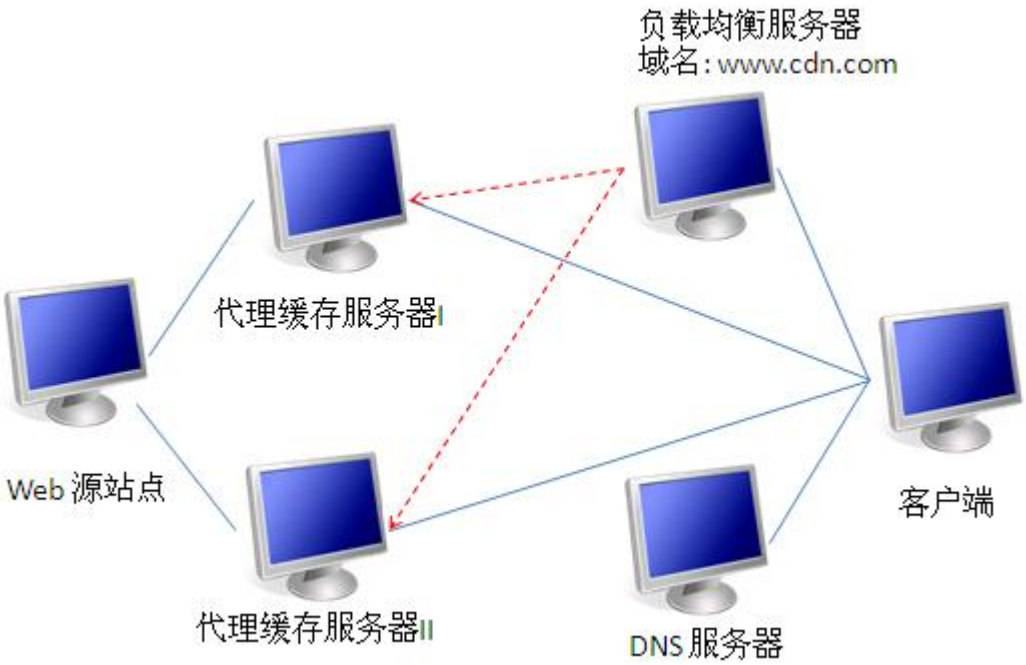

图 3.1 CDN 平台拓扑结构

Web 源服务器：安装了 Apache HTTP 软件，作为存储数据的源头；

代理缓存服务器：安装了 Squid 软件，用来缓存传输的数据；

负载均衡服务器：安装了 Nginx 软件，决定客户端通过哪一个缓存服务器来下载数据；

DNS 服务器：安装了 Bind 软件，可以让客户端通过域名（如 www.cdn.com）访问负载均衡服务器；

客户端：用来测试整个网络的性能。

各节点之间通过 IP 地址连接。当客户端发送一个请求访问负载均衡所对应的域名时，负载均衡通过某种策略（可自己设置）将请求传给代理缓存服务器，缓存如果存在该内容，则将数据包直接发回给客户端；如果不存在，则缓存将请求发给自己的上一级，即 Web 源服务器，最终 Web 源服务器将数据发回给客户端。

3.4 CCN Overlay 实验平台设计

为了能更好地进行对比，我们将 CCN 设计成与 CDN 实验平台一样的拓扑结构。所以 CCN 的实验平台有 5 台机器，包括 1 台 Web 源服务器、2 台中间节点服务器、

1 台负载均衡服务器、和 1 台客户端，其拓扑结构如图 3.2 所示。

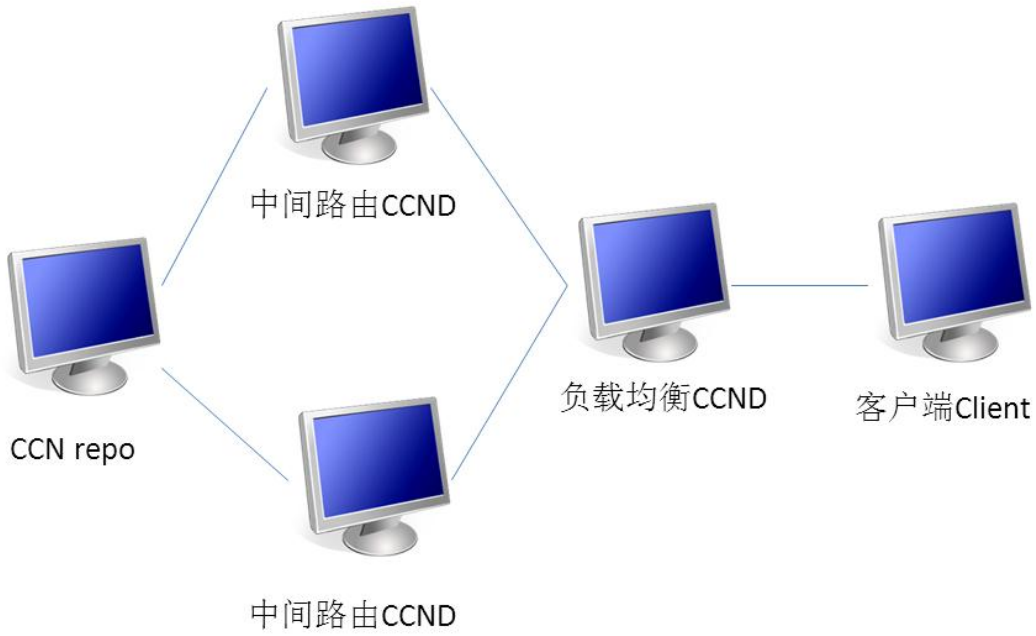

图 3.2 CCN 平台拓扑结构

CCN 网络下的所有机器都安装了 ccnx-0.7.0（CCN 实现的开源代码，2012 年 12 月发布）。

CCN repo：作为存储数据的源头；

中间路由 ccnd：用来缓存数据和作为中间路由传输数据；

负载均衡 ccnd：决定客户端通过哪一个缓存服务器来下载数据，并作为中间路由传输数据；

客户端：用来测试整个网络的性能。

每台服务器通过 IP 和内容命名建立路由表，实现彼此连接。当客户端对负载均衡 ccnd 发送请求时，负载均衡 ccnd 通过判断哪一个中间路 ccnd 由返回数据最快，来决定从哪里转发 CCN repo 的数据，传送数据期间，转发数据的中间路由可能会多次转换。因为每一台机器都是一个 CCN 节点，所以每台机器都可以看到自己是否收到或发出数据包。

3.5 CloudStack 平台搭建

因为进行的实验都是在服务器的多台虚拟机上，无法在一台机器上同时操控管理所有机器，它们之间的来回切换很是繁琐。所以我们希望借助 CloudStack

这个开源平台来实现对实验集群的同时操控，提高工作效率，而且这对未来我们统一维护、管理实验室各服务器都提供了方便。

3.6 实验设计

因为 CDN 和 CCN 的传输原则是不同的：CDN 不改变源数据，直接将数据文件进行传输；而 CCN 是将数据文件首先切分、编码成 4KB 的 XML 数据包，然后再进行传输，客户端再将收到的 XML 数据包解码、重组成正常的数据文件。所以，为了能得到全面的对比结果，本文进行了多种实验进行比较。为了进行文件传输，我们预先将数据储存在 Web 源站点上，客户端发送请求来获取数据（CDN 客户端使用 wget 指令下载数据，CCN 客户端使用 ccngetfile 函数下载数据），并且记录下每次的传输时间。

其中，CCN 的负载均衡传输策略是依次从代理缓存服务器获取数据，如果缓存中没有命中则从源站点获取数据；CDN 的策略是从返回数据包最快的路由节点获取数据。具体实验设计如下。

3.6.1 实验一：传输多个小文件

我们将谷歌的首页文件（62.6KB）作为 CDN 和 CCN 网络下传输的小文件，因为传输单个小文件的时间很短，不便于比较，所以传输的个数分别是 100、200…1100、1200，并记录每次的传输时间。其中，CDN 和 CCN 的缓存均设置为 200MB，已足够缓存全部数据。

3.6.2 实验二：传输单个大文件

分别在 CDN 网络和 CCN 网络下，传输一个大小为 2.2GB 的视频文件。要求客户端一次只请求一个文件进行传输，记录下传输的时间，重复 9 次。其中，CDN 的缓存设为 2.5G；CCN 则取了 2 个实验组，设为 200M 和 2.5G，这样设置我们希望能获得 CCN 网络下除去解码的网络间传输时间。

3.6.3 实验三：传输多个大文件

我们仍然传输相同的大小为 2.2GB 的视频文件。要求客户端每次同时请求多个文件，传输文件个数依次为 2，3，…，9，记录下每次的传输时间。其中，CDN 的缓存设为 2.5G；CCN 同样设为 200M 和 2.5G，我们也是希望能获得 CCN 网络下

除去解码的网络间传输时间。

3.6.4 实验四：限制带宽后的传输单个文件

我们发现在 CCN 网络下传输数据，客户端对数据包的解码、重组时间占据了传输时间的很大比例。所以，如果去掉了 CCN 的编解码机制，那么它的传输效率将会得到很大提高。我们研究了 CCNx 的实现源码，发现它是一个很巨大的工程，编解码机制虽然只是其中的一个部分，但却贯穿了很多地方。删掉它是不太现实的，因此设计了这个实验，希望绕掉解码的过程。

为了使数据包传输时间和数据包解码时间不在同一数量级上，即能忽略掉数据包解码时间，通过软件和防火墙设置限制了 CDN 和 CCN 网络下的带宽，从而延长了网络间传输时间。最终设置了客户端下载速率为 50KB/s，然后传输一个大小为 5MB 的文件，记录下传输时间。

