

命名数据网 (NDN) 可用性关键问题研究

(申请清华大学工学博士学位论文)

培养单位: 计算机科学与技术系
学 科: 计算机科学与技术
研 究 生: 蒋 小 可
指 导 教 师: 毕 军 教 授

二〇一六年六月

Research on the Key Issues of Usability of Named Data Networking

Dissertation Submitted to
Tsinghua University
in partial fulfillment of the requirement
for the professional degree of
Doctor of Engineering

by
Jiang Xiaoke
(Computer Science and Technology)

Dissertation Supervisor : Professor Bi Jun

June, 2016

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：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（2）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容；（3）根据《中华人民共和国学位条例暂行实施办法》，向国家图书馆报送可以公开的学位论文。

本人保证遵守上述规定。

（保密的论文在解密后应遵守此规定）

作者签名：_____

导师签名：_____

日 期：_____

日 期：_____

摘要

根据思科视觉网络指数报告^[1]，全球互联网流量在过去 5 年中增长了 5 倍，并将在接下来的 5 年中增长近 3 倍；2014 至 2019 年期间，IP 流量仍将以年均复合增长率 23% 的速度高速增长；并且内容分发类的应用所产生的流量占大部分；估计到 2019 年，仅视频流量一项就将占据整个 IP 互联网流量的 80%，由此可见当前网络面临最主要的任务和挑战是数据分发。而随着虚拟现实、增强现实这种将二维视频升级为三维视频的新应用不断涌现和发展，网络终端用户将对数据分发的效率和规模产生更高的需求^[2,3]。然而，源自上个世纪六七十年代的 IP 协议 (Internet Protocol)^[4]，通过对设备接入点 (attachment point) 命名，来解决大型机时代亟须的从有限数量的主机上分享资源的需求。而这种设计不能很好地满足当前“后 PC”(post-PC) 时代互联网上无处不在的数据分发的需求。在 IP 通信中，必须有确定的发送方和接收方，两者中间的通信路径也往往由路由协议唯一确定，数据只能沿着唯一的 end-to-end 管道传输，这种简洁但“死板”的通信模式不能利用冗余数据源和链路，严重制约了数据分发的效率和规模。

实际上，终端用户真正关心的是他们想要的数据 (what)，而不是数据如何取得 (how)，不是数据存储在哪里 (where)，也不是数据从哪条链路传输 (which)，根据这一最基本的观察，研究者提出命名数据网 (Named Data Networking, NDN) 这种以信息为中心的网络体系结构。NDN 直接对数据进行命名，数据有效性独立于数据存储和通信会话；在具体设计中，将很多现有的基于 IP 的 patch (IP 补丁) 和 overlay (应用层覆盖) 方案中的设计元素，如基于数据的安全、带状态转发、多路径、网络层缓存等，以对命名数据进行操作的角度创造性的组合起来。这使得 NDN 可以充分利用多路径、多数据源、多通信接口、冗余链路和存储来提高数据分发效率，扩大数据分发规模。

NDN 的目标符合互联网数据分发的潮流，它的设计反映了互联网技术发展的趋势，然而在当前阶段，要把 NDN 应用于实际的数据分发，还面临着一些严峻挑战，本文把其中一些关键挑战分为三类：1) 网络运行问题：路由扩展性，移动性支持等 NDN 网络运行中遇到的问题；2) 模型优化问题：信息为中心的设计导致 NDN 通信迥异于现有的 IP 模式，缺少模型来衡量通信效率，优化通信参数；3) 兼容部署问题：NDN 不能兼容 IP 及无处不在的 IP 应用，同时 NDN 网络也缺少资源吸引用户使用 NDN。

针对这些问题，本文研究分别设计和实现了 nDNS (DNS for NDN)，ACM (Adap-

tive Chunk Mode) 和 nCDN (NDN-based CDN)，分别予以解决。

- nDNS 是为 NDN 网络设计的名字解析系统。目前解决 NDN 网络运行中的路由扩展性方案^[5]，支持移动性^[6,7]等问题的主流方案，都需要的一个提供名字解析的网络基础设施服务，nDNS 就是这种服务的一个具体方案；本文选择 DNS 作为主要参考对象，在继承 DNS 很多基本概念和设计原则的基础上，也将一些重要机制适配于 NDN 这种信息为中心的网络层协议，充分发挥了 NDN 体系结构带来的优点；nDNS 目前已经部署在 NDN 官方测试床 NDN Testbed^[8]，实际服务于各种 NDN 应用。
- ACM 是一个基于 NDN 网络通信效率模型来动态改变包尺寸，优化通信效率的机制。NDN 这种信息为中心设计有两个显著不同于 IP 的特点：1) 每个数据包的开销达到几百个字节，通信亟须优化；2) 数据获取中数据请求者没有确定的通信对端，现有的 end-to-end 服务模型无法衡量 NDN 通信效率。本文考虑了 NDN 在数据包开销、通信模式这些方面与 IP 的不同，建立了 NDN 网络通信效率模型，并用连续时间的马尔科夫链来分析该模型。模型准确描述了 NDN 通信效率，实验与理论误差仅 0.9%；根据通信效率模型，本文提出 ACM 根据网络状态调整数据包尺寸，并把 ACM 用于视频播放器中，实验结果显示通信效率提高 12%。
- nCDN 是用 NDN 来增强现有 CDN 的框架，提供了 NDN 渐进部署方案。由于 NDN 网络上缺乏数据资源且不兼容 IP，本文以 CDN 作为突破口，提出 nCDN，用 NDN 来改造当前 CDN，一方面可以把 CDN 上的大量数据资源引入 NDN 网络中，一方面提供了 NDN 应用于当前 IP 网络，渐进部署的渠道。本文搭建了 8 个节点跨城市、跨 ISP 的小规模 CDN 平台，能够实际加速视频与网页，最好的情况下数据访问延时降低了 99.5%。

综合而言，NDN 的实际应用还面临着很多问题，本文解决了功能、性能和兼容性方面的一些关键挑战，是对 NDN 网络体系结构的重要补充；在具体建模过程中，本文首先指出 NDN 以信息为中心设计的独特性：每个数据包的开销较大，数据获取没有特定的路径；在系统设计中，本文充分利用了 NDN 独有的优势，如多路径、网络缓存，网络路径选择等，来简化系统设计，提高系统系能；并且，从设计、实现到部署，本文为所有方案都提供了可运行的系统和平台，积累了充分的 NDN 应用开发经验，提供了 NDN 应用于当前网络的切实“可用”方案。

关键词：命名数据网，信息中心网，未来互联网，数据分发，名字解析；命名数据网，信息中心网，未来互联网，数据分发，名字解析

Abstract

According to cisco Visual Networking Index^[1], the global IP traffic has increased five times in last five years, and will continue growing to three times in next five years. What is more, it is expected that video traffic will account for 80 percent of all traffic in 2019. It is obvious that data distribution is the main goal and challenge of the Internet. With the development of emerging applications like virtual reality and augmented reality, end users desire faster and more efficient data retrieval service than in the present. Whereas IP, originated in the 1960s and '70s, aims to share resources from a small set of mainframes by naming the attachment points of devices. As a consequence, it is more or less outdated when facing the challenge of large-scale data distribution in post-PC era. In the context of IP, there must be two pre-defined ends, i.e., a receiver and a sender, between whom the path is usually solely decided by the routing system. Data is transmitted in this end-to-end pipe without utilizing redundant links and data, which greatly confines the scale and efficiency of data distribution. In fact, it is the data pe se that end users care, but not how to fetch the data, or where the data is stored, or from which link the data is retrieved. Base on the observation, Named Data Networking (NDN) is proposed to facilitate data distribution by naming the data directly. In the context of NDN, the validity of data is independent of the data container or communication session. Therefore, end users are able to make full advantages of multiple interfaces, paths and data sources to scale up and speed up data distribution.

However, there are still some key issues to apply NDN into practice over the current Internet infrastructure. We categories those issues into three types: 1) issues of running network, such as routing scalability, lack of mobility support; 2) issues of modelling and optimization, which is consequence of this new information-centric design; 3) issues of compatibility with IP and incremental deployment of NDN. In this thesis we have proposed, designed and prototyped nDNS (DNS for NDN), ACM (Adaptive Chunk Mode) and nCDN (NDN-based CDN) to address those issues, respectively.

- nDNS is a name resolution service customized for NDN. Given the well-accepted solutions for routing scalability and mobility^[6,7] require name resolution service, we built such a service, that is nDNS. nDNS inherits some basic concepts and design principles from DNS. Besides, it adapts the detailed mechanisms to the information-

centric architecture, in order to leverage the built-in advantages of NDN, such as in-network caching, hop-by-hop server/link selection. NDNS has been deployed on NDN Testbed^[8] and serves NDN-based applications.

- ACM changes the size of transmitted chunk adaptively based on NDN data retrieval model. The information-centric driven design of NDN leads to two unique characteristics: 1) the overhead of each data chunk is up to hundreds of bytes, which desires optimization for data distribution; 2) data is retrieved without knowing pre-defined correspondents, which makes existing end-to-end model no longer fits NDN. The average relative error between experimental results and that derived from our mathematical model is 0.97%, and the efficiency of data retrieval increases 12% when ACM is applied to an online video player.
- nCDN leverages NDN to enhance CDN framework. Given data resources in current NDN network is very limited, we advocate nCDN, which leverages NDN to enhance CDN, as the breakthrough of introducing massive of data from IP network as well as providing a way of incremental deployment for NDN. We have deployed nCDN on a testbed made up of servers located in different IPSs and cities. Our nCDN is able to speed up the data distribution of webpage and video, and retrieval delay is reduced to 0.5% in the best case.

In sum, there are still many challenges to apply NDN in practice, and we address some key issues in this thesis, making our work important supplements of NDN architecture. During the modelling process, we first, to the best of our knowledge, point out the non-negligible overhead of each NDN data chunk as the consequence of information-centric design. During the design of nDNS, ACM and nCDN, we fully utilize the advantages of NDN architecture, including support for multipath, multiple data sources, in-network caching, hop-by-hop server/link selection, etc, to simplify the application design and improve the efficiency. Furthermore, we make our solutions really “usable” with sophisticated design and solid implementation and deployment.

Key words: Named Data Networking; Information Centric Networking; Future Internet;
Data Distribution; Name Resolution

目 录

第1章 引言	1
1.1 研究背景	1
1.2 NDN 部署和应用中面临的挑战	3
1.2.1 网络运行问题	3
1.2.2 模型优化问题	4
1.2.3 与 IP 兼容性问题	5
1.3 论文主要研究内容	5
1.3.1 nDNS 解决网络运行问题	5
1.3.2 ACM 建立模型并优化传输效率	6
1.3.3 nCDN 解决兼容部署问题	7
1.4 论文主要贡献	8
1.5 本章小结	9
第2章 相关工作综述	11
2.1 信息中心网 ICN	11
2.1.1 ICN 的基本设计理念	14
2.1.2 ICN 数据命名	16
2.1.3 ICN 数据获取	17
2.1.4 ICN 缓存	18
2.1.5 ICN 数据验证	20
2.2 命名数据网 NDN	21
2.2.1 报文格式和节点模型	22
2.2.2 层次化命名	24
2.2.3 基于名字的路由	25
2.2.4 带状态的路由转发	26
2.2.5 基于数据的安全模型	27
2.3 本章小结	28
第3章 nDNS : NDN 名字解析系统	29
3.1 相关背景：现有名字解析系统	30
3.1.1 DNS 设计	30
3.1.2 NDN 中现有名字解析系统设计	34

3.2 nDNS 设计	35
3.2.1 nDNS 命名空间和数据命名	36
3.2.2 数据格式	38
3.2.3 迭代解析	39
3.2.4 数据验证	41
3.3 数据管理与相关问题讨论	42
3.3.1 缓存解析器的角色	42
3.3.2 名字域更新和同步机制	43
3.3.3 nDNS 的特殊地位	44
3.4 系统原型、部署与实验	44
3.4.1 系统原型与部署	44
3.4.2 网络层缓存的优势	45
3.4.3 主机/链路选择的优势	46
3.5 nDNS 总结	48
第 4 章 ACM : NDN 通信效率建模	51
4.1 相关背景: NDN 链路技术 & 应用数据单元、分段与分片	53
4.1.1 NDN 的链路技术	53
4.1.2 NDN 中应用数据单元、分段和分片	54
4.2 数学建模	55
4.2.1 单跳获取数据	56
4.2.2 多跳获得数据	58
4.2.3 多路径传输, 支持多数据源, 网络缓存和请求聚合	59
4.2.4 松弛假设: 异构链路	60
4.2.5 松弛假设: 丢帧率 (Ω) 不是常数	62
4.3 ACM 机制	64
4.3.1 ACM 命名和交互机制	64
4.3.2 包尺寸决定机制	65
4.4 相关讨论	66
4.4.1 动态变化的尺寸影响	66
4.4.2 访问延时	66
4.4.3 Interest 的带宽消耗	67
4.5 系统原型与模型验证	67
4.5.1 ACM 原型与应用	67
4.5.2 ACM 机制有效性	67