第4章 实验结果与分析

4.1 CloudStack 平台搭建

我们部署了 CloudStack 平台来控制、管理虚拟集群，基本安装步骤如下，

①在一台虚拟机上安装 64 位的 ubuntu 操作系统，并安装 openntpd 时间服务器；

②下载并安装 cloud-client、vhd-util 软件，设置 CloudStack 的用户名和密码；

③安装 Mysql 数据库，配置参数；

④安装 NFS，并配置主存储、辅助存储目录；

⑤配置网络和防火墙的相关参数；

⑥禁用系统默认的 tomcat 服务，启动 cloud-management；

⑦用浏览器登陆（http://166.111.135.119:8080/client）该管理节点，将各种软件所管理的云接入；

详细部署参见附录 B。

图 4.1 和图 4.2 分别显示了进入 CloudStack 后的主界面和管理结构图。

图 4.1 CloudStack 平台主界面

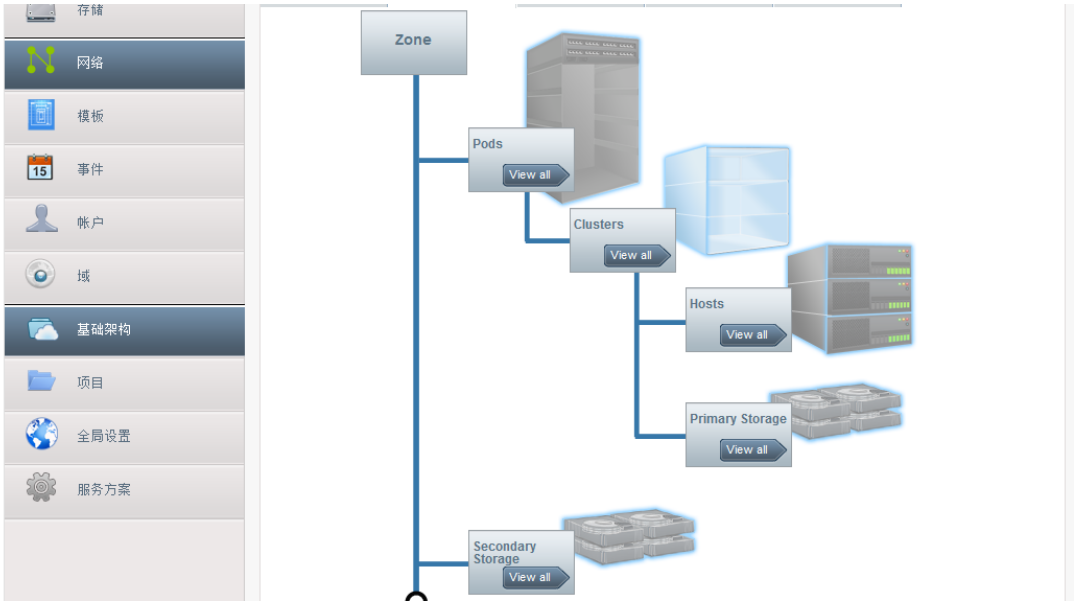

图 4.2 CloudStack 平台管理结构图

4.1.1 NFS 配置

需要配置 NFS 来作为虚拟机的存储卷，NFS 的基本安装步骤如下，

①在一台虚拟机上安装 64 位的 openindiana 系统；

②下载并安装 napp-it 软件，用浏览器登录（如 http://166.111.135.122:81）该管理节点，界面如图 4.3；

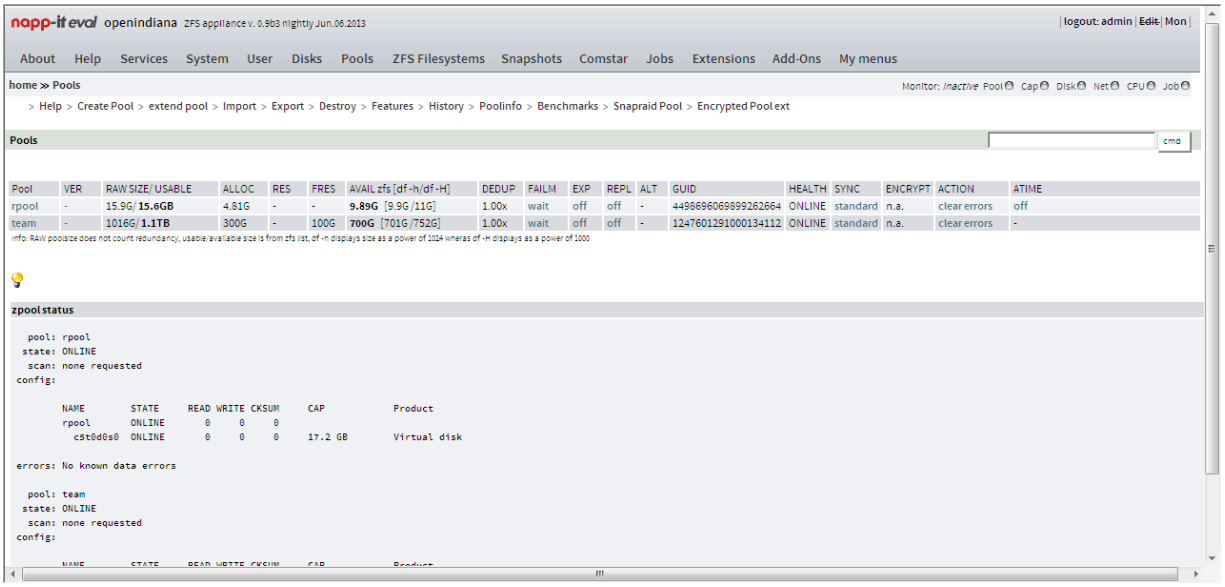

图 4.3 napp-it 管理界面

③配置资源池、文件系统等参数，最终结果见图 4.4；

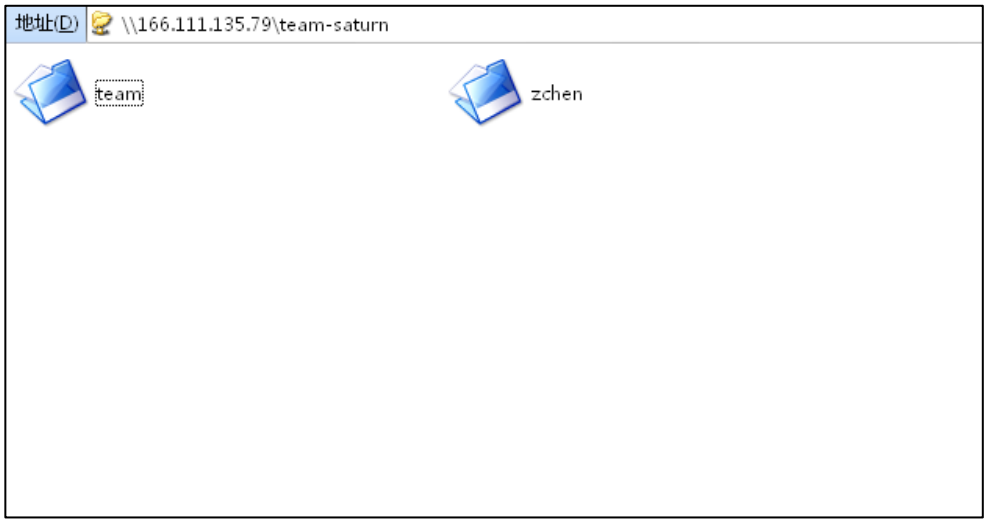

图 4.4 NFS 部署界面

4.2 集群部署开销对比

CDN 网络中的绝大部分节点都是不同的，因此每一个节点的服务器都需要单独配置；而 CCN 网络中包括客户端在内的所有节点都是相同平等的，因此配置好

一台进行迁移复制即可，具体情况参见表 4.1。（详细部署参见附录 B）

表 4.1 部署开销

	CDN	CCN
命令行/ 行数	>80	<20

4.3 实验一：传输多个小文件

在这个实验中，连续传输多个小文件（大小：62.6KB），实验结果如图 4.5 所示。其中，图像的横坐标是每次传输的文件个数，纵坐标取的是传输多个文件的总时间的对数。

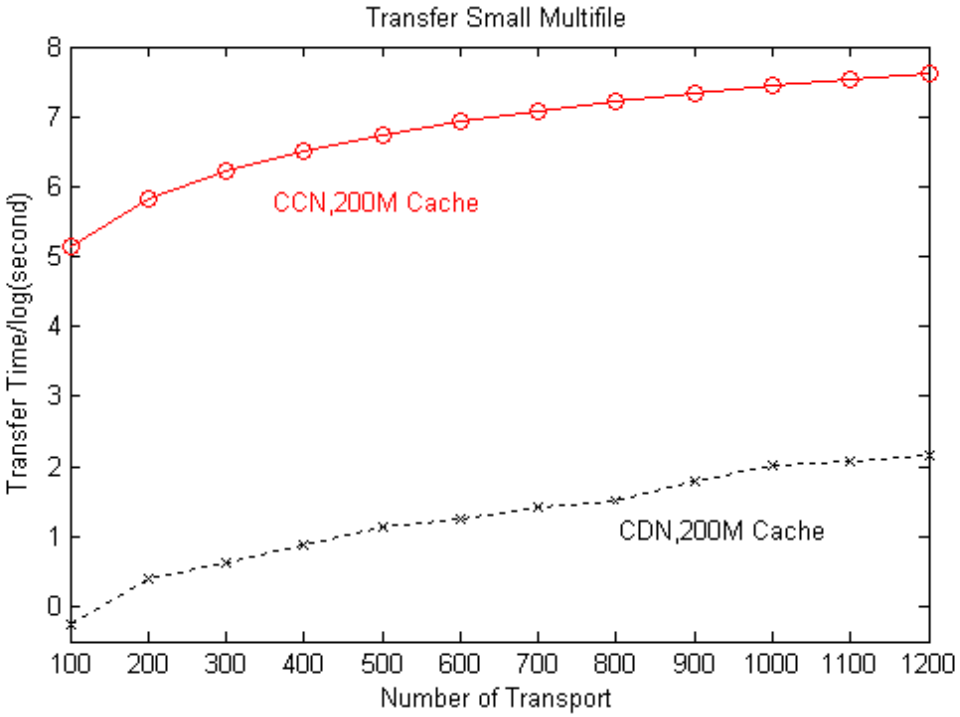

图 4.5 传输多个小文件实验曲线

通过图 4.5 的实验结果，发现 CDN 的传输速度明显快于 CCN 的。于是，我们检查了 CCN 客户端的 CPU 使用情况，发现它的使用率一直维持在 90%以上，而且图中的 CCN 传输曲线接近线性，说明占用 CCN 传输时间的主要部分是数据包的 XML 解码和重组数据过程，而不是网络间的传输过程。

4.4 实验二：传输单个大文件

在本次实验中，传输单个大文件（大小：2.2GB），实验结果见下图 4.6。其中，图像的横坐标是传输文件的个数，纵坐标取的是传输过程的总时间和平均时间的对数。

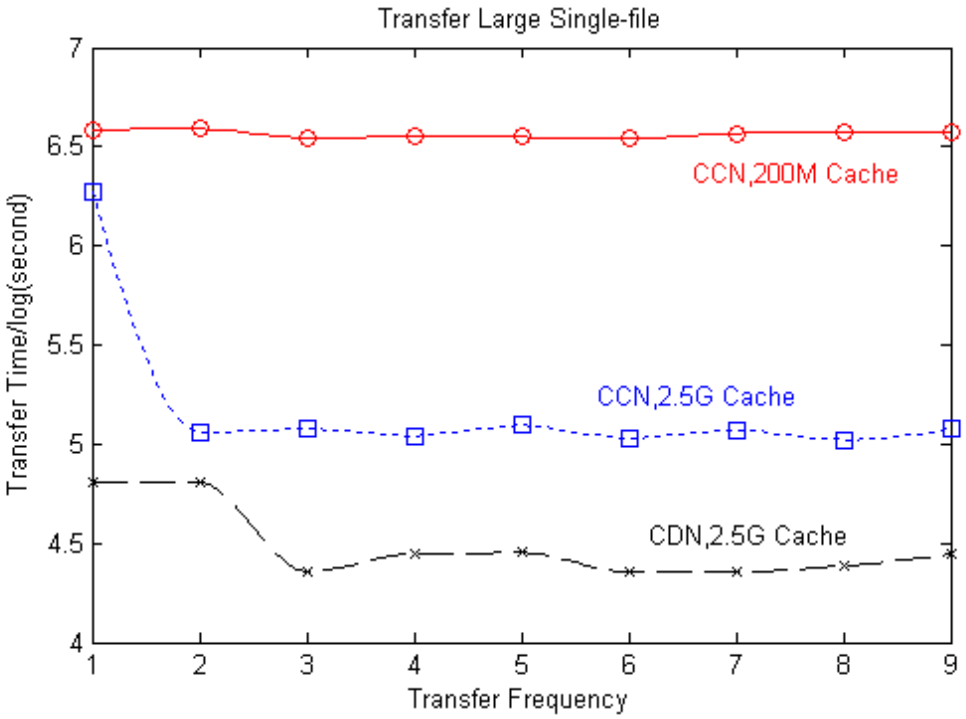

图 4.6 传输单个大文件实验曲线

从图 4.6 中的 CDN 传输曲线可以看出，其第 1 次和第 2 次的传输时间明显长于后面几次。这是 CDN 系统中的负载均衡策略导致的。前两次客户端发送请求后，是从 Web 源站点获取的数据。但是从第 3 次开始，2 个缓存中都存储了该数据，所以客户端开始从代理缓存中下载所需数据，传输时间也大大缩短了。

图 4.6 的 CCN 200MB 缓存曲线表明，客户端下载数据的时间基本都相同。因为 200MB 的缓存远远小于 2.2GB 的源数据，缓存无法存储全部数据，所以客户端每次绝大部分的数据都是从 Web 源站点获取。而对于 CCN 2.5GB 缓存，情况则不同。从它的传输曲线，发现它第一次的传输时间远远长于随后几次。这是因为 CCN 的本地缓存机制，第 1 次请求获取完数据后，CCN 客户端的本地缓存中已经储存了该数据，所以此后客户端获取数据都是从本地缓存中，大大提高了传输效率。对比 CDN 和 CCN 2.5GB 缓存的曲线，它们俩的性能，相对于传输小文件的情况，已经很接近了。

下面再来具体分析一下 CCN 各曲线的时间组成，

$$t_1 = t_3 + t_4 + t_5 \quad (1)$$

$$t_2 = t_6 + t_4 + t_5 \quad (2)$$

其中，t_1是 CCN 200MB 传输时间，t_2是 CCN 2.5GB 传输时间，t_3是网络间传输时间，t_4是本地解码、重组时间，t_5是其它时间，t_6是本地传输时间。

因为CCN 2.5GB缓存下t_6与t_4不在同一数量级上，相对于t_4可忽略不计，所以，我们将这2条曲线做减法，可以粗略得到CCN 单纯 的 网 络 间 传 输 时 间 的一个上界，即$t_3 < t_1 - t_2$ 。如果单纯的对比 CDN 和 CCN 的网络间传输时间，我们发现 CCN 在某些情况下已经优于 CDN 了。

4.5 实验三：传输多个大文件

在本次实验中，同时传输多个大文件（大小：2.2GB），图 4.7 和图 4.8 分别表示传输多个大文件总时间曲线和每个文件的平均时间曲线。

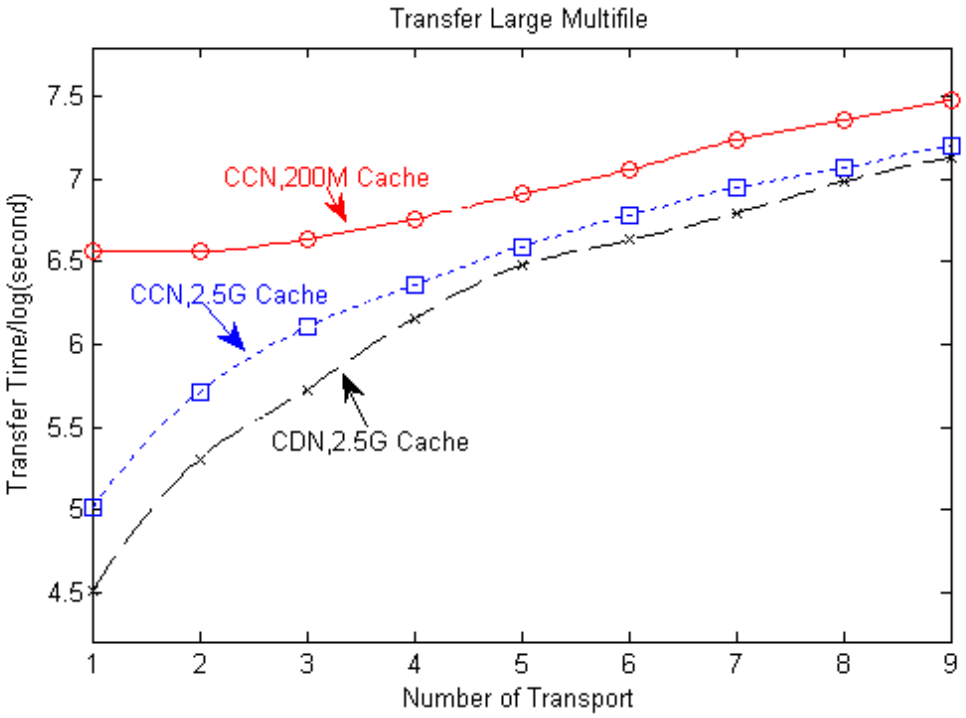

图 4.7 传输多个大文件总时间曲线

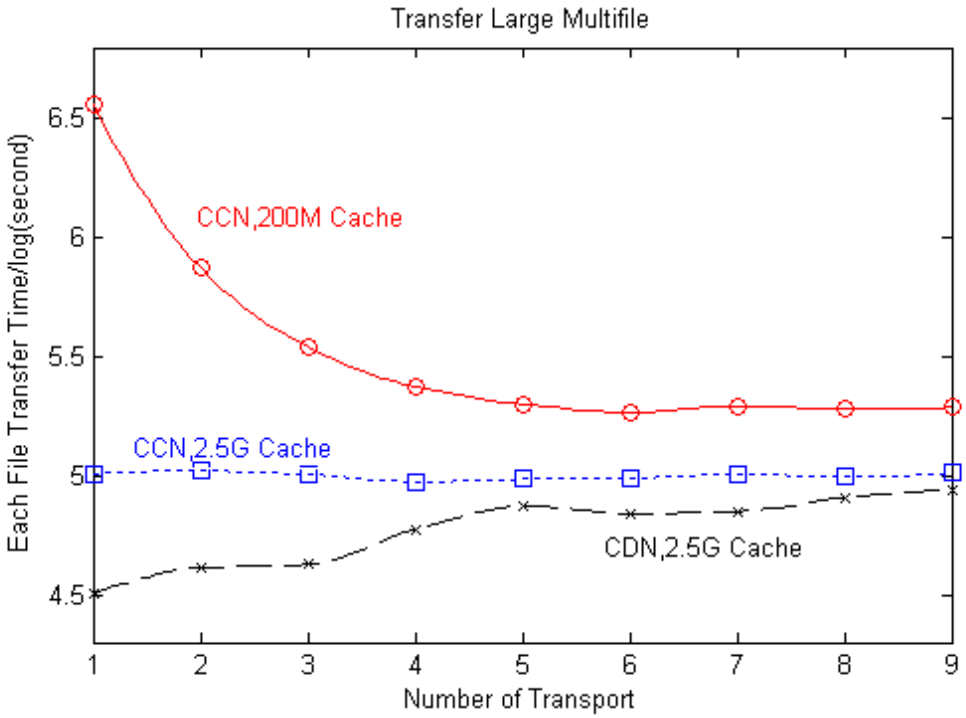

图 4.8 传输多个大文件平均时间曲线

图 4.7 显示了 CDN 和 CCN 同时传输多个文件的总传输时间。虽然每条曲线都是递增的，但是它们的趋势却是不同的。CDN 和 CCN 2.5GB 缓存的加速度是递减的，所以它俩会有更大的优势；而且从具体数据上看，它俩也是非常接近了。

从图 4.8 发现当同时传输多个大文件时，CDN 每个文件的平均传输时间是递增的，CCN 200M 缓存的平均传输时间是递减的，而 CCN 2.5GB 缓存是平稳不变的。与此同时，当传输文件个数多于 4 时，CDN 和 CCN 的传输时间相差并不多。原因是在 CCN 的网络中，每台服务器都是一个 CCN 的节点，包括客户端在内。而 2.5GB 已经足够存储所下载的文件，但 200MB 则显然不够。所以在本地缓存中的数据传输会非常快，这也是 CCN 2.5G 缓存优于 CCN 200M 缓存的原因。至于 CCN 200MB 缓存的每个文件的平均时间会降低的原因是，CCN 网络中的吞吐量会随着传输文件的增多而逐渐达到峰值，平均传输速度会逐渐变快。而 CDN 网络中的平均传输时间之所以会递增，首先是因为 CDN 并没有本地缓存机制，数据都是从上级服务器请求下载的；其次 CDN 在传输过程中并不会改变原始数据，当文件很大时，它会传输一个很大的数据块，尤其当请求增多时，多个大数据块会造成网络拥塞，进而增加了网络时延，降低了平均传输速度。虽然 CCN 的机制上有很大优势，但从实验结果来看，CDN 的传输性能还是优于 CCN，究其原因还是 CCN 对每个数据包的解码、重组过程占据了大量时间。改进甚至去掉该解码算法将是未来提高 CCN

效率的重要方向。

4.6 实验四：限制带宽后的传输单个文件

为了能更好地对比 CDN 与 CCN 在网络传输间的性能，我们做了这个实验，将网络带宽限制为 50KB/s，传输一个 5MB 的文件，实验结果如表 4.2 所示。

表 4.2 限制带宽后的传输数据

	CDN	*CCN*
正常传输时间/s	<1	<5
限制带宽传输时间/s	100	120

从表 4.2 中的数据可以看出，单纯网络间的传输性能 CDN 与 CCN 是近似的。

4.7 实验总结

本文分别从传输个数、传输文件大小，以及限制带宽后方面进行了多组对比实验，表 4.3 显示了进行全部实验后的对比情况。

表 4.3 CDN 和 CCN 数据传输对比

	CDN	*CCN 200M*	*CCN 2.5G*
多个小文件	明显优	明显劣	明显劣
单个大文件	较优	明显劣	较劣
多个大文件	较优	明显劣	较劣
限制带宽传输	较优	较劣	较劣

结论（1）CCN 2.5GB 缓存与 CDN 性能差不多，但 CDN 在传输小文件和单个大文件时仍表现出很大的优势。

结论（2）如果忽略掉 CCN 的解码、重组数据包时间，即单纯地比较网络中的传输时间，从上面具体的实验数据可以得出，在传输多于 4 个大文件下，CCN 的传输速度是比 CDN 快的，这也是我们所希望看到的。