4.5.3 模型验证.....	70
4.6 建模总结	71
第 5 章 nCDN: NDN 渐进部署方案	74
5.1 相关背景: CDN & HTTP/HTTPS.....	74
5.1.1 CDN	74
5.1.2 HTTP/HTTPS & CDN 如何支持 HTTP/HTTPS	77
5.2 nCDN 的设计.....	80
5.2.1 nCDN 系统概述: 把 NDN 支撑 CDN	80
5.2.2 HTTP-NDN Gateway: NDN 与 HTTP 之间的翻译	82
5.2.3 nCDN 内部通信	83
5.2.4 数据格式.....	84
5.3 nCDN 相关问题讨论.....	85
5.3.1 实时数据流.....	85
5.3.2 数据部署和安全性	85
5.3.3 部署激励.....	86
5.4 系统原型、部署与实验	86
5.4.1 原型与部署	86
5.4.2 路径选择	88
5.4.3 不准确路由	88
5.4.4 与 CDN 性能比较.....	90
5.5 nCDN 总结	92
第 6 章 论文总结	94
参考文献	96
致 谢	103
声 明	104
个人简历、在学期间发表的学术论文与研究成果	105

主要符号对照表

IP	互联网协议 (Internet Protocol)
NDN	命名数据网 (Named Data Networking)
ICN	信息中心网 (Information Centric Networking)
ISP	互联网服务提供商 (Internet Service Provider)
CDN	内容分发网络 (Content Delivery Network)
CSP	内容服务提供商 (Content Service Provider)
DFZ	缺省默认路由区 (Default Free Zone)
MTU	链路最大传输单元 (Maximal Transmit Unit)
GSLB	全局服务器负载局衡 (Global Server Load Balancing)
Interest	NDN 网络中请求消息 (NDN Interest Packet)
Data	NDN 网络中数据报文 (NDN Data Packet)
consumer	NDN 网络中数据使用者
producer	NDN 网络中数据生产者
DNS	域名解析系统 (Domain Name System)
HTTP	超文本传输协议 (Hyper-Text Transfer Protocol)
HTTPS	安全超文本协议 (HTTP over TLS/SSL)
FIB	路由转发信息表 (Forwarding Information Base)
PIT	请求等待表 (Pending Interest Table)
CS	内容仓库 (Content Store)

第1章 引言

1.1 研究背景

自互联网被发明之后，它就不断地在演化，从大型机时代服务于少数人从有限数量节点上分享资源的网络，变成了人类社会普遍需要的基础设施之一。但相比大型机时代的互联网，“后 PC”时代的互联网面临的主要任务和挑战已经从建立 *end-to-end* 的连接（从远程输入输出终端到有限的大型机站点），演化为无所不在的数据分发^[9]。而互联网的“窄腰”，即 IP，却依然保持不变。IP 的设计是通过对网络设备的接入点命名，建立 *end-to-end* 管道，来解决远程终端（输入输出设备）从有限数量的大型站点分享资源的问题；而在当前“后 PC”时代，个人电脑、移动设备极为普及，资源（数据资源、计算资源、存储资源）无处不在，这种情况下，IP 的 *end-to-end* 管道模型颇显“过时”，不能很好地满足当前网络通信的需求。研究者尝试提出两类不同的方案来改造 IP，即 *patch*（IP 补丁）和 *overlay*（应用层覆盖）。*patch* 方案往往需要改变 IP 的语义，如 IP 组播是通过 IP 地址来命名通信组，这一类的方案针对性很强，往往只能改进一个或几个特点，并且缺乏部署激励，在实际应用中较少；*overlay* 方案，如 CDN 和 p2p，是在应用层实现一个中间件系统辅助通信，但由于应用层缺乏网络层信息容易导致很多不必要的域间流量，而且由于天然的中间人 (Man-in-the-Middle) 角色，在数据验证方面还面临重要的挑战。虽然这些技术都不能摆脱 IP 的限制，但这些技术本身体现了当前网络真实的需求，反映了网络技术发展最强烈的趋势。

为了从根本上解决当前（和未来一段时期内）互联网数据分发的挑战，研究者从直接对数据命名的角度，提出信息中心网 (Information Centric Networking, ICN) 的概念，命名数据网 (Named Data Networking, NDN) 是这种以信息为中心的未来网络体系结构中代表性的设计。NDN 的设计并非是“无中生有”，而是受到很多设计的启发^[10]，很多的设计元素都是从 IP 的 *patch* 和 *overlay* 方案中借鉴而来。NDN 继承了 DNS 层次化命名在网络层直接标识数据，并在网络层直接操作这种命名数据，如基于名字的路由、基于名字匹配的数据通信、网络层缓存、基于数据的安全等。在 NDN 设计中，数据本身是网络中第一主体，数据的有效性独立于数据存储和通信会话，这使得 NDN 通信可以突破 IP 中 *end-to-end* 管道的限制，可以充分利用多路径、多接口、多数据源、冗余数据和链路来扩大数据分发规模，加快数据分发速度。

NDN 研究在过去几年里发展迅速^[11]，关于 NDN 相关设计及功能支持，如路

由、快速名字查找、安全、缓存、移动性支持等方面都取得了一系列成果。为了把这种新兴的设计真正利用起来，NDN 项目提供了 NDN 协议栈，即 NFD^[12] 和 NDNx^[13]，NDN 路由协议实现 NLSR^[14]，以及多种 NDN 应用开发语言库，如 ndn-cxx^[15] (C++ 库)，Consumer-Producer API^[16] (C++ 库，基于 ndn-cxx 的进一步封装)，PyNDN^[17] (Python 库)，NDN.JS^[18] (Javascript 库)，jNDN^[19] (Java 库) 等。在具体应用方面，目前 NDN 建立了官方测试床 NDN Testbed^[8]，该测试床包含 31 节点^①，覆盖了美国、欧洲和亚洲。该测试床把 NDN 通过 overlay 的方式运行在 TCP/UDP 之上，NDN 应用可以通过测试床进行通信。更进一步的，NDN 还实际解决了一些问题，如 NDN 用于 UCLA 建筑管理系统 BMS (Building Management System)^[20]，用于全球气候监测项目 CMIP5 (Coupled Model Intercomparison Project Phase 5) 中的数据采样和收集系统^[21]，用于欧洲核子研究中心 CERN (European Organization for Nuclear Research) 大型强子对撞机 LHC (Large Hadron Collider) 实验数据的采集和分析系统^[22]，其它的还有灯控系统^[23]、语音会议^[24]、多人在线聊天^[25]、车载通信^[26] 等等。

然而，现有这些工作还不足以使 NDN 大规模应用，这一目标还面临一系列的挑战（图1.1）：

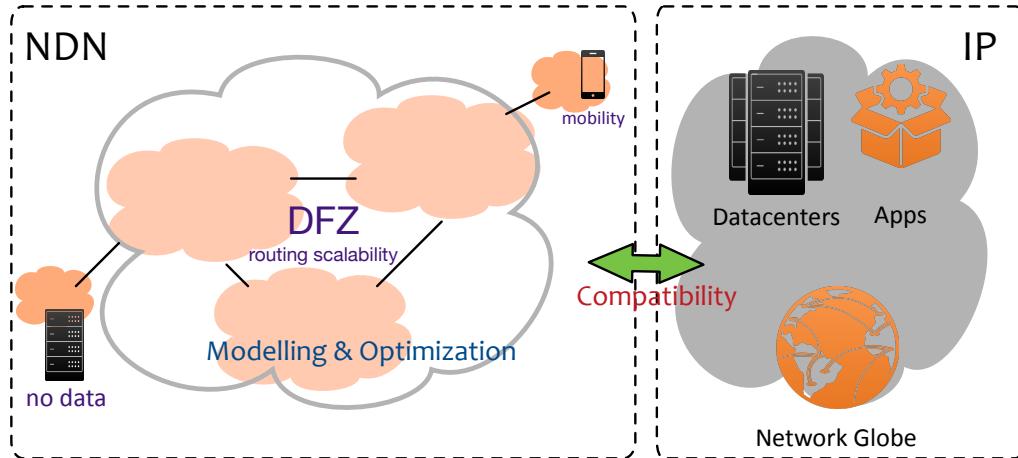


图 1.1 NDN 的广泛部署和使用还面临一系列挑战：既有来自 NDN 本身网络运行的问题，e.g., DFZ 路由器中 FIB 条目过多，iPhone 移动性支持；还有网络通信效率理论与优化问题；目前 IP 作为当前世界性的网络基础设施，NDN 也面临着与 IP 网络兼容互通问题，以增加 NDN 网络资源，促进渐进部署。

- 另外随着 NDN 研究不断地深入，尤其不断从 NDN Testbed 获得运行经验，关于 NDN 网络运行中一些问题凸显，例如 1) 只有一部分应用的名字能存储在 FIB 表中，(图1.1 中 DFZ (Default Free Zone) 中路由器 FIB (Forwarding

^① 该数据是基于 2016.4.11 目的统计结果。

Information Base) 表条目太多), 很多名字需要映射到在 FIB 中的名字来路由;

2) 随着越来越多的移动设备接入到 NDN Testbed , 当数据产生者移动时, 相对的请求报文依然需要能够路由到对应的移动节点(图1.1中 iPhone 移动); 3) 由于数据产生和使用在时间和空间上解耦合, 数据被验证时它的生产者已经不在线的情况下, 仍然需要完成数据验证。

- 由于 NDN 迥异于 TCP/IP , 已有的网络通信模型不能用于衡量 NDN 通信效率, 也无法应用于优化 NDN 通信。这使得目前 NDN 应用使用一些经验值作为数据通信中的参数, 例如大部分应用采用 4096 字节作为数据报文中载荷大小。
- 目前大部分 NDN 应用都有比较明显的局限性: 工作的场景都是纯 NDN 的网络环境, 应用本身都需要支持 NDN 协议栈, 并且通过纯 NDN 的网络完成数据通信; 而实际中, 网络的基础设施是基于 IP , 当前路由器也不支持 NDN 转发。
- 总体上说, 目前 NDN 网络上缺乏数据资源, 不能吸引终端用户部署、使用 NDN ; 因为缺少终端用户的 support, 反过来各种 ISP (Internet Service Provider)、CSP (Content Service Provider) 也缺少足够的激励, 以至于冒着失败的风险, 投入资源来改造升级现有业务, 使之支持 NDN 。

1.2 NDN 部署和应用中面临的挑战

本文在深入分析上述 NDN 应用和部署中已经发现的问题之后, 将它们分成三类, 网络运行问题(功能)、模型优化问题(性能)和兼容部署问题, 下文一一展开叙述。

1.2.1 网络运行问题

NDN 的部署首选需要解决的是 NDN 网络本身运行的问题, 只有解决了这些问题, 部署才有可能。此处举几个目前已经鉴别出的在 NDN 运行网中的问题:

- 命名空间分配和信任管理: 为避免名字劫持, 顶级命名空间应该由相关机构(如 ICANN^[27]) 统一分配, 名字所有者持有相关机构认证的电子证书; 顶级名字空间所有者可以将子命名空间进一步分配。
- 证书仓库: NDN 中的数据验证与数据产生在时间和空间上解耦合, 在 consumer (数据使用者) 验证数据时可能 producer (数据产生者) 已经不在线, 在这种情况下 consumer 仍然需要所有相关数字证书才能完成数据验证。
- 路由扩展性: NDN 中数据单元可能多达 $10^{15} - 10^{21}$ 个^[28], NDN DFZ 路由器

中 FIB 标的条目，考虑到 NDN 名字可以聚合，一个根据当前顶级域名数量基础上的估计值是 6×10^8 ^[29]，这个数量是当前 IP DFZ BGP 路由表的 200 倍。考虑到当前的硬件技术及价格，NDN 路由器 FIB 只能载入一部分名字；在这种情况下为保证整个网络可达性，NDN 团队提出 SNAMP^[5] 方案，通过 LINK 对象（图2.7，把一个名字链接到其它一组名字上的数据结构）把不能路由的名字绑定到可以路由的名字，来解决路由扩展性的问题，该方案中需要一个名字解析系统来支持这样的映射。

- producer 的移动性支持：NDN 中 consumer 移动之后，重新发送请求即可；但对于 producer，它的移动则需要额外的名字解析系统来处理^[30]；一个典型支持 producer 移动的方式是及时更新名字解析系统里 producer 的 LINK 对象。

由此可见，NDN 的核心网络，数据请求者，数据使用者等都面临一些实际运行上的挑战，只有解决了这些挑战，才能让 NDN 网络运转起来。

1.2.2 模型优化问题

NDN 设计与 TCP/IP 不同，这些差异来自于包格式、服务模型等，这也导致了数据传输数学模型的不同。一方面，TCP/IP 中数据通信有严格的接收方和发送方，两者通过路由系统决定的确定“管道”中进行通信；而在 NDN 中，多路径、多接口、多数据源、网络层缓存等因素使得 consumer 不能确定它发出的请求从哪里取得对应的数据，也就是说，NDN 数据获取中数据请求者没有确定的通信对端，这种信息为中心的通信方式完全突破了 end-to-end 服务模型，也使得现有的基于 IP 的通信效率数学模型不再适用，以至于 NDN 具体应用需要根据经验选取参数来优化通信效率，例如 4096 字节为数据包默认载荷尺寸。

另一方面，NDN 作为以内容为中心的设计，在网络层直接操作命名数据，包括内容的路由、缓存、安全等，带来很多的便利。但这种内容为中心的设计并非是没有代价的，它要求数据包具有幂等性：1) 数据包的名字能够独立于内容存储和通信会话；2) 数据包本身需要包含网络层操作的所需的众多信息：例如内容本身描述（Content 域），缓存相关属性（FreshnessPeriod 域），安全相关的信息（Signature 域）等。从应用和用户的角度上看，一个数据包中真正有用的应用层的载荷（payload），也就是 Content 域；除载荷以外的其它域都是为了传输载荷所需要的开销（overhead）^①。信息为中心的设计导致 NDN 中的数据包中含有往往高达几百个字节的开销，当前 NDN 原型系统实现中一个典型的开销值是 650 字节^[12,13]，

^① TCP/IP 中常用包头（packet header）代指这些部分，但 NDN 数据包中有一些开销是位于包尾部，如签名，故本文用开销来指代这些部分。

相比于 TCP/IP 数据包中的几十字节的包头开销^①，NDN 每个数据包开销非常巨大而在传输中不可忽略，这种情况下，NDN 通信需要优化，尤其是数据包尺寸，会极大地影响通信效率。因此，建立 NDN 传输理论模型来衡量 NDN 网络传输效率，准确优化传输参数，提高传输效率是当前 NDN 应用的关键问题之一。

1.2.3 与 IP 兼容性问题

不同于基于 IP 的 patch 和 overlay 设计，NDN 是一种颠覆式 clean-slate 的设计，网络层直接对内容命名，而不是对接入点命名，因此 NDN 节点不能直访问 IP 的主机，而 IP 的应用也不能直接接入 NDN 网络。

作为一个新兴的技术，NDN 还在发展阶段，虽然它更有利于数据分发，但 NDN 网络中几乎没有数据资源，因此很难吸引终端用户安全 NDN 协议栈，使用基于 NDN 的应用程序。与此同时，当前互联网是基于 TCP/IP 的基础设施，几乎所有网络应用都是通过 TCP/IP 接入互联网，并以 TCP/IP 的方式访问其他节点，互联网的价值正是通过保障这些应用的正常运行来体现，而这大量的应用并不能直接利用 NDN。

在这种条件下，如何让用户使用 NDN，让 NDN 的应用和流量从无到有是一个巨大的挑战；与此同时，如果 NDN 最终能取代 IP，从 NDN 到 IP 的迁移不应该是一蹴而就的，而是逐步推进的，在过渡阶段如何让两种不同类型的网络互联互通也是重要的需求。

1.3 论文主要研究内容

如图1.2 所示，本文针对把 NDN 在具体部署和应用中面临的三类挑战，分别提出 NDN (DNS for NDN)，ACM (Aaptive Chunk Mode) 和 nCDN (NDN-based CDN) 分别予以解决。

1.3.1 nDNS 解决网络运行问题

针对 NDN 网络运行问题的解决方案，如路由扩展性解决方案 SNAMP^[5]，移动性支持方案 Kite^[7] 和数据验证（第 2.2.5 章），都依赖于一个名字解析系统。通过这个名字解析系统，终端用户能够通过名字获得证书、LINK 对象或其它数据，而 nDNS (DNS for NDN) 则是本文为 NDN 的网络运行和管理而设计并实现的名字解析系统。该系统的设计受到了 DNS^[31,32]+ DNSSEC^[33–35] 启发，继承了很多 DNS 中重要的概念和设计理念，如层次化的命名空间、域名、名字域、缓存数据和冗余

^① 不含可选包头，TCP 和 IP 包头大小分别是 20 字节、8 字节。

数据、递归查找、迭代查找、数据签名与验证等；但由于在网络层 NDN 与 IP 在数据获取与验证等方面有着本质的区别，nDNS 设计需要适配 NDN 下的通信模式，在此基础上还充分利用 NDN 数据通信和安全方面的优势，所以 nDNS 与 DNS 在迭代查询机制、缓存、主机/链路选择、数据验证等方面也有明显的区别，更多地依赖 NDN 固有支持来简化设计，提高效率。

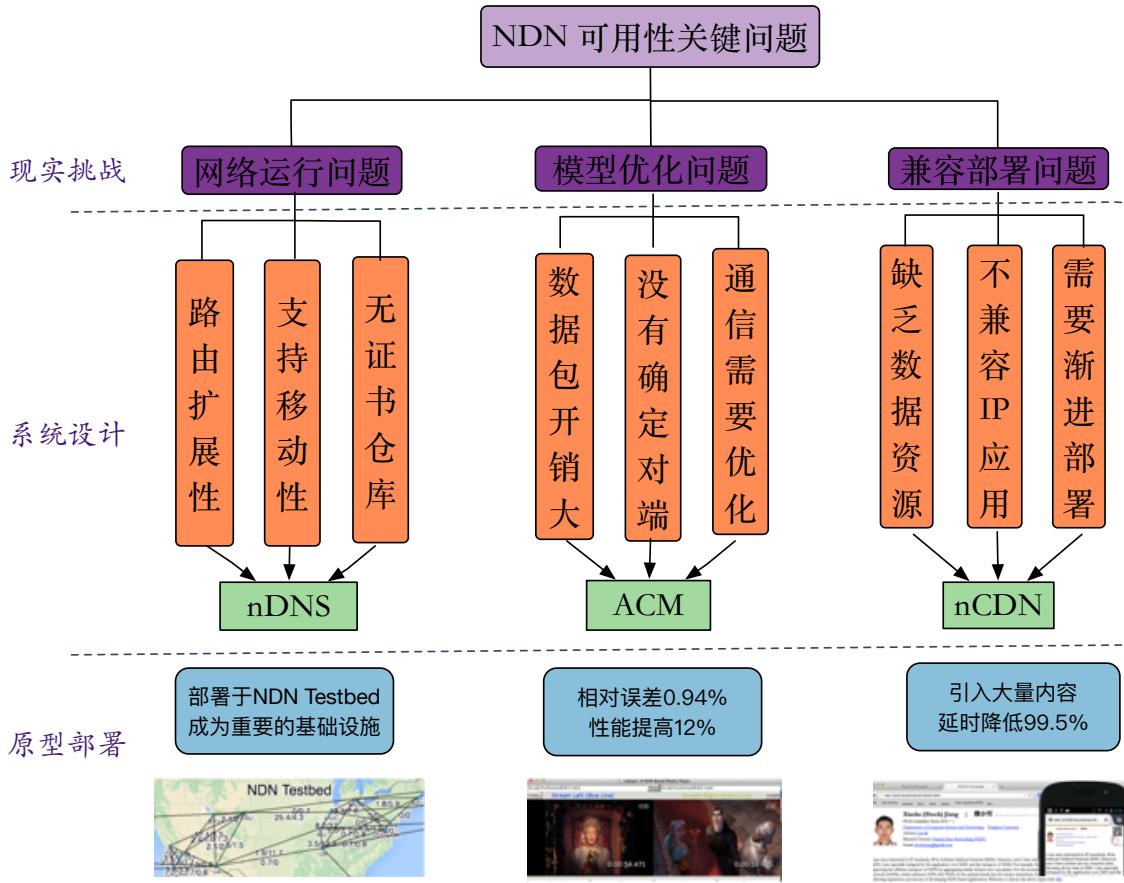


图 1.2 本文内容总结：本文分别用 nDNS、ACM、nCDN 来解决 NDN 走向普通应用中面临的一些功能、性能和部署问题，这是对 NDN 网络体系结构的重要补充。本文的解决方案有设计、实现和部署，切实“可用”。

1.3.2 ACM 建立模型并优化传输效率

针对网络传输效率模型，本文首先提出用所有吞吐量 (throughput) 中的有效吞吐量 (goodput) 的比例来衡量网络传输效率。本文定义这个比值为 G2T (goodput to

throughput):

$$\begin{aligned} G2T &= \frac{Goodput}{Throughput} \\ &= \frac{payload}{(payload + overhead)/(1 - ChunkLossRate)} \end{aligned} \quad (1-1)$$

针对 NDN 通信没有确定的通信对端的情况，本文考虑了 NDN 网络中数据通信的种种情况，从单跳、多跳、多路径（包括多数据源）等不同情况，由简单到复杂建立了网络通信效率数学模型，并用连续时间的马尔科夫链来分析链路上丢帧率的影响。这个模型在给定开销、链路上丢帧率和最大传输单元 (Maximal Transmit Unit, MTU) 下推导出最优化包尺寸，能最大化传输效率。在此基础上本文提出 ACM (Adaptive Chunk Mode) 机制，能够利用该模型，动态改变传输数据包的尺寸，从而最大化传输效率。

1.3.3 nCDN 解决兼容部署问题

CDN (Content Delivery Network) 作为 IP 网络上大规模数据分发最重要的手段，它承载了海量的资源。传统的 CDN 设计中的重点与难点是把 TCP/IP 这种以连接两个节点为目的的设计适配于通过内容冗余来达到大规模数据分发的目的，这种适配通过全局网络内容的实时监控、DNS 动态映射、以及私钥/证书共享等“hacking”手段得以实现，但同时也留下了一些问题。

NDN 网络中缺少数据，缺乏对传统应用的支持；但 NDN 的优势能够扩大数据分发的规模，提高数据分发的效率，并且它的设计与 CDN 有很多相似之处。因此，本文以 CDN 为突破口，提出 nCDN (NDN-based CDN) 框架，把 NDN 作为一个“薄层”(shim layer) 置于 IP 之上、冗余数据之下，这样既把 NDN 的优势用于 IP 网络上的大规模数据分发，又能把 CDN 中大量资源引入到 NDN 网络。而 nCDN 则充分利用 NDN 名字转发，hop-by-hop 主机/链路选择，快速失败恢复，网络层缓存支持等信息为中心的设计，既简化了系统设计，又充分利用了资源（数据资源、带宽资源等），提高了数据分发效率。

总体来说，nCDN 把 NDN 应用于当前 TCP/IP 网络基础设施上，提供了一个效率更优、设计更简的大规模数据分发框架；潜在地，向 NDN 网络中引入大量的数据资源，并为 IP 向 NDN 过渡提供了一个渐进部署的方案，这对于 NDN 的研究与发展有重要的意义。

1.4 论文主要贡献

本文针对 NDN 运行中遇到的功能、性能和兼容性挑战展开深入研究，如图1.2 所示，本文贡献总结如下：

- 通过名字解析系统解决了网络运行中的挑战，该设计的原型系统已成为当前最大的 **NDN** 网络中重要的基础设施服务。针对网络运行中的挑战，本文为 NDN 网络设计了一个分布式的名称解析系统 nDNS，允许不同的应用根据已有的名字安全地获取解决不同挑战需要的信息（如核心网路由扩展性，数据产生者移动性支持，数据使用者数据验证等）；nDNS 在 NDN 网络中的角色就像 DNS 在 IP 网络的角色，虽然是在应用层实现，但工作在网络层和应用层之间；它向应用提供路由和数据验证等所需要的信息，从而引导其他程序启动数据获取；nDNS 最后部署于 NDN Testbed，实际服务于不同的应用，如多人在线聊天应用 ChronoChat^[25]、在线视频点播程序 NDNTube^[36]，成为目前最大的 NDN 网络的必不可少的网络基础设施服务。在 nDNS 设计过程中，本人还积累把传统的基于 IP 的系统迁移到 NDN 中的一些经验，这些经验有助于未来更多的应用迁移（第 3 章）。

- 首次指出 **NDN** 通信中巨大的开销，提出 **NDN** 通信效率模型，模型误差仅 **0.9%**，实际应用中网络传输效率提高 **12%**。针对模型优化问题，本文首先指出 NDN 通信中两个重要特点：1) 数据请求者没有特定的通信对端；2) 较大的通信开销；根据这两个特点，本文提出了 NDN 网络数据传输的效率模型，并用连续时间的马尔科夫链来分析链路丢帧率的影响，该模型准确地描述了 NDN 通信的效率，实验结果与理论结果的相对误差仅为 0.9%；进一步本文根据该模型设计了 ACM 机制，能够根据每个数据包开销，网络状态动态地调整包尺寸，从而提高应用传输效率；本文将 ACM 应用于视频播放应用 nPlayer^[37]，其传输效率提升 12%（第 4 章）。

- 首次将 **NDN** 与 **CDN** 结合起来^①，发挥各种优点，测试床显示数据访问延时最好情况下降低了 **99.5%**。针对兼容性问题，本文以 **CDN** 作为突破口，1) 将 NDN 的设计优势应用于 **CDN** 中，简化系统设计，提高运行效率；2) 将 **CDN** 中的海量数据引入到 **NDN** 里，增强部署激励，吸引终端用户。本文根据 nCDN 设计建立了 8 个节点，跨城市、跨 ISP 的小规模 **CDN** 平台，作为 NDN 渐进部署的突破口，该平台能够支持 IP 与 NDN 应用，实际加速视频与网页。实验结果显示，相比与传统 **CDN**，nCDN 将服务用户数量的瓶颈提高 60%（网络流量非过载的情况下），平均延时缩短 20%；是实际应用中，利用 nCDN 加速可以将访问延时减少

^① 目前有一些支持 NDN 网络与传统 **CDN** 互联互通的研究^[38]，但尽本人所知，本文是首个将 NDN 技术应用于 IP 网络，用来全面改造 **CDN** 的研究。**NDN** 网络与 **CDN** 互联互通的工作连接两种异构网络，本文是应用 NDN 技术，并且能够接入各种传统的 IP 应用。

99.5%。

需要特别说明的是，本文中的 nDNS，ACM 和 nCDN 三个工作在并不是完全独立，1) 它们服务与同样的目标，即使得 NDN 真正可用：nDNS 首先解决了自身功能问题，这使得大规模 NDN 网络得以运行；ACM 则解决了性能问题，通过建立网络通信的数学模型，来实现性能优化；而 nCDN 则是应用的问题，激励终端用户和 CSP 使用 NDN，允许 NDN 渐进部署。2) 在实际部署中它们可以相互协作，例如 ACM 机制可以用于 nDNS 和 nCDN 中的通信优化，nCDN 中也需要 nDNS 来提高所需要的路由和安全信息。总体上说，本文通过 nDNS、ACM 和 nCDN 提供了 NDN 应用于实际内容分发的方案，成为 NDN 体系结构的重要补充。