总而言之，CCN 在性能上仍有很大的改进空间。

第5章 CCN Underlay 网络实现

当前实现的 CCNx 源码是基于 TCP/IP 和 UDP 协议之上的覆盖网络，这是为了 CCN 能更好地利用当前网络所设计。而我们希望能建立一个完全没有 IP 地址限制，通过数据命名来进行通信的真正的 CCN 网络。

5.1 相关技术

5.1.1 以太网帧

在以太网中，一个以太网帧[21]就是一个完整的数据包。表 5.1 显示了以太网帧的结构。一个以太网帧是以 7 个 8 位二进制的同步位和 1 个 8 位二进制的分割位开始，中间是 60 至 1518 位二进制的负载区，然后是 4 字节的校验序列，最后是 12 字节的结束位。

表 5.1 以太网帧结构

标志	同步位	分隔位	负载	校验序列	结束位
长度/字节	7	1	60-1518	4	12

大部分系统默认的以太网帧负载部分的长度最大为 1518 字节，其中一般都用其中的 18 字节作为 MAC 头，还剩下 1500 字节，因此大部分网卡设备默认的最大传输单元长度（MTU）为 1500 字节，如图 5.1 所示。

在本节的实验中，我们是将 CCN 的数据包直接装载到以太网帧的负载部分中。不过，CCNx 在设计中默认的一个数据包长度是 4KB，而普通以太网帧默认的 MTU 一般为 1500 字节，加上原来的 MAC 头地址部分最多可以用到 1518 字节，因此不管是在覆盖网中，还是在我们设计的网络中，一个 CCNx 数据包必然会被切分开来传递。对此我们尝试去修改最大传输单元长度（MTU），而我们也将 MTU 的长度改得足以装载整个 CCNx 数据包。

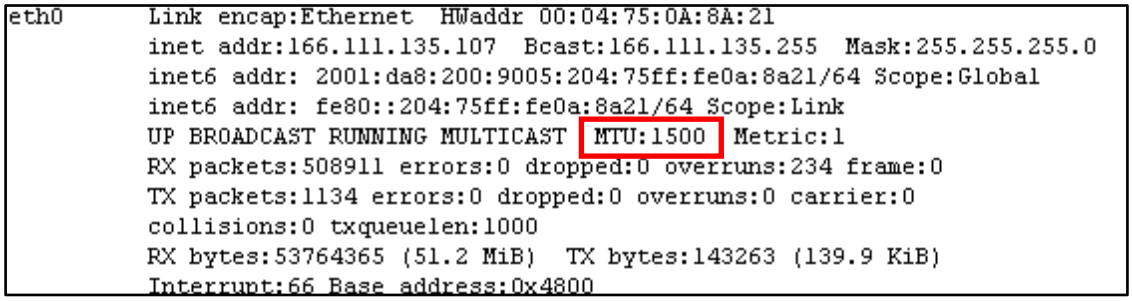

图 5.1 原始网卡的 MTU

最大传输单元长度（MTU）超过 1500 的以太网帧被称为超大以太网帧（Jumbo Frame）[22]。我们将 MTU 最大改到了 7000 字节。下图 5.2 展示了一个网卡被修改成超大以太网帧后的效果。

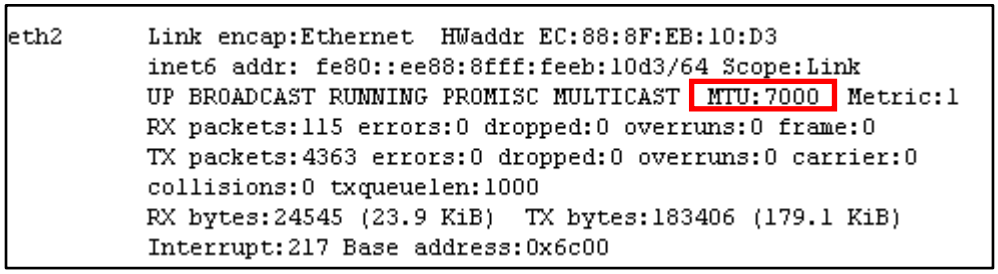

图 5.2 修改后的网卡 MTU

5.1.2 混杂模式

网卡通常有正常模式、混杂模式[23]等。正常模式下，无论是有线网卡还是无线网卡，对于接收到的数据帧，都是先校验 MAC 地址，如果 MAC 地址不是自己的帧、广播帧和组播帧，就会将其丢弃，其它的帧包才会传递给上层程序。混杂模式下，所有经过网卡的帧包，无论什么格式、什么地址的，都会被接收传递给上层程序。因为要建立一个无 IP 地址的网络，已经去掉了 MAC 地址，因此不能让网卡选择丢弃这些包。于是修改网卡为混杂模式，将所有的帧包都收集起来，在上层再去判断哪些是需要的数据包，然后进行取舍。 图 5.3 显示了混杂模式下的网卡。

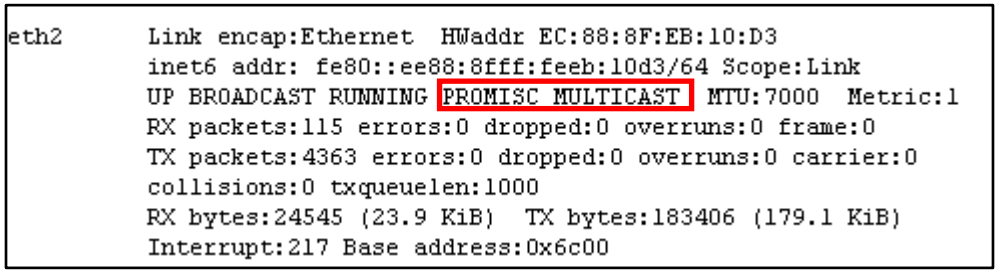

图 5.3 混杂模式下的网卡

5.1.3 原始套接字

原始套接字[24]是网络编程协议栈的一个软件接口，而且允许开发人员对网络协议进行很底层的编程。因为原始套接字可以接收和发送原始的以太网帧，即直接在第二层以太网帧层进行操作，所以使用原始套接字来进行接口的代码编程。

5.2 基本思想

网络间各节点的通信最基础的就是两台机器 host-to-host 的连接，图 5.4 展示了协议架构变化，现在的覆盖网是物理层到以太网层到 TCP/IP 协议层到 CCN 协议层，我们希望改进物理层到以太网帧层到 CCN 协议层到 CCN 数据层的协议栈结构。

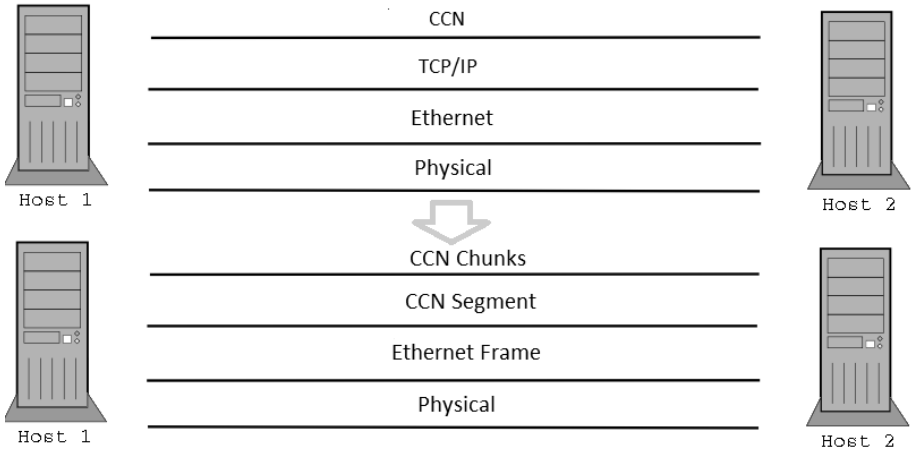

图 5.4 协议架构变化

在设计中，最上层的协议仍使用典型的 CCNx 模型；下面的链路层需要保证 CCNx 的数据包在传输过程中不能被分割，所以更改了以太网帧的 MTU 值；我们将两台机器的网卡通过网线直接相连，并且都去掉了静态 IP 地址，这说明该局域网是完全运行 CCN 协议的；将网卡设置为混杂模式，进而可进行广播和数据包的接收。这些都保证了 CCN 网络的物理链接。

按照图 5.4 的设计完成了 CCN 的物理链接，而使用原始套接字则作为软件的接口。在 Linux 系统下，使用原始套接字来直接控制以太网帧层来进行数据包的收发。此外，还去掉了以太网帧的 MAC 头，这是因为对于 host-to-host 连接的网络，MAC 头是无用的。

最后，还修改了 CCNx 的代码。主要工作是在 CCNx 中增加支持原始套接字的部分，并且将网卡绑定到一些特定的转发接口上去。主要修改的代码部分包括

ccnd，ccndc，ccngetfile 等。

这样，就实现了一个无地址的网络。我们将一个名字为 test 的数据存储在 CCN 的 repository 中，然后让主机 A 通过网卡 1 广播自己有数据 test，主机 B 通过网卡 2 收到该广播，主机 B 则通过 ccngetfile ccnx:/test test 指令请求该数据，最终 A 将数据 test 发给了 B。图 5.4 显示该过程。