1.5 本章小结

ICN 和 NDN 的兴起是互联网自身演化的结果。经过近半个世纪的发展，尤其是 20 世纪 90 年代互联网大发展之后，互联网在规模、拓扑、用例、场景等方方面面发生了剧烈的变化，而 IP 及基于 IP 的 patch 和 overlay 方案都不能从根本解决当前和未来互联网面临的数据分发的挑战。在这种情况下，研究者提出 ICN 概念，通过在网络层直接操作数据来打破 IP 的限制，而 NDN 是 ICN 中代表性方案。NDN 沿用了 DNS 的层次化命名方法在网络层来标识数据（Data 报文），直接通过名字来路由请求报文（Interest 报文），采用“软状态”记录请求报文的转发路径，数据报文则通过该路径的反向路径返回给数据请求者，它还独立于数据存储和通信会话，对数据本身进行签名来保证数据分发的安全性。它的设计借鉴了已有的 IP 的 patch 和 overlay 方案中的很多设计元素。

虽然从体系结构的角度看，NDN 有比较明显的优势，但 NDN 的实际应用还面临着很多问题。本文解决了功能、性能和部署方面的一些关键挑战，是对 NDN 网络体系结构的重要补充。在具体建模过程中，本文首先指出 NDN 以信息为中心设计的独特性：每个数据包的开销较大，数据获取没有特定的通信对端；在系统设计中，本文充分利用了 NDN 独有的优势，如多路径、网络缓存，网络路径选择等，来简化系统设计，提高系统性能。并且从设计、实现到部署，本文为所有方案都提供了可运行的系统和平台，积累了充分的 NDN 应用开发经验，提供了 NDN 应用于当前网络的切实可用的方案。

后文按如下结构组织：第 2 章先介绍相关背景，由于 ICN 和 NDN 这个较新的研究领域，本文先从网络演化的角度阐述了 ICN 的兴起，从网络体系结构的角度介绍了 NDN 的设计，其中包括 IP 网络演化中主要挑战的变化，基于 IP 的 patch

和 overlay 方案，ICN 产生的背景和基本概念，NDN 基本设计及其设计中的一些挑战；第 3 章介绍 nDNS，在继承 DNS 中基本概念和基本原则，着重介绍它适配 NDN 方面的设计，如严格的自顶而下名字解析过程，利用网络层缓存来应答大量的流量，利用 hop-by-hop 的主机/链路选择来选择不同的名字服务器；第 4 章介绍 ACM，本文选择有效吞吐量比率来衡量网络传输效率，根据 NDN 传输的特点进行建模和优化；第 5 章介绍 nCDN，该章首先介绍了传统 CDN 的设计和“痛点”，再介绍了 nCDN 的设计、优势与需要克服的挑战；最后，第 6 章总结全文。

第2章 相关工作综述

2.1 信息中心网 ICN

如图2.1所示，互联网的设计可以追溯到20世纪六七十年代（互联网的前身 ARPANET 于 1967 开始）^[39]，当时一些稀缺昂贵的资源，例如读卡器、高速磁盘驱动器、计算单元等，只分布在有限数量的站点，并被整个社区分享。这种情况下，从有限数量的站点分享资源是建立网络最主要的目标。研究者发明了 IP 协议（1974 年首先提出 Transmission Control Program^[40]，1981 年正式拆分成 TCP/IP)^[4,40]，通过命名网络设备接入点 (attachment point) 来实现这个目标。

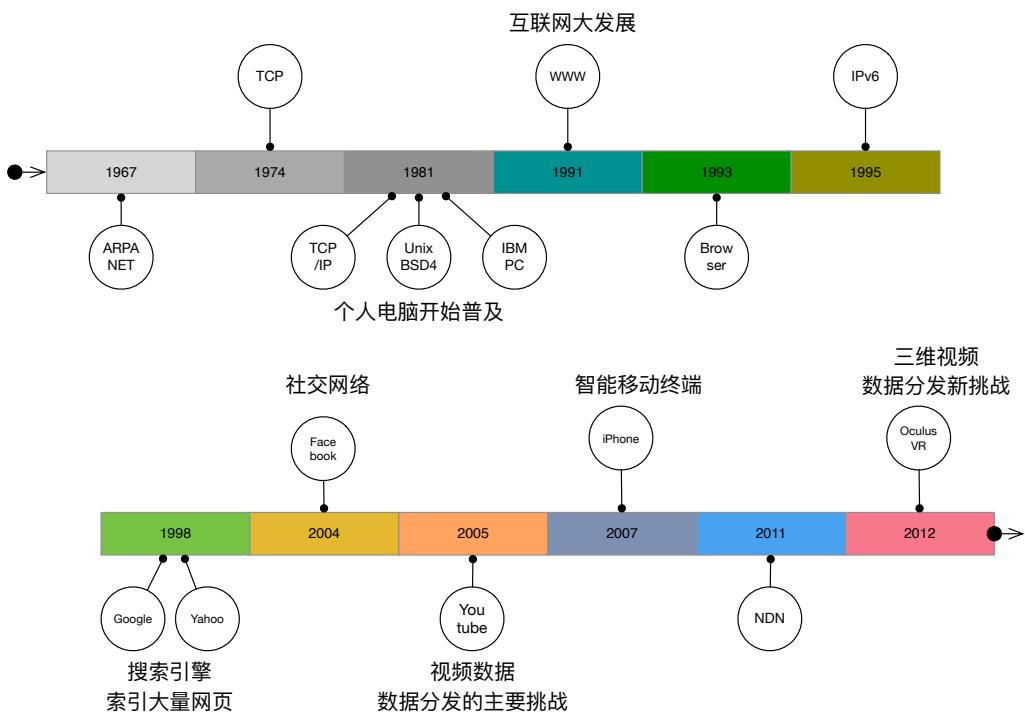


图 2.1 互联网发展史：IP 一开始是为从有限数量的大型机站点上分享资源设计的，而目前网络主要的挑战是数据分发，虚拟现实、增强现实这种新的应用对数据分发的效率和规模有更高要求。

时至今日，互联网已然发生根本性的变化：一方面，互联网高度普及，有大量的 CSP 和无数的终端用户，导致互联网上数据分发规模惊人。根据思科视觉网络指数报告^[1]：1) 全球互联网流量在过去 5 年中增长了 5 倍，并将在接下来的 5 年中增长近 3 倍；2) 2014 至 2019 年期间，IP 流量仍将以年均复合增长率 23% 的速度高速增长；3) 并且内容分发类的应用所产生的流量占大部分；4) 估计到 2019

年，仅视频流量一项就将占据整个 IP 互联网流量的 80%。由此可见当前网络面临最主要的任务和挑战是数据分发。另一方面，由于物联网、车载网络、自组织网络 (Ad-Hoc Network) 的兴起，资源（电子数据、计算能力、存储等）不仅仅是分布在大型网络站点，同时也广泛分布在各种终端设备，包括通过不同方式连接起来的、异构网络中的各种设备，例如移动手机、可穿戴设备、传感器、车辆、卫星等。由以上可见，互联网的当前主要任务和挑战已经从连接有限数量的主机，演化为分发无处不在的数据。而随着新的应用，如虚拟现实、增强现实这种将二维视频升级为三维视频的新应用不断涌现和发展，网络终端用户将对数据分发的效率和规模产生更高的要求。

面对数据分发的需求，IP 作为承载整个网络空间通信的基本设计，实际上已经超出了设计者们最初的目标，毕竟即便是设计者们也不能预测到，现代网络空间 (cyberspace) 在人类社会中承担着如此重要的角色；当 IP 这种对接入点命名，以主机作为网络第一主体的设计被用于当前无处不在的数据传输，就导致了一系列的挑战：

- 通信必须有确定的接收端和发送端，而且数据往往只能沿着路由系统决定的一条路径进行转发，而不能充分利用冗余的数据、路径和物理接口；这种 end-to-end 的方式严重限制大规模数据分发的规模和效率 (图2.2)。
- 对接入点进行命名要求把 IP 与其下面一层（通常是数据链路层）的标识符进行绑定，这使得移动支持变得困难；同样使得它不能充分利用广泛分布于网络边缘的广播信道。
- IP 不能消除异构网络间的差异，对于异构网络之间的通信，例如 IP 网络与传感器网络 (Sensor Network)、车载网络 (Vehicle Network)，延迟容忍网络 (Delay Tolerant Network, DTN) 之间的通信，通常很难直接建立起稳定的连接来传输数据。
- IP 设计本身没有任何的安全机制；后来增加了一些安全机制，如 TLS, IPSec，都是针对点到点之间的管道安全，只能在保障“临时”的信任——它们建立的数据安全只在连接时间内（时间限制）、对通信两端双方（空间限制）有效。

从根本上讲，上述问题是由于应用层数据分发这一目标与网络层提供的 end-to-end 连接两个节点的服务之间不匹配所造成的；相应的，解决这些问题的方案是打破 IP 语义的限制。为了将 IP 应用于大规模数据分发场景，解决前文所提到的挑战，研究者提出了很多基于 IP 的适配性的方案。这些方案可以被分为两类：patch(补丁) 方案和 overlay(应用层覆盖) 方案。patch 是在网络层对 IP 进行补充修改，例如 IP 组播^[41]、多路径 TCP^[42]、多路径路由^[43]、流控制传输协议 (Stream

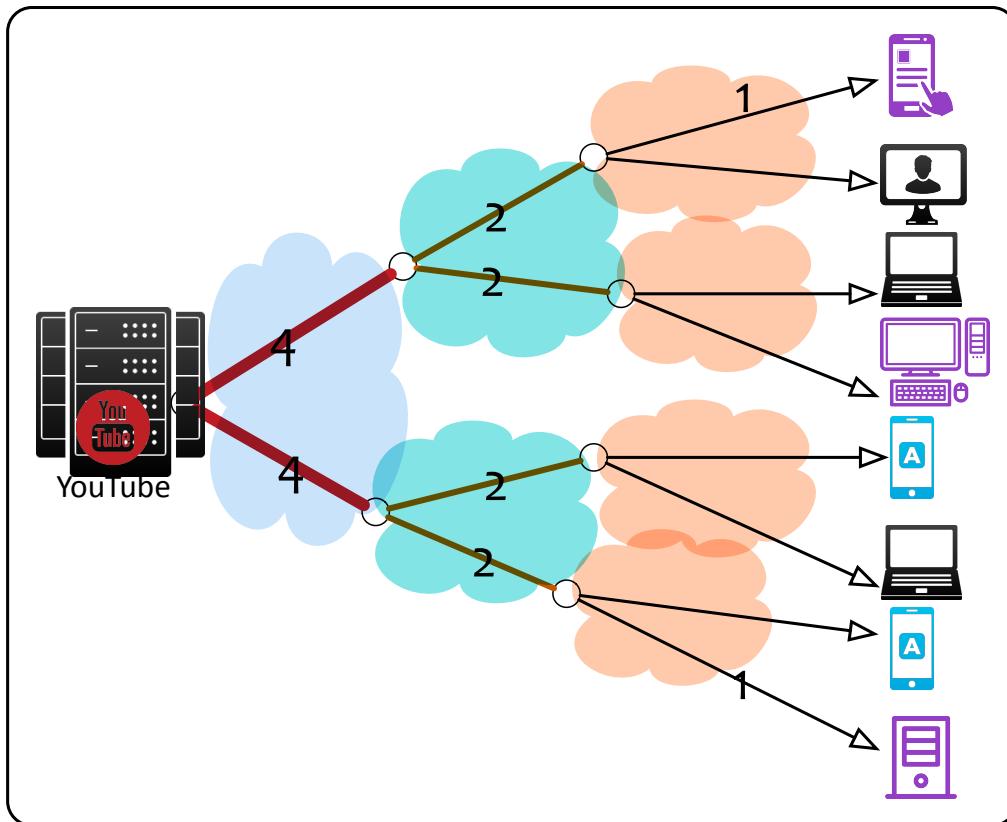


图 2.2 IP 传输服务模型，需要有确定的发送端、接收端、沿着一条路由系统决定的路径传输；在大规模数据传输中，需要为每个终端用户建立一条通信管道，这种 end-to-end 的方式严重限制大规模数据分发的规模和效率。一些利用中间缓存，如 CDN，的办法被用于改善这种模式。

Control Transmission Protocol, SCTP)^[44]、报文拥塞控制协议 (Datagram Congestion Control Protocol, DCCP)^[45]、下一代传输机制 (Transport Next-Generation, Tng)^[46] 等。几乎所有这些工作都在试图打破 IP 命名的语义，用 IP 命名空间（以及端口、序列号等）来标识其它网络实体，如通信组、端到端的路径、基于消息的数据等，从而破除 IP 带来的 point-to-point 的通信限制。但由于兼容性、部署激励等问题，这些方案基本没有得到大规模部署。overlay 方案是在应用层实现的打破 IP 语义限制^①，一些 overlay 方案得到了广泛的应用，如应用层组播 (Application Layer Multicast, ALM)^[47]、CDN、p2p(peer-to-peer)。但是 overlay 方式并没有从根本上解决上层内容传输与网络层 IP 点到点通信语义中间的不匹配，导致了非常多的缺陷，包括安全问题、异构网络间通信不匹配、较高的路径延展、关键链路拥塞等。例如在 CDN 中，需要采用非常精巧的方式来获得终端用户的信任，让他们相信通信对端是目标 CSP 而不是一个中间人，而实际上 CDN 就是中间人，这种天然的第三方角色成为 CDN 中信任管理的根本矛盾，导致了现实网络中非常严峻的安全威胁^[48]；p2p

^① overlay 方案往往工作在网络和应用之间，成为网络中间件的形态。

文件分享系统依赖应用层的路由，这种路由很大程度上是独立于 IP 路由和拓扑的，这导致了很多不必要的域间流量，吞噬了大量可用带宽并影响到其它应用^[49]。patch 和 overlay 方案还有一个共同的弊病，它们都是从一个或有限几个方面出发改进网络数据分发机制，而不能从根本上解决这些问题。即便如此，这些方案本身体现了当前网络真实的需求，反映了网络技术发展最强烈的趋势。

在这种情况下，研究者提出颠覆式的 (clean-slate) 方案，不考虑与 IP 及其应用的兼容性，从整体宏观角度去设计新的网络体系结构。研究者观察到，终端用户真正关心的是他们想要的数据 (what)，而不是数据如何获得 (how)，不是数据存储在哪里 (where)，也不是数据从哪条链路传输 (which)；根据这一最基本的观察，研究者提出独立于存储节点和通信会话对数据进行命名，形成了信息中心网 (Information-centric Networking, ICN) 的概念。ICN 领域已经有比较多的研究，因为对具体机制 (包括命名方式、路由、数据安全等各个方面) 缺乏共识，研究者提出一系列的不同的 ICN 设计方案，如 Data-Oriented Network Architecture (DONA)^[50]，Publish Subscribe Internet Technology (PURSUIT)^[51] (它的前身为 Publish-Subscribe Internet Routing Paradigm, PSIRP^[52])，Network of Information (NetInf)^[53,54](包含前期设计 Architecture and Design for the Future Internet, 4WARD^[55] 和后期提出的 Scalable and Adaptive Internet Solutions, SAIL^[56])，Content Centric Networking (CCN)^[9] 以及根植于 CCN 的 Named Data Networking (NDN)^[57]。其中 NDN 是众多 ICN 方案中研究界最受关注，影响最大的方案。

2.1.1 ICN 的基本设计理念

上文提到的 ICN 出于一个基本的观察结果：“终端用户真正关心的是他们想要的数据，而不是数据如何获得，不是数据存储在哪里，也不是数据从哪条链路传输。”这个观察反映在两个非常普遍的场景，一个场景是拥有海量数据的 CSP 向巨大的用户群体提供服务，在这种情况下大量数据被集中起来，通过 CDN、数据中心这种网络中的内容分发基础设施来实现；另一个场景是单位数据规模小，但总体数量巨大、广泛分布于网络边缘的设备上的数据通信，这些设备往往通过不同的技术接入互联网，如手机、可穿戴设备、车载设备、传感器、物联网等。这两种场景下的数据通信都体现出非常明显的以信息为中心的特点：

- 大规模数据分发基础设施与下层 IP 网络的拓扑分离。用户通过名字，例如 URL (Uniform Resource Locator)，而不是直接通过连接取得数据；而在网络层，用户是与数据分发的基础设施选中的节点进行通信，而不是由用户选择的节点进行通信。在这种情况下，URL 更多承担地是内容标识的角色，而不

是主机的别名。

- 网络边缘设备上的数据通过多种方式进行分发，例如中央控制服务器（最常见于手机、平板上的各种社交软件）；对于有边缘节点组成的网络，如车载网络、传感器网络、延时容忍网络等，一方面很多时候需要通过专门的通信协议实现网络内通信，另一方面常用网关来实现协议翻译，从而保障与互联网中节点通信。在这些情况下，用户只关心数据，而不关心是什么节点产生、存储了什么数据。

鉴于以上的观察，ICN 直接把数据本身作为网络的第一主体，直接对数据命名，而不是接入点，从而消除从数据名到 IP 映射，用更简单、更高效的方式支持上述两个最重要的数据分发场景。通过直接对数据进行命名，数据在不同存储上的差异、不同网络里的差异、不同会话中的差异都被破除，数据的有效性独立于其存储、会话和链路，所以可以充分利用各种资源来增强数据分发的规模和效率，例如多种本地接口（以太网、WiFi、2/3/4G、蓝牙、红外）、多个数据源、终端用户到任意数据源之间的多条路径；同时一些技术，如组播、相同流量在路由器上的聚合、路由器上根据实时流量选择路径、广播（尤其是广播信道上）都能被有效地组织起来，提高数据传输效率、应对网络流量和网络拓扑的动态变化；与此同时异构网络之间的通信也变得更为简单。需要说明的是，很多 ICN 中的元素，如缓存、带状态转发、多路径等并非是 ICN 独创的，而是从现有 IP 的 patch 和 overlay 种种方案中继承而来；ICN 本身不是 TCP/IP 的 patch 或 overlay，但它吸收和集成了很多现有的 IP 里的技术。

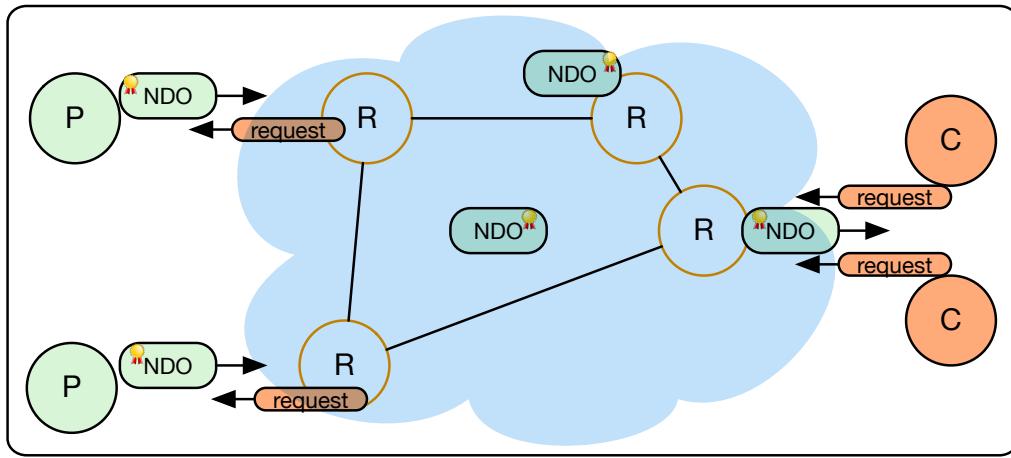


图 2.3 ICN 框架。网络中节点有三种角色：producer (P), consumer (C) 和内容路由器 (R)。两种报文：请求报文 request 和含有签名的命名数据对象 NDO。ICN 通过 request/NDO 之间的交换来实现数据通信，内容在网络中被缓存。

在对数据直接命名的基础上，ICN 在网络层提供直接操作命名数据的原语 (primitive)；作为一个完整的网络体系结构方案，这些原语至少包含两方面功能：数据获取和数据验证。图 2.3 展示了 ICN 网络通信的框架，图中包含最重要的元素，包括请求报文 (request)，命名数据对象 (Named Data Object, NDO)、基于请求/应答的通信模式、网络缓存、以及基于数据的安全模型。然而不同的 ICN 具体设计可能大不相同，有的完全依靠名字来完成请求路由，数据获取等功能；有的同时利用数据名字和 IP（或其它 locator），通过类似于当前 HTTP-DNS-IP 这样的方式去除语义重载，网络中数据名作为内容标识符，IP（或其它类型的 locator）作为路由标识符。在数据获取过程中，缓存普遍被 ICN 方案采用，利用数据访问的局部性来提供效率。

目前已经有为数不少的 ICN 综述文献^[58-62]，故本章不赘述每一个 ICN 的具体机制，仅在下文对当前 ICN 中的数据命名 (第 2.1.2 章)、数据获取 (第 2.1.3 章)、缓存 (第 2.1.4 章) 和数据安全 (第 2.1.5 章) 的各种不同设计做简要总结介绍。对此 ICN 相关设计熟悉的读者，可以忽略这一部分，直接进入 NDN 的介绍 (第 2.2 章)。

2.1.2 ICN 数据命名

数据命名是 ICN 设计中最重要的、最根本的设计要素，所有的 ICN 操作原语都是基于名字。NDO 的名字必须满足两个要求：1) 唯一性，在内容的作用域内能够区分彼此，同一个名字只能对应同样的内容；2) 持续性，名字唯一性应该在不同数据存储、通信会话、各种网络或链路上都能保持。现在主要有两种命名方法：层次化命名和自验证命名。

层次化命名是沿用现有网络中的应用（几乎所有的应用）都采用的命名方法。层次化名字由多个名字部件，按照一定的顺序连接而成。大部分名字部件具有可读性 (human-readable)。URL 就是层次化命名的一个例子，如 “www.google.com/maps/@44.8, -100.9, 5z”。通过层次化的名字，人类可以判定一个 NDO 是否是他想要的，这对于数据的信任管理有非常重要的意义 (第 2.1.5 章)。

自验证名字一般是无结构 (non-structured, flat)，它是通过密码技术对内容进行计算获得的，所以可以直接验证一个内容与它的名字是否匹配，这是这种名字被称作“自验证”的原因。形式最简单的自验证命名是哈希摘要 (hashing digest)，这种命名广泛用于 p2p 系统中。自验证名字的一般形式是 P:L，其中 L 是用于确保唯一性的标签 (label)，P 是用于生成该 NDO 签名对应的公钥的哈希摘要。需要说明的是如果 L 是签名本身，则 L 与内容天然的绑定在一起；否则需要额外的签名来实现这样的绑定^[63]。虽然不能通过 P 直接得到相应的钥匙，但 P 可以用于验证公钥。

自验证命名的主要优势是防止缓存毒害 (caching poisoning)。

这两种命名方法都能实现唯一性和持久性两个目标，但是它们有非常本质的区别。层次化名字沿袭了人“认识”世界万物的方法，所以层次化名字提供了可用性和信任管理方面的优势：即便人只能记住或者区分出非常少一部分名字，这些名字对内容的获取和安全有很大的帮助，可以作为用户进入网络世界的门户和信任源。

自验证名字是纯网络空间里的概念，它与密码技术相关语义紧紧绑定在一起，但是它缺乏与用户相关的可用性和信任管理方面的语义，所以需要额外的依赖来解决可用性和信任管理。例如 p2p 系统中广泛使用自验证的名字（哈希摘要），通常用一个特殊的文件，例如 BitTorrent 里的 torrent 文件，eDonkey 使用的 ed2k 文件，来记录所有目标 NDO 的名字。通过这个特殊文件，p2p 的节点可以下载并验证对应的数据，但是这样的通信包含关于这个特殊文件的两个假设：1) 文件本身安全验证：用户必须通过某种方式安全获得这个特殊文件；2) 文件内容相关性：更重要的是，用户必须相信这个文件里包含他想要的内容的描述。

2.1.3 ICN 数据获取

获取 NDO 可以分成两步，第一步是发现 NDO，即把 request 发送到 NDO 所在的数据源；第二步是传输 NDO，即把 NDO 传输到内容使用者。NDO 一般在传输过程中会被缓存到内容路由器上来满足更多未来的请求。需要说明的是，在 IP 中，数据发送方和接收方都通过 IP 来标识位置，所有数据包都通过 IP 地址来做路由；但在 ICN 中，独立于数据存储和通信会话的名字可以用来标识数据源（名字此时也可以作为路由标识符，但并不拓扑相关），但无法标识数据请求者，所以 request 和 NDO 的路由转发与 IP 中大不相同。

2.1.3.1 内容发现

为了支持 request 转发，一些 ICN 设计使用了非常直接的路由方法：内容产生者先把它产生的数据的名字或前缀宣告到路由系统中，内容路由器收集这些宣告，建立和维护基于名字的路由表，这样内容路由器就可以把数据使用者发送的 request 转发到对应的内容产生者。这种方法叫做基于名字的路由 (name-based routing)，其机制本身与 IP 的转发相似，但名字空间与语义不同。

有一些 ICN 设计引入了新的路由标识符，例如 IP、路径标签，在发送 request 之前先把名字映射到数据源的路由标识符，如 SAIL 采用这种方案。这种方法与 HTTP over IP 类似，URL 作为 NDO 的名字首先被映射成内容生产者的 IP，然后

HTTP request 通过 IP 路由转发到内容生产者。还有一些 ICN 引入新的路由标识符的同时，在网络中建立一套专门用于基于名字转发请求的基础设施转发请求而不需要预先完成做映射，如 DONA 方案中解析处理器 (Resolution Handler)，PURSUIT 中会合点 (Rendezvous Point) 就是这样的实现。值得说明的是在广播信道中，可以直接通过广播 request 来获取内容，不需要更复杂的内容发现机制，而在骨干网中，路由则是必不可少的。

2.1.3.2 内容传输

当 request 遇到对应的 NDO 之后，NDO 需要被传输给数据使用者。对于那些引入新的路由标识符的方案，NDO 可以通过请求者的路由标识符转发，可以是 HTTP over IP 这样的机制，如 DONA, SAIL；或者源路由这样的机制，如 PURSUIT。然而对于没有引入新的路由标识的 ICN 设计，如 NDN，则需另辟蹊径。NDN 依赖于路由器上逐跳的软状态 (hop-by-hop soft state)，内容路由器记录下它转发每一个请求，以及对应的接收和转发端口。NDO 到达一个内容路由器的时候，找到名字匹配的请求，然后沿着该请求的接收端口转发；最终 NDO 沿着 request 转发路径的反方向被传输到内容使用者。软状态还有很多其它的用途：它让 request 和 NDO 形成对称的路径，实现了相同请求的聚合，支持 hop-by-hop 的流量控制，同时消除了转发中数据包的循环等。

表 2.1 展示了几个有代表性的 ICN 设计总结。

表 2.1 各种 ICN 方案对比

	DONA	NDN	PURSUIT	SAIL
数据命名	自验证	层次化	自验证	自验证/层次化
路由标识符	接入点标识符	数据名	路径标签	接入点标识符
数据发现	基于名字的路由	基于名字的路由	基于名字的路由	基于名字的路由 / (名字解析 & 基于地址的路由)
数据传输	基于地址的路由 / 逐跳状态	逐跳状态	源路由	基于地址的路由 / 逐跳状态