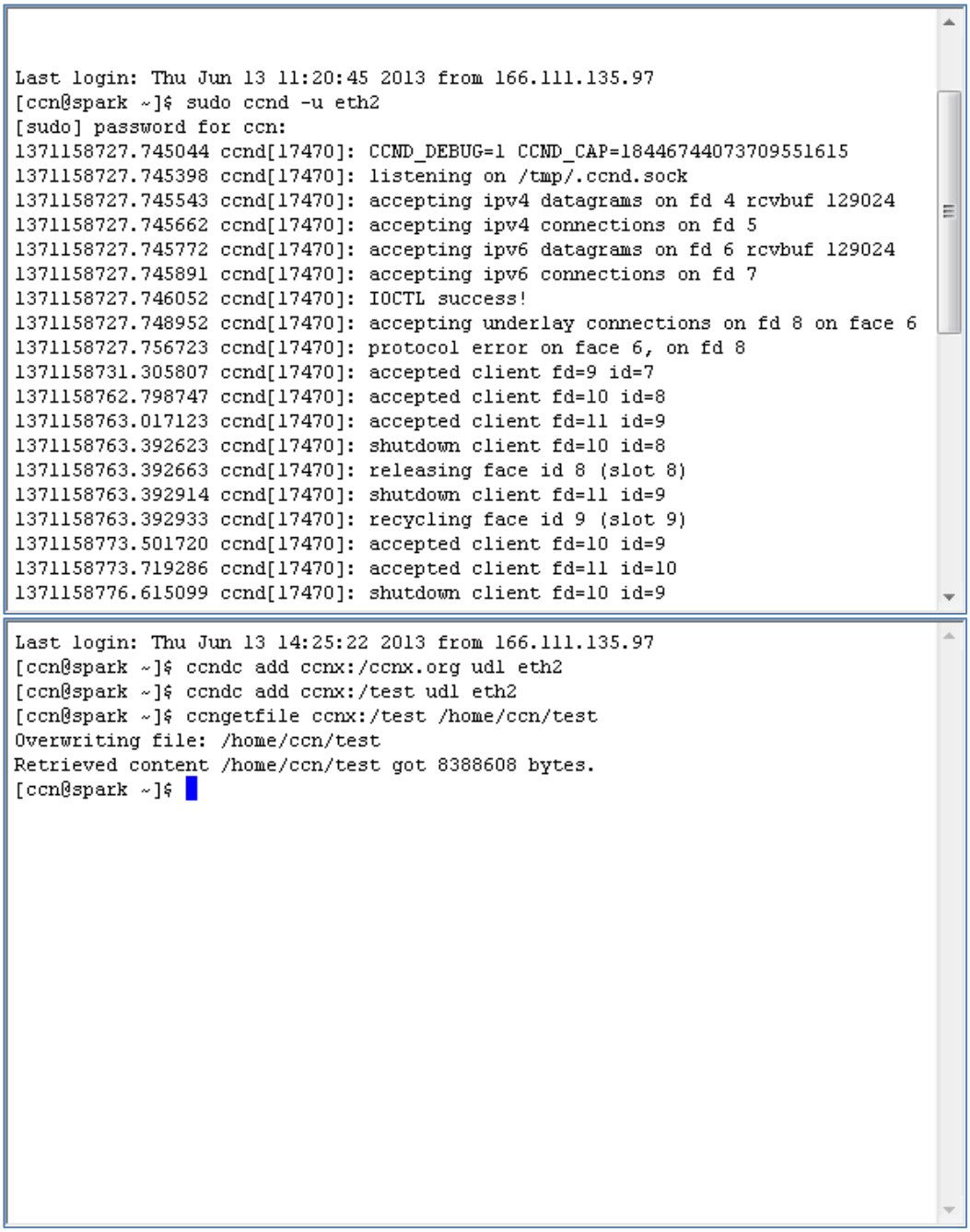

图 5.5 CCNx 收发过程

可以观察到我们部署的传输协议完全独立于 TCP/IP 协议栈，显示出端到端连接部署的成功。

5.3 性能测量

本节将从协议效率和下载时延两个方面，来对比 CCN 覆盖网络和一种基于以太网帧的 CCN 网络性能。

5.3.1 数据传输效率

CCNx 协议中默认的一个数据包大小是 4772 字节（包括有效数据、签名、编码信息等），传输的有效数据为 4096 字节。而传统的以太网帧一次只能传输 1500 字节的内容，因此 CCN 中 4772 字节的数据包要被切分成 4 个以太网帧传输，通过计算可以分别得到 CCN 覆盖网和原始以太网帧的 CCNx 协议的传输效率。

表 5.2 传输效率对比

	传输效率	公式计算
Overlay	82%	$\frac{4096}{4772+54*4}$
Underlay	86%	$\frac{4096}{4772}$

其中，54 字节是一个以太网帧的 TCP 报头长度。

上表是协议效率的实验结果。对于数据传输，CCN 覆盖网的效率大约为 82%，原始以太网帧的 CCNx 协议大约为 86%。这个结果是比较符合理论结果的。理论上，CCN 覆盖网既有 TCP 报头的消耗，又有签名和编码的消耗；而基于原始以太网帧的 CCNx 协议省掉了 TCP 报头，因此效率会变高。

5.3.2 下载时延

由于时间所限，本节所用到的代码并不完善，在数据包传输时仍存在一定的 bug，导致 Underlay 的传输时延普遍偏大，具体结果见表 5.2，

表 5.3 下载时延对比

文件大小/MB		2	8	32	128	256
下载时延/s	Overlay	3.0	8.2	12.3	30.2	52.5
	Underlay	2.1	3.0	4.6	18.6	33.2

5.3.3 CCN Underlay 星形拓扑实验

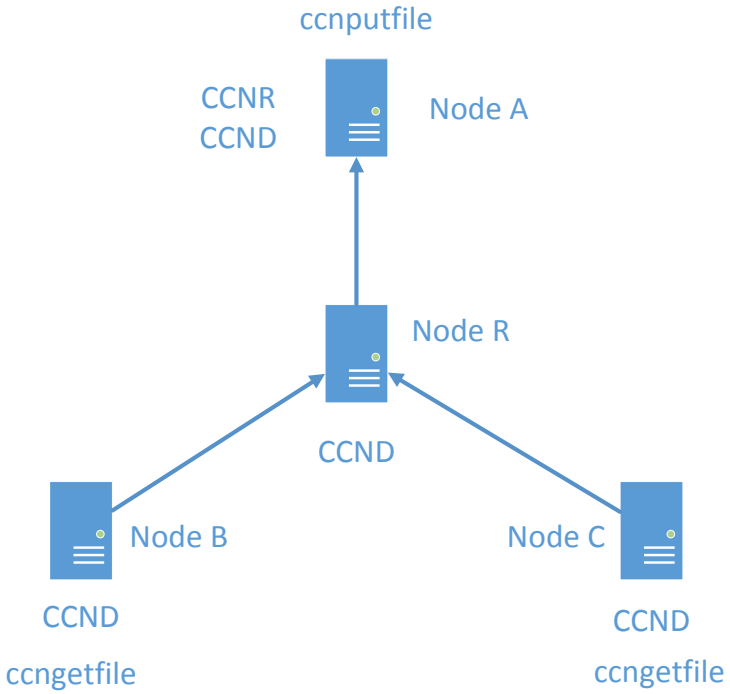

图 5.6 CCN Underlay 网络星形拓扑

CCN 节点 A，B，C 和 R 部署为星形拓扑，B 向 A 以及 C 向 A 均索取大小不一文件，时间上 B 先于 C。实验获取两者下载数据的传输速度如图 5.7 所示。

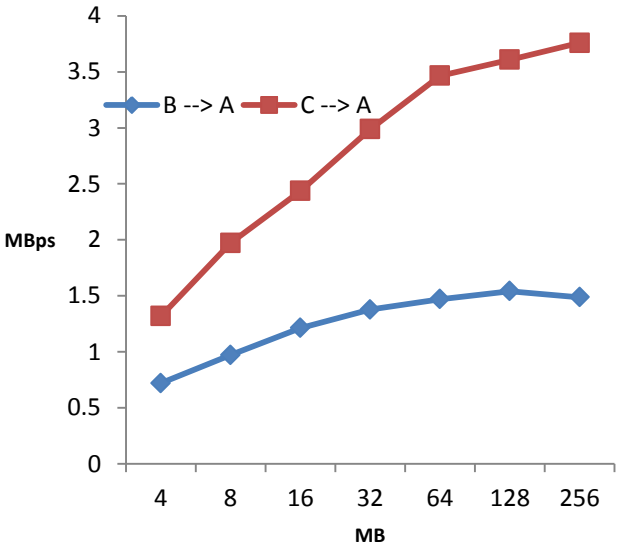

图 5.7 CCN Underlay 网络星形拓扑数据传输性能

从图中可以看到，因为 CCN 节点 R 的缓存的存在，节点 C 下载速率有 2-3 倍的提升。

5.3.4 CCN Underlay 线形拓扑实验

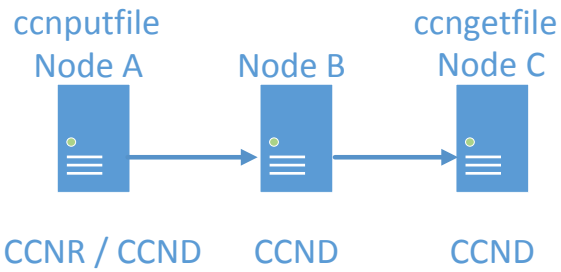

图 5.8 CCN Underlay 网络的线形拓扑

CCN 节点 A，B，C 部署为线形拓扑，B 向 A 以及 C 向 A 均索取大小不一文件，两次实验独立，B 向 A 索取数据后，清空缓存。实验获取两者下载数据的传输速度如图 5.9 所示。

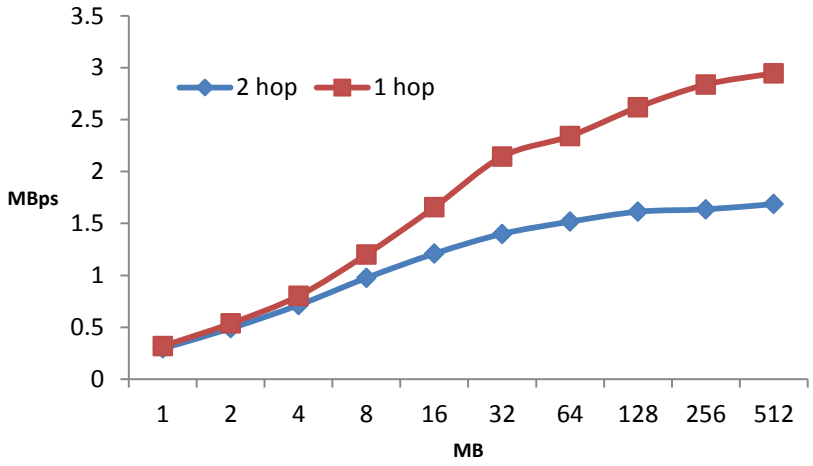

图 5.9 CCN Underlay 网络线形拓扑数据传输性能

从图中可以看到，B 向 A 获取数据为一跳，C 向 A 获取数据为二跳，随着传输数据文件大小增加，性能差异逐渐增大。

第6章 结论与展望

6.1 结论

在当前的互联网形式下，数据传输的高效性仍是目前最重要却又难度很高的需求。CDN 作为一个缓解这种需求的商业架构，在全球得到了广泛部署，所以它不是那么容易改变的。而作为一个全新的网络体系结构，CCN 仍在演化发展中。

在本文中，我们部署了 CloudStack 开源云计算平台来管理虚拟集群，并搭建了 CDN 和 CCN 网络的实验平台，从传输文件个数、大小等多方面，全面系统地比较了 CDN 和 CCN 在传输数据方面的性能。尽管 CCN 会花费大部分时间在数据包的解码、重组上面，但在某些情况下，CCN 的性能已经接近甚至超过了 CDN，比如在传输大文件的情况下。而且，签名验证开销增强了 CCN 网络内容的可信性与安全保护。因此，如果只是单纯地比较网络间的传输时间，那么 CCN 将会在很多情况下优于 CDN。而作为一个专用的内容网络架构，CCN 在安全性、本地缓存机制等方面都有很多优势。

为了进一步提高 CCN 的性能，实现一个完全基于内容的 CCN 网络，我们设计了一套不依赖 TCP/IP 协议，基于以太网帧，完全用命名数据路由的 CCN 网络。而且在实验室搭建了这个网络环境，并且通过多组实验对比，从传输时延、协议效率等方面验证了改良后的网络性能优于原来的 CCN 覆盖网。

6.2 未来展望

虽然在许多情况下，CCN 的性能已经不比 CDN 差，但从实验数据可以发现 CCN 仍有许多改进之处，比如在传输小文件或者传输较少文件情况下，CCN 的性能并不优越。主要原因还是对数据包的解码、重组过程占用了大部分时间。因此，改进 XML 的解码算法将是未来提升 CCN 性能的有效途径。而且在路由的策略选择、内容的多宿主等方面仍有充足的研究空间来提升 CCN 的性能。

我们在实验室搭建了一个不依赖 TCP/IP 协议的 CCN 网络，通过实验验证了其性能确实比原来的 CCN 覆盖网络要有所提高。但从 ccnd 的进程中发现，在数据的传输时有大量的冗余数据包，这在一定程度上限制了 underlay 的性能。未来在程序的调试中需要找到问题所在，提升 underlay 网络性能。

在结构上，实现了 CCN 节点物理直连下的 CCN Underlay 网络，未来我们希望能部署中等规模的 CCN Underlay 网络，并以 Underlay 网络架构为基础，整合本地文件计算与存储，设计一种新型的分布式计算架构。并希望这种计算架构可以实现存储介质上的数据块直接通过网络进行计算。我们未来将设计和完成这样的系统，并进行某些具体应用的大数据分析。

插图索引

表格索引

参考文献

[1] 陈震，曹军威，信息中心网络［M］. 第 1 版. 北京：清华大学出版社，2013

[2] V. Jacobson et al., Networking Named Content, CoNEXT'09, New York, NY, 2009, pp. 1-12.