2.1.4 ICN 缓存

网络层缓存是 ICN 体系结构中非常重要的一个特性。通过在网络层对数据直接进行命名，ICN 可以支持透明的（不需要应用作任何处理）、通用的（适用于所

有的应用)、细粒度的全局缓存,这对大规模数据分发有很重要的意义。现有研究显示缓存可以明显地提高网络性能,例如提高吞吐量,降低服务器压力,缩短传输延时,缓解网络拥塞。直接对数据进行命名使得 ICN 在网络层天然支持缓存。实际上,这种以存储换带宽的背后有更深层次的经济考量:存储器的价格远比带宽价格下降的快,网络设备的存储空间可以很廉价地扩展,因此各种 ICN 设计里普遍采用缓存。但需要说明的是缓存是 ICN 设计中的一个重要部分,但远不是 ICN 的全部;通过对 ICN 的整体描述可以发现,ICN 在网络层直接操作被命名的数据,是对网络体系结构根本性的改变,而不仅仅是增加了缓存支持。

对 ICN 中缓存的研究非常多,主要包括下面三个方面,第一个方面是缓存替换策略 (caching replacement policy),是指充分地利用有限的缓存空间,用更合适的新内容替换原有内容的替换策略, Yi Sun 等^[64] 研究了不同缓存替换策略对应用的影响,如 LRU, LFU, FIFO; 该研究发现简单的 LRU 策略即可取得非常好的缓存效果。第二个方面是缓存部署策略 (caching deployment strategy),是指缓存应该放置的合适位置和大小。Seyed Kaveh Fayazbakhsh 等^[65] 研究了一个来自 CDN 的数据集,得出结论认为边缘的路由器减少了绝大部分可以减少的流量。而 Claudio Imbrenda 等^[66] 根据一个 ISP 的数据认为全局的缓存更为有效。这两个相反的结论有它的理论基础:一个完全符合 zipf 模型的请求分布,因为低频率的请求非常多,所以需要非常大的缓存空间才能取得较大的缓存命中率;而在韦伯分布的情况下,任意请求大的缓存命中率都可以通过增加缓存大小有效达到^[67]。由此可见,请求的分布是缓存部署策略最根本性的因素,而在目前阶段,文章^[65,66] 中展示的数据集都有一定代表性,但还不是真正的 ICN 网络的流量。

第三个方面是缓存决策策略 (caching decision policy),是指应在缓存系统的节点决定是否缓存某个 NDO 的决策,这也是缓存研究中的热点。LCE(leave copy everywhere),是许多 ICN 设计中的默认缓存决策策略,即当对象返回时,沿途的所有节点都缓存对象;但这种方式容易造成缓存冗余,即相同的对象在多个节点同时存有副本,降低了缓存系统所能缓存内容的多样性。为了降低缓存冗余、提高系统的缓存多样性,需要缓存节点之间进行简单而有效的协同。根据系统机制是否需要节点间通信,又可以分为显式协同和隐式协同。

显式协同需要相关节点之间交换信息来决定是否缓存一个对象,常见的有全局协同、路径协同和邻域协同。全局协同继承于 CDN 中的技术,但由于 ICN 中缓存节点数据、内容数据量与动态性都远多于 CDN,这种机制需要进行改进才能为 ICN 所用,如基于哈希的全局协同^[68,69]。路径协同是指在 NDO 从命中点(请求遇到数据的节点)到数据使用者之间的传输路径上的节点进行是否缓存的决策,

如协同路径缓存 (Cooperative En-Route Web Caching, CERC)^[70,71], 这类方法要在请求报文中携带沿途节点状态。邻域协同是指一个节点在邻域范围内协同，如基于哈希的机制 CINC^[72]。

与显式协同不同，隐式协同无需了解其它节点的信息，不需要或者只需要很简单的节点间通信，根据节点自身的状态做决策。这种方式机制简单、扩展性好，其中典型的隐式缓存协同方法有下面几种^[73]:

- LCE (leave copy everywhere): NDO 传输的沿途都缓存该内容，这是很多 ICN 设计默认的方案，节点之间几乎没有协同；
- LCD (leave copy down)^[74,75]: 缓存命中时，仅在命中点的下游邻节点缓存该内容，避免大量冗余缓存，同时也将内容移到使用者附近；
- MCD (move copy down)^[74,75]: 缓存命中时，仅在命中点的下游邻节点缓存该内容，并删除命中点上的缓存拷贝；
- Prob (copy with probability)^[74]: 沿途节点按照一定的概率决定是否缓存一个 NDO；
- RCOne^[76]: 在沿途节点中随机选择一个节点缓存，对于层数为 L 的层次化缓存来说，该方法等价于概率为 $1/L$ 的 Prob 方法；
- ProbCache^[77]: 沿途节点按照一定概率决定是否缓存一个 NDO，缓存概率反比与当前节点与命中点间的距离。

有关于各种缓存研究的具体细节，综述文章^[73,78-80] 有专门的介绍与分析，本文不再赘述。

2.1.5 ICN 数据验证

由于 ICN 中的数据通信在时间和空间上都脱离了数据存储和通信会话的限制，ICN 中的数据安全需要适配这种新的通信模式，让数据验证也突破数据存储和通信会话的限制。数据导向的安全机制通过把数据验证信息所需要的签名（及验证该签名的证书或其索引）直接包含于 NDO 中，允许任何收到该 NDO 的人、在任何时候来验证这个 NDO。在数据导向的安全机制中，NDO 的三个特性需要验证^[63]:

- 1) 有效性 (validity), 包含传统意义上的完整性 (integrity) 和真实性 (authenticity)，表明 NDO 经由一个生产者产生之后，是完整的，传输过程中没有经过篡改；
- 2) 溯源合法性 (provenance), 该 NDO 的生产者是可以合法的、是为数据使用者接受的；
- 3) 相关性 (relevance), 该 NDO 的确是消费者想要获取的数据。

Ali Ghodsi 等^[81] 提出可获得性 (availability) 也作为安全的一个指标，这里可获得性主要是指防止因缓存毒害，而造成内容层面的拒绝服务 (denial-of-service)。上

面提到的四个目标，有效性、溯源合法性、相关性、可获得性分别处于不同层面：有效性的验证是完全语义形式的，它仅仅是验证 NDO 是否被它宣称的那个私钥加密，这样的验证完全是密码技术的计算，所有的 NDO 验证遵循同样的规则。但是溯源合法性和相关性则完全不同，它们是内容使用者和产生者之间的约定，不同的应用验证规则可以是完全不同的。针对同一个 NDO，不同的应用、不同的用户对该 NDO 的溯源合法性和相关性验证可能不同，其中一方面涉及到应用使用的信任模型 (trust model)，另一方面则涉及到数据使用者的信任锚点 (trust anchor)。而可获得性更倾向于内容路由器的一种功能，而不像其它三个特点一样是验证 NDO 本身的属性；但它是通过让路由器拥有验证 NDO 的能力来实现。

对于自验证命名，P:L 形式的命名极大简化了数据验证，但这种情况下数据验证需要两个隐含条件：1) 简单的安全模型，即该 NDO 必须是由 P 对应的私钥签名，中间没有任何层次；2) P 同时也充当了信任锚点。

2.2 命名数据网 NDN

NDN 是 ICN 研究领域最有影响力、研究最活跃的设计方案之一，在世界范围内有广泛的研究群体。它的基本设计根植于 Van Jacobson 2009 年在 PARC (帕拉托研究中心，现为施乐 Xerox 公司的子公司) 提出的 CCN (Content Centric Networking)^[9]，美国自然科学基金 (NSF) 于 2010、2013 年连续两次资助了 UCLA 教授 Lixia Zhang 领导的 NDN 项目^[9,57,82]，Van Jacobson 是 NDN 项目的体系结构架构师。目前 NDN 与 CCN 分别由不同的单位独立研究，NDN 项目成员包含多所大学，偏学术研究；CCN 在 PARC 主导下，相比而言更关注相关技术商业化，包括专利、软件授权等。

图2.4 展示了 IP 与 NDN 的体系结构对比，两者都具备“窄腰”(narrow waist)，IP 的窄腰是以接入点命名的 IP 数据包，而 NDN 的窄腰是以应用层名字直接命名的 NDN 数据包；NDN 还有安全层和策略层，安全层是通过基于数据的安全模型来保障数据安全，策略层则是为了充分利用多路径、多接口的优势，实现更灵活、高效的数据获取。

NDN 把层次化命名、基于数据的安全、带状态的数据转发、基于名字的路由、缓存这些革命性的设计元素都放到网络层，通过在网络层直接操作数据，把通信的焦点从 where 移到 what，实现以信息为中心的网络体系结构设计。而这些设计元素被广泛应用于当前网络中，如 CDN 和 p2p 中的直接操作命名数据，DNS 中的层次化命名空间，RSVP (resource Reservation protocol)^[83]，Multicast^[84,85] 中的带状态的数据转发等。NDN 团队同时也强调 NDN 的概念和思想并非凭空而来，而是受到很多设计的启发^[10]，包括 NETBLT (NETwork BLock Transfer)^[86]，RSVP^[83]，

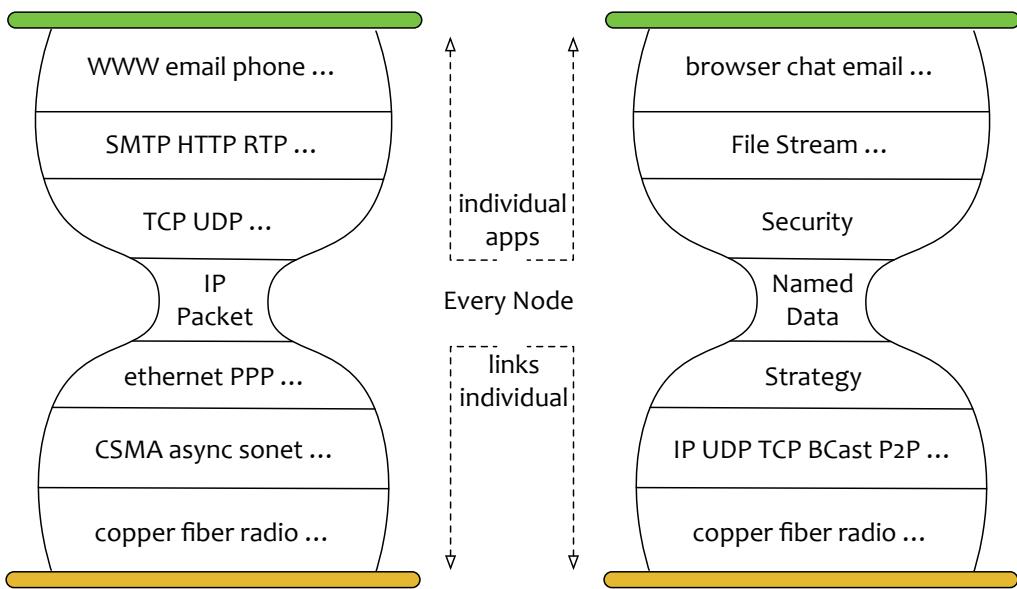


图 2.4 IP 与 NDN 体系结构对比，两者都有窄腰，窄腰以上支持丰富的应用，窄腰以下允许不同的链路服务。

Multicast^[84,85], web caching^[87] 等。总体来说，NDN 将很多原本是 TCP/IP 中 patch 和 overlay 方案的一些设计元素组合起来；这些技术本身体现了当前网络真实的需求，反映了网络技术发展最强烈的趋势，NDN 对这些技术的组合不是简单的生搬硬套，而是通过对命名数据进行操作创造性的将这些元素组合起来，并提供信息为中心的网络层操作原语，实现了精巧、完整、高效的解决当前网络下数据通信的方案。与此同时，NDN 在可靠性和安全验证上也继承了“端到端原则”。

2.2.1 报文格式和节点模型

NDN 设计中有两种报文，请求报文命名为 Interest 报文，而 NDO 则定义为 Data 报文，两种报文中各个域格式并非固定，而是通过 TLV (type-length-value)^[88] 形式来定义（见图 2.5）。

两种报文中都以 Name 域作为标识，NDN 通信依靠两者的名字匹配进行。在两种报文做名字匹配采用最长前缀匹配的方法，即 Interest 的名字与 Data 名字相同或为其前缀；Interest 中还用 Selector 域来选择目标数据，包括是否必须从数据源中获得（排除缓存中的数据的干扰），最长前缀匹配时对名字的限制，目标数据公钥摘要哈希值等；Nonce 域是一个随机数，用于判断相同名字的 Interest 是同一个 Interest 在网络中循环，还是不同用户请求相同的数据（Nonce 域中包含不同的随机数）；InterestLifetime 域则是 Interest 的生命周期，超过这个生命周期，Interest 及它转发过程中在路由器上建立的软状态应视为过期无效。Data 除包含应用数据外，还含有一些描述目标的可选域，如判断 Data 尚未过期的时间限制，公钥摘要

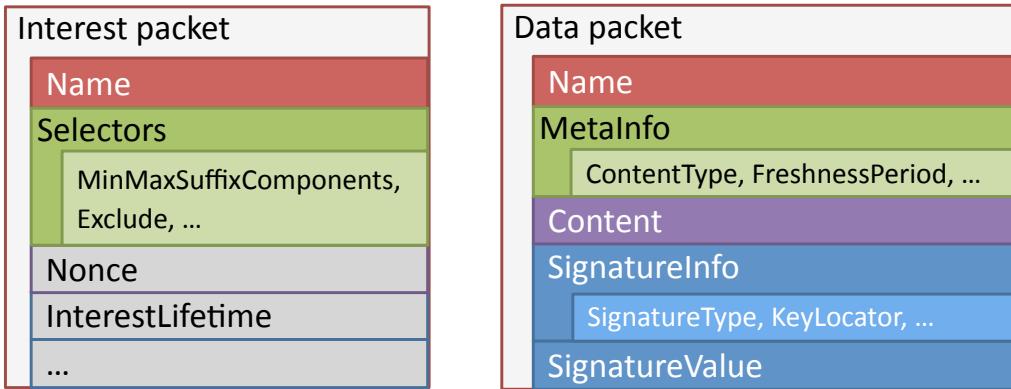


图 2.5 NDN 的报文 TLV 格式：NDN 中有两种报文，Interest 和 Data，两者都含有 Name 域作为报文标识。

哈希值；Data 中含有签名部分，包括对签名的比特位值，以及签名所对应公钥信息，通过这些信息可以获得验证该签名所需要的证书。

NDN 路由器中需要维护三个数据结构来完成 Interest / Data 交换 (图2.6)：用于 Interest 转发的 Forwarding Information Base (FIB)，用于记录转发路径的 Pending Interest Table (PIT) 和用于缓存数据的 Content Store (CS)。

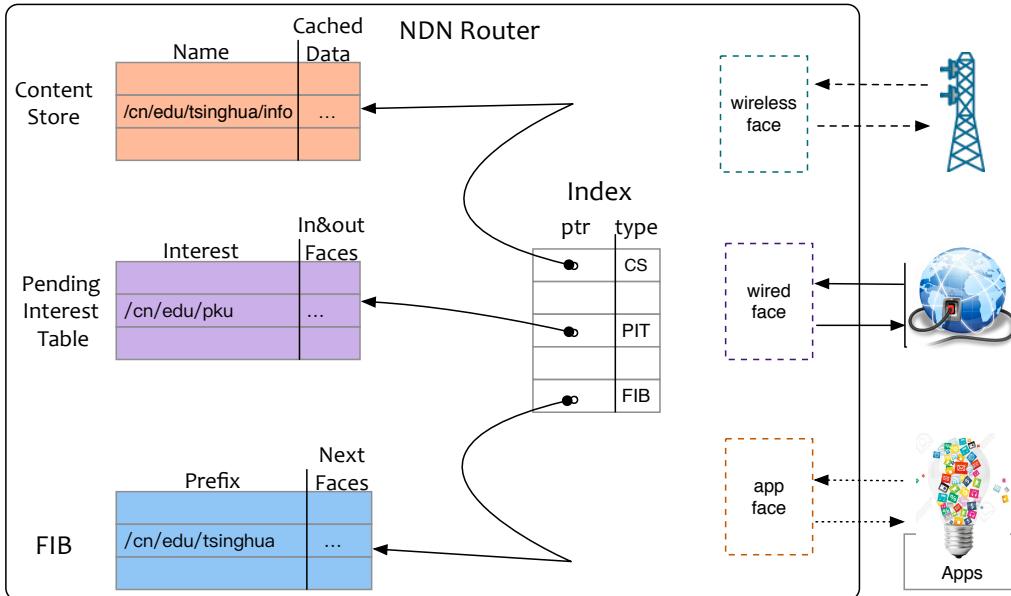


图 2.6 NDN 节点模型：NDN 节点中至少需要维护三种结构，CS，PIT 和 FIB，NDN 节点支持不同类型的端口。

NDN 通信是由 consumer 驱动的，consumer 首先发送包含目前数据名字的 Interest 到网络中，每个路由接收到 Interest 后查看是否为在网络中循环的报文 (检查 Nonce 域)，如果是则直接丢弃；否则：

- 首先查找 CS，如果能找到匹配的数据则直接返回该数据；

2. 否则继续在 PIT 中查找，如果能找到对应的表项，则表明此前已经转发了同样的请求（请求相同的目标数据，但不是同一个 Interest 报文，可能是由别的数据请求者发送的），此时只需要把该 Interest 的接收端口加入到对应的 PIT 表项中，等待 Data 返回；
3. 否则继续查找 FIB，按照对应的转发端口转发这个请求，并创建相应的 PIT 表项记录下相应的 Interest，及其接收和转发端口；并启动定时器（等待时间由 InterestLifetime 决定，默认值为 4 秒），等待 Data 返回。

Data 是由 producer 生成，通常 producer 拥有一对公/私钥；Data 在生成的时候就通过公钥签名，并将对应的公钥信息也包含在 Data 中，以便 consumer 验证数据。当 Interest 遇到对应的 Data 时，则按照 PIT 表项记录的 Interest 接收端口转发，沿着 Interest 路径的反方向最终到数据请求者。

下文将针对命名、基于名字的路由、带状态转发、基于数据的安全四个方面介绍 NDN 的具体设计。

2.2.2 层次化命名

数据命名是最为重要的设计，它关系到数据的获取与安全；NDN 要求数据名字独立于数据存储和通信会话。NDN 沿用了目前互联网上广泛使用的层次化命名空间。层次化命名是人类认识世界万物最自然的方式，对世界万物进行层次化归类并予以标识。生物学分类，网络域名，电脑中文件系统等都是按照这样的方式设计；而 DNS 命名空间则是网络应用中使用最广泛的。

NDN 中的层次化名字有多个名字部件连接而成，名字部件之间用"/" 作为分隔符，例如 “/com/google/www” 。数据名字可能还包括版本号 (version number) 和分段号 (segment number)，以此来区分不同的版本和分段。这样的命名使数据成为不可变对象 (immutable object)，每个名字都可以标识唯一的数据。数据存储和通信会话和不可变的数据对象，使得 NDN 能够天然地支持网络层缓存。

层次化命名有利于缓解路由扩展性，路由条目以名字前缀的方式记录在 FIB 表中，而无需以每个名字的粒度创建条目；同时层次化名提供了可读性，人类可以记住部分名字，并把网络空间中的名字与真实世界的实体 (real-world identity, RWI) 绑定在一起，这对数据验证有非常重要的意义。即便人只能记住或者区分出非常少一部分名字，这些名字对内容的获取和安全有很大的帮助。有些名字成为用户进入互联网的门户，例如名字 “/com/google/www” ，它把网络用户引向 Google 公司的网络搜索服务，通过这个搜索服务用户可以获得几乎任何网络上公开的内容；有些名字同时也承担着信任源的作用，例如 “/cn/boc/www” 是中国银行的官方网页 Data

D 的名字，网页中包含某个超链接对应的 Data $D_{hyperlink}$ 的名字及其它内容描述信息（如内容哈希值、公钥摘要）；在这种情况下，用户在名字 “/cn/boc/www” 帮助下验证 Data D，再通过 D 中对 $D_{hyperlink}$ 的描述信息来简化 $D_{hyperlink}$ 的验证，此时 “/cn/boc/www” 充当了信任源（这种叫基于证据的信任模型，见第 2.2.5 章）。

需要说明的是，根据应用的需要，名字还可能包含多种语义，可以是控制命令，或者一个接入点，NDN 的命名依然保留了这种能力。

2.2.3 基于名字的路由

每个内容生产者都可以宣告相应的内容前缀到路由系统中，路由器通过路由协议收集路由信息，建立 FIB 表。由于名字并不是拓扑相关的，NDN 存在多个 producer 宣告相同的前缀的情况，NDN 的路由系统也保存转发到所有数据源的信息；而且针对每一个数据源，NDN 路由系统也支持多路径。NDN 路由器收到 Interest 后，通过名字的最长前缀匹配找到最佳的 FIB 表项，并转发 Interest，这被称为基于名字的路由。

需要指出的是，一个 producer 在没有前缀 N 下的所有内容情况下，依然可以宣告 N。因此，一个请求可能被路由到一个目标数据不存在的服务器上，本文把这种情况称为“不准确路由”。不准确路由避免过细粒度的路由宣告，简化了数据管理，增加了名字前缀聚合的能力。NDN 中通过网络层 NACK (Negative Acknowledgment) 和自适应转发机制来修正不准确路由^[89,90]。

即便名字聚合可以大量减少路由条目，路由扩展性也是最令 NDN 研究者忧虑的问题之一。根据当前顶级域名的数量的估计值^① NDN 名字前缀将达到 6×10^8 ^[29]。考虑到目前硬件价格、查找速度等因素，DFZ FIB 只能保存一部分名字。在这种情况下，保证全网路由可达性成为一个重要挑战。Alexander Afanasyev 等提出 SNAMP^[5] 方案，把一部分名字放入 DFZ FIB 中的名字作为路由锚点 (routing anchor)，其它的名字安全地链接到路由锚点，然后通过路由锚点进行路由。一种名为 LINK 对象 (图 2.7) 的特殊数据报文承载了这样的链接信息，它附着到 Interest 上一起发送；当路由器发现无法通过 Interest 的名字转发这个请求时，就按照 LINK 对象中的路由锚点来转发请求。

需要说明的是，在这样的路由转发模式中，路由锚点依然是普通的名字，可以与拓扑无关，一些热点的名字前缀，如 “/com/youtube” 很可能会装载如 DFZ FIB 中。在 NDN 通信中，LINK 对象中的路由锚点只解决转发问题，而在缓存中数据匹配，PIT 表项匹配中，依然是用原名字，这样的设计不改变 NDN 最基本的基于

^① 顶级域名大约有 2×10^8 个，以 3 作子前缀宣告系数^[29]。

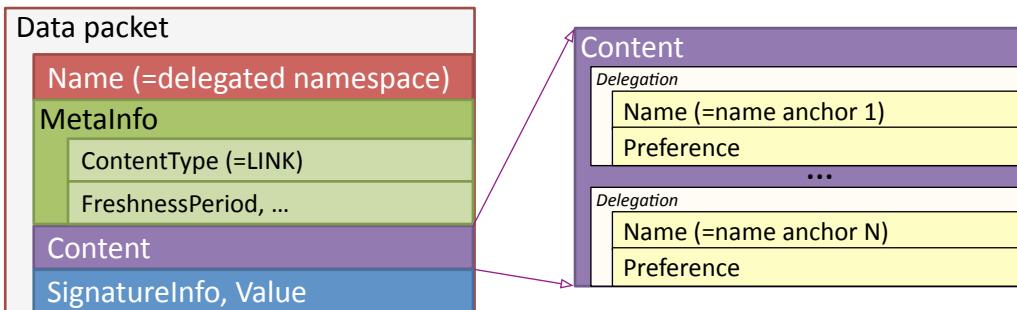


图 2.7 LINK 对象 TLV 格式 (NDN Data): 这种数据包安全地把一个名字“链接”到另外的名字，可以用来解决路由扩展性，支持移动性。

名字匹配的通信模式。

2.2.4 带状态的路由转发

数据名可以作为转发 Interest 的路由标识符，把 Interest 指引到 producer 上；然而数据名无法把对应的 Data 引导到 consumer 上。因此，Data 的报文转发迥异于 IP，需要另辟蹊径。NDN 通过 PIT 记录状态，即转发的 Interest 及其接收端口、转发端口来转发 Data。当 Data 被返回的时候，这个状态被删除；并且这个状态本身也有时间限制，如果时钟超时，该状态也被删除。这样的状态被称为“软状态”。NDN 通过逐跳的软状态 (hop-by-hop soft state) 来转发数据，但软状态的作用远不止于此。

首先，这样的通信模式使得节点只能返回被请求的数据，而不能主动推送数据，这种“拉模式”通信杜绝了攻击者主动推送垃圾数据；第二，软状态可以消除 Interest 转发循环，从而消除了 Interest 路由和转发中的后顾之忧，有利于路由器采用积极地采用各种灵活的转发策略（图2.4策略层）；第三，这种 Interest / Data 交换形成了对称的数据流，通过观察这个对称的数据流可以准确地、实时地判断链路/路径状态；第四，在判断链路状态的情况下，同时根据路由信息、历史和实时的链路状态信息给链路判定优先级，而且这种优先级可以是在每个前缀级别^[89]，当一个 Interest 有多个转发端口时，可以根据优先级信息来选择最佳的转发端口；最后，如果某个路由器上检测到之前转发的 Interest 无法取得数据^[89]，它可以快速向另外一个或多个端口再次转发 Interest 来恢复数据获取，而不用把这个任务留给 consumer 来完成而造成相对较大的延时和开销，这就是中间节点上的“快速失败恢复”^[89]。

2.2.5 基于数据的安全模型

NDN 的通信在时间和空间维度分离了数据使用与其产生，数据验证也需要有这样的能力，而不能依赖于数据存储与通信会话。因此 NDN 在 Data 生成时，直接对 Data 中的名字、内容（已经对应的内容描述）进行签名，同时允许以任何方式获得 Data 的 consumer 来验证数据，这种直接保护内容本身而不是通信管道的安全模型被称为基于数据的安全模型。每个 Data 中含有签名域（图2.5），既包含签名本身，同时也包含用于生成共签名的私钥对应的公钥证书的信息 (KeyLocator)；公钥证书本身也被封装成一个数据包（图2.8），一般来说 KeyLocator 中放入的是该证书数据包的名字。需要说明的是，NDN 中数据产生和使用在时间和空间上解耦合，这意味着当数据被 consumer 验证时，对应的 producer 可能已经下线，这时候需要稳定的证书仓库来提供数据验证所需要的证书。