[3] Buyya, Rajkumar, Mukaddim Pathan, and Athena Vakali. Content delivery networks. Vol. 9. Springer, 2008.

[4] Yuan, Haowei, and Patrick Crowley. Performance Measurement of Name-Centric Content Distribution Methods. Architectures for Networking and Communications Systems (ANCS), 2011 Seventh ACM/IEEE Symposium on. IEEE, 2011.

[5] Project CCNx: http://www.ccnx.org/,2012 年 12 月访问.

[6] 雷葆华，孙颖，王峰，CDN 技术详解［M］. 第 1 版. 北京：电子工业出版社，2012

[7] CloudStack: http://www.cloudstack.org/,2013 年 1 月访问.

[8] Squid: http://www.squid-cache.org/,2012 年 12 月访问.

[9] Apache HTTP: http://httpd.apache.org/,2012 年 12 月访问.

[10] Nginx: http://nginx.org/,2012 年 12 月访问.

[11] James F. Kurose, Keith W. Ross. 计算机网络—自顶向下的网络设计. 陈鸣译.北京:机械工业出版社，2008

[12] T. Koponen et al., A Data-Oriented (and Beyound) Network Architecture, SIGCOMM'07, 2007, pp. 191-92.

[13] M. Gritter and D. R. Cheriton. TRIAD: A New Next-Generation Internet Architecture. http://www-dsg.stanford.edu/triad/, July, 2000.

[14] Lingyun Ruan, Anan Luo, Zhen Chen. TasteBuddy-based Version Selection Strategy for BitTorrent Users against Content Pollution. Proc. of the 3rd International Conference on Communications and Mobile Computing (CMC), 2011.

[15] Reformat Content-Centric Networking with Cryptographic Message Syntax, CCNx Community conference 2012. (Poster)

[16] 闵二龙, et al. "内容中心网络 CCN 研究进展探析." 信息网络安全 2 (2012): 007.

[17] Shuai Ding, Zhen Chen and Zhi Liu, Parallelizing FIB Lookup in Content Centric Networking,the 3rd International Conference on Network and Distributed Computing, ICNDC'2012.

[18] Haopei Wang, Zhen Chen, Feng Xie, Fuye Han, Content Cache Management in Content-Centric Networking, the 3rd International Conference on Network and Distributed Computing, ICNDC'2012.

[19] Lagutin, D., Visala, K, and Tarkoma, S. Publish/Subscribe for Internet: PSIRP Perspective. Towards the Future Internet-Emerging Trends from European Research, 2010. (Valencia FIA book 2010).R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[20] 李 军，陈 震，石 希，ICN 体系结构与技术研究，信息网络安全，2012 年第 4 期

[21] Ethernet frame. http://en.wikipedia.org/wiki/Ethernet_frame, 2013 年 5 月访问.

[22] Jumbo frame. http://en.wikipedia.org/wiki/Jumbo_frame, 2013 年 5 月访问.

[23] Promiscuous mode. http://en.wikipedia.org/wiki/Promiscuo-us_mode, 2013 年 5 月访问.

[24] Raw socket. http://en.wikipedia.org/wiki/Raw_socket, 2013 年 5 月访问.

致谢

首先感谢指导我毕业设计的陈震老师。在完成毕业设计的整个过程中，陈震老师从题目的选择、进度的安排，到最后毕设的完成，都给予我了悉心指导和热心帮助。

同时，感谢曹军威老师对我的毕业设计的指导与建议。虽然与他的交流次数有限，但每一次的交流都让我受益良多。衷心感谢老师们对我的支持和鼓励。

再者，我要感谢陈硕、万宇鑫、石岳等同学。在实验室的日子里，正是有了他们的帮助和陪伴，我的毕业设计才能顺利完成。

最后，感谢清华大学的培养及四年来所有教育、指导、帮助过我的老师、辅导员、同学们！

声　　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____ 日　期：_____

附录 A 外文资料的调研阅读报告或书面翻译

可拓展的 NDN 传输：概念、问题和原则

一、摘要

数据命名网络（NDN）是一种最新提出的网络架构，旨在解决了现有网络的缺点并且充分利用其长处。NDN 对数据包而不是终端进行命名，由此引出了 NDN 架构的多种性质。本文着重介绍 NDN 的数据传递模型。它同 IP 协议中的数据传递模式十分不同，确切来说，NDN 传递是基于可变长度名和可读写的数据层。对可扩展的 NDN 传递的点结构的设计和评估是 NDN 研究的重点之一。本文展示了可扩展 NDN 传递层设计的概念， 课题以及原理： NDN 传输层的核心是快速的名字查询。通过对 NDN 引用实施性能（CCNx）的研究以及对传递架构的简化，我们对三个核心课题进行阐述：1）带有快速更新的字符串匹配，2）可变长度以及无界限名字的最长前缀匹配，3）大型数据流的维护。 另外本文提出了 5 个传递层设计理念来实现 1Gbps 的软件吞吐量以及 10Gbps 的硬件加速。

二、简介

数据命名网络（NDN）是一种最近提出的网络架构，它支持高效的内容分配。NDN 着眼于内容本身而不是内容的位置。每个 NDN 数据包有一个唯一的名字。因为在 NDN 的数据包里没有源和目标的地址，所以数据的传输是要依靠它们的命名。NDN 网络中有两种数据包：兴趣包和数据包。兴趣包包含所取内容的名字， 而数据包同时包含内容与名字，它在所取内容可用的时候被返回。NDN 节点可以抓取数据包，临时存储节点被称为 CS。 大型文件可以在其中被分解、存储和转移。当网络包到达 NDN 节点时，所取内容被缓存，而数据包从节点直接返回。如果名字名字不在节点缓存中，节点记录下内容名称和到达信息在 PIT 中，且依据 FIB 中命名查询结果对网络包进行转发。这样要求 NDN 的转发层支持快速的列表查询和内容插入更新。而 IP 协议相比之下只有一个只读的数据层。。NDN 的路由是结构性管理的，正如 IP 一样。但相对于地址前缀，NDN 路由使用的是命名前缀。总的来说，我们期望 NDN 路由表要比 IP 路由表大得多，因为命名通常要长得多。实际上 NDN 包中的命名有着和 HTTP URL 相似的命名结构，每个名字包含多个组

分。例如，一个 NDN 包命名形如：ndn://wustl.edu/web/research，其中 wustl.edu，web，research 为各组分。

一个 NDN 软件原型 CCNx，已经被 PARC 公司所实现。CCNx 的核心内容是 ccnd 这个后台进程，它实现了数据包的转发层。CCNx 程序可以运行在多个操作系统上，包括 windows，mac，linux，android，甚至作为覆盖网运行在 IP 网络上。NDN 并没有要求 IP 的存在，但有 IP 可以提供全球的联系。

可扩展的转发层是将 NDN 广泛发展的关键，而展现其真实世界中的可行性。本文首先阐述了 NDN 转发层的核心思想，另外解释了其实现的挑战。通过对 CCNx 数据结构的分析和组织，我们确定核心课题是可扩展的转发。 最后我们讨论了设计实施响应系统的理念，这也是未来工作的重点。

NDN 转发层需要支持快速的命名查询，智能化的转发策略和有效地缓存重置策略。本文重在快速命名查询，尤其是转发层高速率基础上的可扩展性。我们测量结果显示 CCNx 实现的吞吐顶峰远比我们预想的最小要求 1Gbps 的链接速率要小。 设计可扩展的转发层的挑战在于可变长度的命名以及包转发的可读写特性。结果，NDN 转发的提升要求高效的算法和数据结构以及可靠的硬件设备。

我们注意到近期的论文已经涉及了相关的主题，比如一般内容中心路由的设计、建立一个内容中心网络的花费估计。到目前为止，仍没有讨论或者调查显示何种数据结构和算法被应用在特定的命名数据传输上。特别地，没有文献关于 CCNx 软件是如何完成的，以及它的使用细节。我们认为 NDN 转发的本质问题需要进一步阐明，并且这些挑战的具体描述将对更广泛的研究有着深远意义。

三、NDN 转发层

在这一部分，我们首先介绍 NDN 转发层所支持的 3 个功能，并且展示转发层在 CCNx 中的使用。我们指出快速命名查找是可扩展 NDN 中的最核心问题。

NDN 转发层支持快速命名查询，智能转发策略和有效地缓存策略。逻辑上讲，NDN 转发层包括 CS（缓存数据包），PIT（存储待定网络请求）和 FIB（存储转发规则）。

1）快速命名查询。由于 NDN 包不像 IP 包那样包括源地址和目的地址， NDN 包根据内容命名进行查询之后转发。NDN 姓名查询可以发生在 PIT，CS 和 FIB，这涉及到最长前缀查询和带有快速更新的精确字符匹配。

2）智能转发策略。NDN 允许每个包包含多个出接口。转发策略选择最高效的

接口，不仅利用 FIB 查询结果，而且考虑网络环境。

3）高效的缓存策略。CS 是负责 NDN 内容分配的组件。高效缓存重置策略可以改善缓存率从而进一步提高内容分布的性能。

NDN 转发层决定每一个收到的数据包要做什么。事实上，NDN 转发层根据功能可以被分成多层。策略层选择转发策略，并且影响转发决定。数据转发层则进行数据包转发，等待兴趣包控制和缓存内容，并且被策略层控制。传输层控制网络的联系，并作为一个借口在 IP 网络上传输数据。

可扩展的 NDN 转发层每秒必须处理数以万计的网络包和数据包，来支持 10Gbps 的链接速率。另外，为了使缓存更有效， 缓存的数据包数目不可避免的很大，这使得 CS 命名查询更加耗时。这里的时间消耗会影响最大数据速率。 智能转发策略和有效的缓存策略可以改善NDN内容分配的性能,但不能直接影响NDN的可扩展性。所以，快速命名查找是使得 NDN 转发可扩展的关键问题。

四、CCNx 的性能研究

了解最新 NDN 原型和它的实际限制是很重要的。在这一部分，我们展示对 CCNx 性能的初步评价。我们的测试结果显示了最新的 CCNx 不能满足 1Gbps 网络的要求。在分析完 CCNx 的峰值吞吐量，我们认为影响其性能的原因主要有两个：工程原因和方法原因。

我们的性能研究是在 ONL（Open Network Laboratory）中进行的。CCNx 的软件我们选取的是 ccnx-0.4.0。它的核心部分 ccnd 被配置了所有默认的环境变量。内容存储的大小，即数据包可以被缓存在路由上的数量将影响整个网络的性能。这个大小默认是 50000。CCNx 支持 TCP 和 UDP，但在我们的实验中只是用了 TCP。我们研究了数据包的负载大小对吞吐量的影响。为了生成 CCNx 的传输，内部函数 ccncatchunks2 和我们编的 ccndelphi 程序分别作为客户端和服务端。Ccncatchunks2 函数生成一系列的兴趣包来获取一个大的文件。生成的兴趣包从名字 ccnx：/URI/0，后面是数据的索引。它获取下一块数据通过增加索引数。Ccndelphi 程序生成随即负载的数据包，并且被设计成尽可能快地返回数据包。

当 CCNx 的路由的 CPU 被完全使用时，CCNx 的吞吐量就达到了峰值。一般地，客户端通过中间路由发送兴趣包到源端，然后源端会通过中间路由返回数据包。为了使路由饱和，我们用了多个客户端发送兴趣请求，并且多个源服务器会生成数据包。

CCNx 路由的吞吐量包括 2 个数值，外出的吞吐量，用 Out 表示，和进入的吞吐量，用 In 表示。我们区分这两种吞吐量，因为它们不一定是相同的，不像 IP 网络中，进出的吞吐量都是很接近的。我们每隔一秒同时记录了进出的吞吐量。对于每一个实验设置，我们选取最高的 20 个吞吐量数值求出它们的平均峰值，并用 90%作为置信区间。

我们使用 Gprof 来描绘 ccnd 这个进程。我们进行了 3 次实验并且列出了耗时最多的 10 个过程。出乎意料的是，超过 60%的时间都被花在了命名解码上面。根据更进一步地研究发现，最初的 CCNx 实现选择存储编码后的内容。此外，CSL 是缓存的一个索引，不是存储解码数据包本身。

根据以上的分析，有 2 个方面的问题限制了 CCNx 软件。第一个是工程问题，比如名字的解码。另一个是方法问题，比如名字的查找。尽管工程问题有更大的影响，我们认为学术上的研究应更多的集中在方法问题上面。工程问题与设计决策密切相关，而解决方法问题的关键是实现快速的命名查找。

Function Name	Percentage(%)
ccn_skeleton_decode	35.46
ccn_compare_names	12.53
ccn_buf_match_dtag	8.95
ccn_buf_advance	6.64
ccn_buf_match_blob	5.07
ccn_buf_decoder_start_at_components	4.94
ccn_buf_match_some_blob	3.32
content_skiplist_findbefore	3.06
ccn_buf_check_close	2.34
hashtb_seek	1.36
Other functions	15.7
Sum	100

五、可扩展 NDN 转发的课题

在这部分，我们讨论可扩展 NDN 转发层设计的核心问题。我们通过对简化的工作任务流进行分析，并且同 IP 转发进行比较，确定了可扩展 NDN 网络中三个最为重要的课题。

1）数据结构与工作任务流。

NDN 转发节点有三个组件：CS，PIT 和 FIB。图 5 显示了实现这三种组件的数据结构。

逻辑 FIB 和 PIT 共享一个 Hash 表 NPHT,它负责存储 PE 和 FIE。NPHT 中每个桶中存有只想 PE 和 FI 的指针，而 PE 和 FIE 两种数据结构分别负责存储

待处理的网络信息和转发信息。 PHT 表的 key 值由 nonc 域决定，为每个网络包分配唯一键值。

对 CS 来说,每个数据包被分配一个单独的累计值,随着新入的数据包而增加。缓存的数据包存入名为 CA 的数组，并以累计值作为序号，正如滑动窗的数据结构。 另外两种数据结构对其加以改善：CHT（以数据包全名作为键值）和 CSL（跳转表的标准实现，支持内容顺序的查询）

工作任务流方面，网络包和数据包在 NDN 中的处理方式是不同的。图 6 显示了 CCNx 实现中的网络包处理方式。当网络包到达时，它以 2 进制方式进行解码和编译。之后 PHT 进行精确字符串匹配保证这不是本身转发的包。旧的网络包被弃置，而新的网络包收到 NPHT 的检查（前缀搜索），保证所有前缀都在 NPHT 中出现，或者新的条目被插入到 NPHT 中。实际上，CCNx 对所有前缀进行核查，以便最大化利用网络包中的信息，所以一个数据包会消耗多个网络包，同时会消耗更多的时间。如果所有前缀检查不是必须的要素，那么我们可以将其简化为一个精确字符串匹配的问题，正如我们之后在简化工作流设计中做的那样。

数据包的处理在图 8 中得以显示。数据包到达后同样被解码和编译，之后精确字符串匹配由 CHT 完成并保证其在 CS 中没有出现，并且以数据包全名作为键值进行查询。如果包没有重复，则将该名插入 CHT 和 CSL，将内容存入 CS。

2）简化数据结构及工作流

CCNx 是 NDN 转发节点的一个模型，但其中一些设计不是 NDN 必须的。图 9 显示 CCNx 节点实现一机 NDN 转发层的概念图。NHPT 在 CCNx 中被用于 FIB 和 PIT 存储前缀的缓存站。但 FIB 中支持最长前缀匹配和 PIT 中的精确字符串匹配就可以满足 NDN 转发的核心要求。

所以，简化后工作流如图 10（网络包）和图 11（数据包）。对新进的网络包，转发层负责检查重复情况，并且依据 PIT 的精确字符串匹配避免重复，然后利用 CS 查找可能的匹配内容。如果内容没有找到，FIB 执行最长匹配前缀匹配确定转发信息，网络包被转发；如果内容被找到，网络包被弃置并且从 PIT 删除。这基本上同图 6 显示得 CCNx 的工作流相同，除了这里不需要 NPHT。每个进入的数据包在精确字符串匹配后被存入 CS,对应网络包被弃置。事实上如果没有 NPHT，工作流过程会更好理解。

简化后的工作流包含了 NDN 转发机制的所有核心问题。基于此的高效数据结构和算法可以用来提升转发效能。特别的，降低重复性工作流对流水线架构的硬件设计尤为重要。

3）待解决的核心问题

同 IP 转发相比，NDN 转发更复杂也更有挑战性。表 2 列举了 IP 与 NDN 的区别。我们的目标是为了达到 1Gbps 的软件转发速率和 10Gbps 的硬件加速。为了达到这个目标，我们确定了三个 NDN 转发中有待解决的核心问题：

a）带有快速更新的精确字符串匹配。NDN 转发层中，带有快速更新的精确字符串匹配由 PIT 或者 CS 查询实现。最坏情况下，每个包都需要更新；

b）可变长度和无界限命名的最长前缀匹配（LPM）。尽管 LPM 在 IP 查询和 URL 过滤中被广泛研究，但现有的 IP 查询方法并不适用于 NDN：NDN 包拥有可变长而且无界限的命名，而 IP 包只有固定 32bit 长的地址。大多数提出的 IP LPM 方案在处理长键值是会变慢，且消耗大量内存资源；

c）大流量维护。流量维护是实现 PIT 的另一种方式。在硬件设计中，实现指针结构和动态 PIT 效率很低。取而代之，我们选择维护（名字，输入接口 ID，输出接口 ID）的数据对而实现转发层的功能。流量维护和 IP 网络中基于流量的监控很相似，不同的是 NDN 包需要处理可变长的命名，而不是固定长度的数组。这也导致了流量表会变得很大，是的维护工作变得很有挑战。

六、 设计原理

对于那些大规模的 NDN 部署，我们最低的目标是用软件实现 1Gbps 的转发性能，硬件实现 10Gbps 的转发性能。为了有效地解决上面论述的问题，我们提供几种设计思路和确定潜在的机会来达到优化效果。

1）常数时间复杂度。快速命名查找的目标在于实现可变长度命名的常数查询时间。鉴于命名长度不受限制，这个目标理论上不可行。但实际中多数命名很可能落入同一个网络包中。

NDN 转发中，长命名需要更多的处理时间。我们测量了 CCNx 路由中命名成分的数量对其性能的影响，并对此行为作出实验性的验证。实验环境的设置、系统拓扑结构和部分 III 相同。我们设置数据包的负载大小为 1024 字节。这个实验和之前的不同之处在于请求的名字更长。我们改变包命名的内容数量，测量的实验结果在图 12 显示。

图 12 显示，正如我们预料的，CCNx 路由吞吐随着命名组分的增多而降低。尤其是，当命名部分的数量达到 32 时，数据的速率接近 40Mbps，这远远低于 1Gbps 的数据包处理目标。

为了实现常数查询时间的机制，一个可能的设计方向是在每个 NDN 节点内将长命名映像为短命名。

2）优化 URL 格式

当设计算法时，我们可以利用 HTTP URL 类似的命名规范在 NDN 的命名机制里。其次我们探讨最优化的 URL 命名格式。其中，URL 的两个关键格式可以被利用：基于组件的字符串匹配（而不是基于字符），和 URL 特征。大部分的 URLs 都是少于 30 个命名组成。这这些的指导下，我们可以设计实现能够快速不断地查找命名组成少于 30 个的潜在目标的结构。

3）支持快速更新的简化数据结构

扩充的网络包需要被插入至 PIT 中，从而当对应数据包返回时，它可以被正确的传送至请求方终端。之后，PIT 中的记录被删除。最坏情况下，每次包的到达都会一起更新，而数据层中数据结构的更新没有被研究过。我们认为使用更简单的数据结构是通向高性能的唯一途径，因为复杂的数据机构会导致过多的处理时间。这些数据结构包括 Hash 表，d 坐 Hash 标和计数 bloom 滤镜。

4）高效的包编码和解码算法

根据我们的 CCNx 实验结果，很大一部分时间消耗在包解码上，这在高性能路由设计中并不多见。CCNx 引入了复杂的 XML 格式进行包编码，尽管它的灵活性利于网络架构和协议设计，但它会使整个数据层变慢。两种方法可以解决这个问题：开发高效快速的 XML 解码算法，或者定义新的包格式以实现快速包编码和解码。举个例子，一个简单有效的数据包格式应该和 IP 的数据包比较相像，每一个数据包的字段有固定的长度。这些字段可以存储固定的值或者数据包的其它信息，如标志等等。

5）多元化内容存储策略

这里 CS 不仅应支持数据包精确缓存和分配，而且支持模糊匹配的内容。举个例子，一个兴趣包可以运输额外的标志去告诉 CCNx 路由来回应一个内容的最新版本,而不用从源端产生一个回应。为了使边缘路由器的性能达到 1Gbps 的范围，这个特征应该被支持。所以，我们认为设计中应保证简洁，限制 CS 匹配为硬件加速的精确匹配。

七、结论

本为介绍了 NDN 转发层以及其 CCNx 实现，并且指出了实现可扩展 NDN 转

发层的一个发展方向，即简化数据结构。本文提出了一种数据层组成方案，并且指出 NDN 转发机制中的 3 个关键课题和 5 个基本原理，旨在引导未来可扩展 NDN 传输网络的研究方向。

参考文献

[1] The NDN project team. Named Data Networking (NDN) Project. In PARC Technical Report NDN-0001, October 2010.

[2] Van Jacobson et al.; Networking Named Content, In Proc. of CoNEXT2009, Rome, December, 2009.

[3] CCNx: http://www.ccnx.org/.

[4] Somaya Arianfar; Pekka Nikander; Jorg Ott; On Content-Centric Router Design and Implications. In Proc. of ACM ReArch 2010, November 30,2010.

[5] Diego Perino; Matteo Varvello; A Reality Check for Content Centric Networking. In Proc. of ACM ICN 1011, August 19, 2011, pp: 44-49.

[6] Giovanna Carofiglio; Massimo Gallo; Luca Muscariello; Bandwidth and Storage Sharing Performance in Information Centric Networking. In Proc.of ACM ICN 2011, August 19, 2011, pp: 26-31.

[7] Jiachen Chen; Mayutan Arumaithurai; Lei Jiao; Xiaoming Fu;K.K.Ramakrishnan; COPSS: An Efficient Content Oriented Pub-lish/Subscribe System. In Proc. of ANCS 2011, Brooklyn, New York.

[8] William Pugh; Skip Lists: A Probabilistic Alternative to Balanced Trees;Commun. ACM, Vol 33, Issue 6, June, 1990, pp: 668-676.

[9] John DeHart et al.; The Open Network Laboratory; In Proc. of the 37th SIGCSE Technical Symposium on Computer Science Education, 2006.

[10] Charlie Wiseman et al.; A Remotely Accessible Network Processor-Based Router for Network Experimentation; In Proc. of 4th ACM/IEEE ANCS, 2008.

[11] Graham, Susan L. and Kessler, Peter B. and McKusick, Marshall K.;Gprof: A Call Graph Execution Profiler; SIGPLAN, April, 2004.

[12] Nan Hua; Haoyu Song; Lakshman, T.V.; Variable-Stride Multi-Pattern Matching For Scalable Deep Packet Inspection; In Proc. of IEEE INFO-COM 2009, April 19-25 2009, pp: 415-423.

[13] Tian Song;Wei Zhang; Dongsheng Wang; Yibo Xue; A memory efficient multiple pattern matching architecture for network security. In Proc. Of IEEE INFOCOM 2008, April

13-18 2008, pp: 166-170.

[14] Derek Pao; Xing Wang; Xiaoran Wang; Cong Cao; Yuesheng Zhu;String Searching Engine for Virus Scanning; In IEEE Transactions on Computers, Nov. 2011, Volume: 60 Issue:11, pp: 1596-1609.

[15] Sailesh Kumar; Patrick Crowley; Segmented Hash: An Efficient Hash Table Implementation for High-Performance Networking Subsystems;In Proceedings of the 2005 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS). Princeton, N.J.October, 2005.

[16] Yi-Hsuan Feng; Nen-Fu Huang; Chia-Hsiang Chen; An Efficient Caching Mechanism for Network-Based URL Filtering by Multi-Level Counting Bloom Filters. In Proc. of IEEE International Conference on Communications (ICC), 5-9 June 2011.

[17] Zhou Zhou; Tian Song; Yunde Jia; A High-Performance URL Lookup Engine for URL Filtering Systems. In Proc. of IEEE International Conference on Communications (ICC), 23-28 May 2010.

[18] Haowei Yuan; Benjamin Wun; Patrick Crowley; Software-based imple-mentations of updateable data structures for high-speed URL matching. In Proc. of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS ' 10), October 2010.

[19] Andrei Broder; Michael Mitzenmacher; Using multiple hash functions to improve IP lookups; In Proc. of IEEE INFOCOM 2001, April 22-26 2001, pp: 1454-1463.

[20] Andrei Broder; Michael Mitzenmacher; Network Applications of Bloom Filters: A Survey; Internet Mathematics 1 (4) 2005. pp: 485-509.

附录 B 相关程序代码

1. CDN 的 Web 源服务器：

   cd /usr/local/src/

   wget http://archive.apache.org/dist/httpd/httpd-2.2.21.tar.gz

   tar -zxvf httpd-2.2.21.tar.gz

   cd httpd-2.2.21

   ./configure -prefix =/usr/local/apache

   make

   make install

   yum install gxx

   cd /usr/local/apache/bin

   ./apachectl start

2. Squid 代理缓存服务器

   cd /usr/local/src/

   wget

   http://www.squid-cache.org/Versions/v3/3.1/squid-3.1.16.tar.gz

   tar -zxvf squid-3.1.16.tar.gz

   cd squid-3.1.16

   ./configure -prefix =/usr/local/squid

   make

   make install

   yum groupinstall “development tools”

   cd /usr/local/squid/sbin

   ./squid -z

   Chmod 777 /usr/local/squid/var/log

   ./squid

   cd /usr/local/squid/etc

   vi squid.conf

   更改配置文件

3. Bind 域名服务器

   cd /usr/local/src

wget http://ftp.isc.org/isc/bind9/9.8.1-P1/bind-9.8.1-P1.tar.gz

tar -zxvf bind-9.8.1-P1.tar.gz

cd bind-9.8.1-P1

./configure -prefix =/usr/local/bind

make

make install

cd /usr/local/bind/etc

/usr/local/bind/sbin/rndc-confgen > rndc.conf

cat rndc.conf > rndc.key

chmod 777 /usr/local/bind/var

tail -n10 rndc.conf | head -n9 | sed -e s/#\ //g > named.conf

vi named.conf

更改配置文件

cd /usr/local/bind/sbin

./named-checkconf

Cd /usr/local/bind/var

vi cdn.com.zone

更改配置文件

cd /usr/local/bind/sbin

./named-checkzone cdn.com /usr/local/bind/var/cdn.com.zone

cd /usr/local/bind/sbin

./named

4. Nginx 负载均衡服务器

cd /usr/local/src

wget http://nginx.org/download/nginx-1.1.10.tar.gz

tar -zxvf nginx-1.1.10.tar.gz

cd nginx-1.1.10

./configure -prefix =/usr/local/nginx

make

make install

还需要安装 PCRE，zlib 库

cd /usr/local/nginx/sbin

./nginx

Curl -i http://localhost

cd /usr/local/nginx/conf

vi nginx.conf

更改配置文件

cd /usr/local/nginx/sbin

./nginx -t

./nginx -s reload

5. CCN 节点

apt-get install git-core python-dev libssl-dev libpcap-dev libexpat1-dev athena-jot libcrypto libxml2

下载解压 ccnx，jdk，ant

wget http://...

wget http://...

wget http://...

tar -zxvf …

tar -zxvf …

tar -zxvf …

配置环境变量

vi /etc/profile

cd /../ccnx..

./configure

make

make install

ccndstart

6. CloudStack 配置安装

配置 CloudStack 官方源

sudo vim /etc/apt/sources.list.d/cloudstack.list

deb http://cloudstack.apt-get.eu/ubuntu precise 4.0

配置 CloudStack 官方源证书

wget -O - http://cloudstack.apt-get.eu/release.asc| sudo apt-key add

更新系统的安装源

sudo apt-get update

安装时间服务器

sudo apt-get install openntpd

安装 CloudStack Management Server

sudo apt-get install cloud-client

将用户 cloud 加入到 sudo 用户组

sudo adduser cloud sudo

配置 sudo 用户组免密码切换 //同样是为了解决和上面相同的权限问题

sudo visudo

%sudo ALL=(ALL:ALL) NOPASSWD:ALL

下载 vhd-util

sudo wget http://download.cloud.com.s3.amazonaws.com/tools/vhd-util

sudo mv vhd-util
/usr/lib/cloud/common/scripts/vm/hypervisor/xenserver/

安装配置 MySQL 数据库

sudo apt-get install mysql-server

修改 MySQL 配置文件参数

sudo vim /etc/mysql/my.cnf

sudo service mysql restart

初始化数据库 cloud

sudo cloud-setup-databases cloud:cloudstack@localhost
--deploy-as=root:cloudstack -e file -m cloudstack -k cloudstack

安装 NFS

sudo apt-get install nfs-common nfs-kernel-server

创建目录

sudo mkdir -p /export/primary

sudo mkdir -p /export/secondary

编辑 NFS 配置文件

sudo vim /etc/exports

刷新配置

sudo exportfs -a

挂载测试 NFS 共享

```
sudo mkdir /mnt/primary
sudo mount -t nfs 10.6.203.10:/export/primary /mnt/primary
sudo mkdir /mnt/secondary
sudo mount -t nfs 10.6.203.10:/export/secondary /mnt/secondary
df -h
sudo vim /etc/fstab
```

增加以下内容

```
1 10.6.203.10:/export/primary /mnt/primary       nfs rw,tcp,intr 0 1
2 10.6.203.10:/export/secondary /mnt/secondary       nfs rw,tcp,intr 0
1
```

准备 System VM Template

```
sudo
/usr/lib/cloud/common/scripts/storage/secondary/cloud-install-sys-
tmplt -m /mnt/secondary \
-u
http://download.cloud.com/templates/acton/acton-systemvm-02062012.
qcow2.bz2 -h kvm -F
```

安装配置 Agent

```
sudo apt-get install cloud-agent
```

安装配置 libvirt

```
sudo vim /etc/libvirt/libvirtd.conf
sudo ufw allow proto tcp from any to any port 22
sudo ufw allow proto tcp from any to any port 80
sudo ufw allow proto tcp from any to any port 1798
sudo ufw allow proto tcp from any to any port 16509
sudo ufw allow proto tcp from any to any port 5900:6100
sudo ufw allow proto tcp from any to any port 49152:49216
sudo /etc/init.d/tomcat6 stop
sudo update-rc.d -f tomcat6 remove
sudo /etc/init.d/cloud-management restart
```

在学期间参加课题的研究成果

[1] Ge Ma, Zhen Chen, Shuo Chen, et al., Performance Comparison on CDN and CCN-poster, CCNxCon, 2013.（Under Review）

[2]陈震，马戈，曹军威等,内容中心网络对比分析及其原生实现的研究,计算机学报，2013.（审核中）

[2]陈震，陈硕，马戈，一种内容中心网络的底层实现方法专利申请.（审核中）