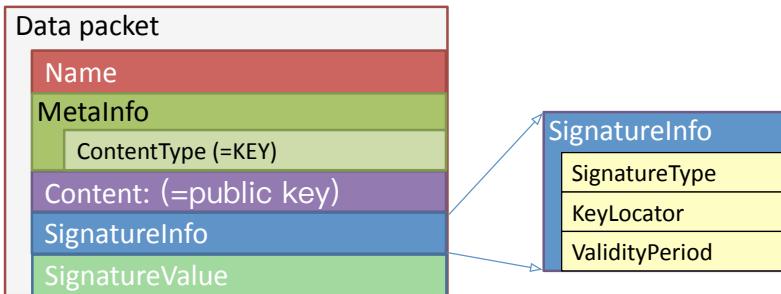


图 2.8 NDN 公钥证书 TLV 格式 (NDN Data): 公钥部分作为数据包的内容域，用于颁发该证书的证书名字放在签名域，证书作为普通 NDN Data，它的获取和验证可以采用正常的 NDN 通信，极大地增加了该证书的可用性。

如第 2.1.5 章所说，NDN 中需要对 Data 进行三个方面的验证：有效性，溯源合法性 (provenance) 和相关性。Diana Smetters 等^[63] 提出三步来验证网络中的内容，1) 验证该 Data 由某个特定的私钥签名；2) 验证这个私钥，确定该私钥的拥有者，也就是该 Data 的产生者；3) 判断由这个生产者来生产该 Data 是否可以接受。在这三步中，第一步是决定有效性；后面两步确定溯源合法性，标明该数据来自一个可以接受的 producer；在前两部的基础上，数据的名字确定了相关性。

在手动或者自动配置之后，数据的验证可以完全自动化；这些配置包括为应用的配置验证策略（应用安装的时候自动配置），为名字和应用选择信任锚点。在这个过程中，数据验证策略和信任锚点的配置至关重要，NDN 允许应用定制数据验证策略，可以是层次化的，web-of-trust 或其它应用决定的模型；同时也允许 consumer 在任意长度的名字前缀的粒度下配置信任锚点。NDN 这种灵活性对于应用有非常重要的意义。Yingdi Yu 等提出并实现了一个将 NDN 中数据验证自动化

的方法^[91]，极大地简化了 NDN 中数据验证，增强了 NDN 安全机制的可用性。

除了上述数据验证模型以外，NDN 还支持一种简化的数据验证模型，基于证据的安全模型 (evidence-based security model)，这个模型要求预支目标 Data 的一些关键属性，例如 Data 本身的哈希值、对它签名的公钥或其哈希值等。用这些属性作为“证据”来验证数据，则不需要上面那么多复杂的步骤。基于证据的安全模型在 p2p 系统里使用广泛，p2p 系统里用一些特殊文件，例如 BitTorrent 里的 torrent 文件，eDonkey 使用的 ed2k 文件，记录目标数据的哈希值；当节点获得数据后验证其哈希值是否匹配即可。在 NDN 里，只需要一个信任源头，如某个通过验证的某网页数据，这样的一个信任源头中可以包含其它数据验证所需要的“证据”，如该网页中其它超链接 (Data 的名字)，则可能的证据是该数据对应公钥的哈希摘要，甚至是 Data 本身的哈希值。consumer 可以直接通过验证这些证据来达到验证目标 Data 的目的，这样内容验证将变得非常简单。NDN 设计中提供了对基于证据的安全的支持，如在 Interest 定义了 PublisherPublicKeyDigest (PPKD) 可选域，可以与 Data 中的 KeyDigest 匹配，这与自验证名字中的 P 作用是一样的；Interest 和 Data 中还定义了 ImplicitSha256DigestComponent 可选域，可用哈希值来验证数据。

2.3 本章小结

本章包含部分内容，即 ICN 和 NDN。首先本章从互联网演化的角度阐述了 ICN 兴起的原因，即当前网络主要的挑战从建立连接转移到数据分发，而 IP end-to-end 的语义限制了数据分发的规模和效率，而基于 IP 的 patch 和 overlay 方案均有缺陷，并能从根本上满足互联网数据分发的要求，于是研究者提出 ICN 的设计，以信息为中心，而不是网络节点为中心，从而打破 end-to-end 的限制。网络层的缓存支持是打破 end-to-end 限制的一个重要标志和优势，它意味着数据通信中，数据的产生和使用在时间和空间上分离；另外，这种以存储换带宽的思想背后还有着非常实际的经济考量。

从网络体系结构的角度看，NDN 至少包含数据命名，数据获取（包含两个过程，即请求路由，数据传输）和数据验证三方面的设计。NDN 将很多 IP 的 patch 和 overlay 中的设计元素创造性的组合起来，提供了这三种功能的设计：1) 沿用当前 DNS 层次化命名，这是当前应用普遍使用的方法；2) 通过名字本身（而不要引入其他路由标识）直接路由 Interest；3) 通过名字匹配找到对应的 Data；4) Data 沿着请求报文的反向路径返回给 consumer，并在中间节点缓存；5) consumer 通过验证 Data 的签名来保障安全性。这个过程“简洁”而“自然”。

第3章 nDNS：NDN名字解析系统

如第 1.3.1 章所述，NDN 目前已经鉴别出的挑战，如路由扩展性解决方案 SNAMP^[5]，移动性支持方案 Kite^[7] 和数据验证（第 2.2.5 章），都依赖于一个名字解析系统。本章的目标就是为 NDN 网络建立这样的一个系统。

当前比较有代表性的名字解析系统有 DNS (for IP)^[31,32] 和 GNS (for Mobility-First)^[92]。DNS 与 GNS 这两种名字解析系统有非常大的区别。DNS (Domain Name System) 定义了层次化的应用命名空间，采用树形结构来组织资源数据，通过自顶而下的迭代查询来解析数据，利用缓存和数据冗余来应对大量请求；DNS 设计简单，其优势在 IP 网络中得到证实。GNS(Global Name Service) 是为 MobilityFirst^[93] 体系结构量身打造的名字解析系统，它使用无结构的名字来标识数据，根据名字哈希的方式来组织和解析数据，主要利用数据冗余来应对大量请求，优势是支持快速的、大规模的记录更新，表 3.1 对两者的一些特性进行了比较。

表 3.1 DNS v.s. GNS

	DNS	GNS
数据命名	层次化	无结构
数据组织	树形	无结构
数据解析	自顶而下	利用哈希定位
系统扩展性	缓存和数据冗余	数据冗余

考虑到 NDN 的一些核心设计，如层次化数据命名，网络层缓存支持，DNS 显然更适合 NDN；并且，DNS 在过去 30 年的成功，也证明树形的数据组织形式和自顶而下的查找方法能很好的解决这样一个整个网络基础设施面临的扩展性和鲁棒性等方面的挑战。因此本章选择 DNS^[31,32] 以及它的安全功能扩展 DNSSEC^[33–35] 作为主要参考对象设计了 NDN 网络中的名字解析系统，这也是称之为“DNS for NDN”的原因。根据 DNS 设计 NDN 的名字解析系统并不是全新的构想，Alexander Afanasyev 于 2013 年提出 NDNS (NDN DNS)^[94,95] 的概念和设计，把 DNS 迁移到 NDN 中，本章工作是在 NDNS 上进一步的研究。相比于 NDNS，nDNS 在设计的各个方面均有所改进，如 1) 设计了新的名字解析机制（核心设计），能够自举的解决自身请求路由的问题；2) 增加了新的数据验证模型，提高了数据验证的灵活性和效率。本章将在第 3.1.2 章介绍 Alexander Afanasyev^[94] 的工作与本章工作的区别。

通过一个名字来获得对应的内容这是 DNS 提供的服务，内容可能是 IP 地址、公钥证书、服务记录或其它任意数据；到目前为止 IANA 给 DNS 分配了 80+ 多个记录号^[96]，存储不同的数据，服务于不同的应用。DNS 在 IP 中的广泛应用已经证明它是网络运行和管理不可或缺的基础设施。本章对 nDNS 的期望绝不单是解决已经认识到的这三个问题，还有其它潜在的问题同样需要依赖 nDNS；某种程度上说，当前 DNS 中 80+ 多种记录号对应的问题都可能出现在 NDN 中，都需要通过 nDNS 来解决。因此，作为一个名字解析的基础设施服务，从功能上说，nDNS 必须具备以下特性：

- 通用性：允许各种不同的应用按照它们各自的需求存储不同类型的数据；
- 扩展性：提供可以为数量庞大的用户提供数据查询；
- 鲁棒性：非常稳定，作为网络基础设施必须具备高度可用性；
- 安全性：必须安全地提供名字解析服务，查找得到的数据必须可以验证。

nDNS 继承了 NDN + DNSSEC 中的很多基本概念和设计原则，例如层次化的命名空间和数据组织方式、域名、名字域、递归查询和迭代查询、对记录进行签名等。nDNS 与 DNS + DNSSEC 有很多相似点，都是通过自顶而下的名字解析获取内容，直接对内容进行签名，依靠缓存来增强扩展性，这些相似点似乎意味着 nDNS 的设计是非常简单和直接的。但是我们也不能小看它们底层网络的区别，DNS + DNSSEC 是在应用层实现那些功能，并且依靠 IP 来实现请求和应答的传输；NDN 在网络层支持缓存、数据签名与验证、名字路由支持多路径和多源，这些特点简化了 nDNS 的设计；与此同时，NDN 要求请求与回复的数据必须匹配，所以名字服务器不能使用最佳的“答案”来回复一个请求，而只能是对应问题的答案，所以 nDNS 中的递归查询不能一直发送同一个问题，这导致了额外的复杂性。

本章将按如下方式组织：首先介绍本章相关背景，包括 DNS，并简要分析迁移到 NDN 中的挑战（第 3.1.1 章），然后介绍 nDNS 与本章工作的联系与区别（第 3.1.2 章）；之后介绍 nDNS 的具体设计（第 3.2 章）；再讨论一些 nDNS 数据更新和同步，缓存解析器的作用，nDNS 在 NDN 网络中的特殊角色和作用（第 3.3 章）；此后介绍 nDNS 原型、部署与实验（第 3.4 章）；最后总结本章（第 3.5 章）。

3.1 相关背景：现有名字解析系统

3.1.1 DNS 设计

DNS^[31,32] 在 IP 网络中广泛应用，向多种不同的应用提供了不同的信息，例如广泛应用的 WWW 中的域名到 IP 的映射，email 应用中 email 地址到主机的映射等。DNS 首先维护了一个层次化的域名空间，这个域名空间里的元素（也就是域

名) 构成树形结构, 拥有一个共同的根域名, 层次化名字通过“.”区分层次。当前绝大多数应用都是直接利用 DNS 定义的层次化命名空间, 从而打破了 IP 对应用的限制。DNS 中存储的信息被称为“资源记录”(resource record, RR), RR 与域名树上的节点关联在一起。RR 还有几个属性, 如类型 (type)、TTL、类别 (class)^①。并按照它们关联的节点划分为不同的名字域 (zone), 名字域是名字服务器上 (name server) 信息管理的基本单元, 一般来说, 每个域需要有多台名字服务器。网络中部署还需要部署缓存解析器 (caching resolver), 顾名思义, 缓存解析器从名字服务器上取得数据, 并缓存起来, 以便应答未来同样的信息查询。终端则运行则终端解析器 (stub resolver) 的进程 (往往通过调用系统 API 来实现), 如图3.1所示。

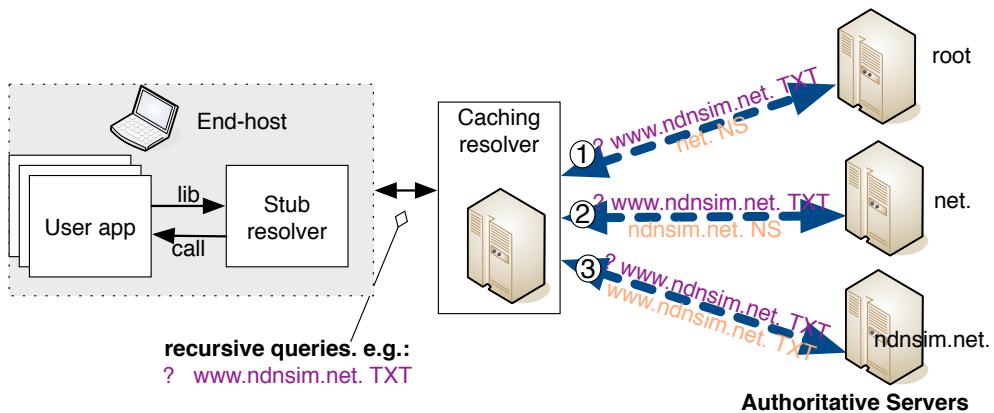


图 3.1 DNS 解析范例：在迭代解析过程中，同一问题发送给不同的名字服务器，而名字服务器根据自身的数据选择最好的答案来回复，由 IP 来保证问题/答案的可达性。

需要说明的是, 缓存解析器必须知道它要发送请求的名字服务器的 IP。缓存解析器配置中包含一组根域名服务器的 IP, 而在域名层次中, 父名字域中把子名字域服务器的名字保存为类型为 NS (name server) 的 RR, 并通过 glue RR, 把服务器名字及其 IP 作为下一级名字域的引用 (referral) 作为迭代查询的应答。换句话说, 根名字域中存放在所有 TLD (Top-level domain) 的名字服务器的 IP, 而 TLD 的服务器中则存放在 SLD (Second-level domain) 的名字服务器的 IP, 这样从根到叶子一直延续下去。

在 IP 网络中所有的名字服务器的地址都认为是可达的 (nDNS 中并不是所有名字服务器对应的域名都可路由), 所以缓存解析器可以使用至上而下的迭代查询机制, 从根名字域开始来解析任何名字; 在一个目标名字解析过程的每一步迭代中, 缓存解析器用不同的目的 IP 地址, 发送相同的请求 (目标问题) 到不同的名字服务器中。名字服务器可以根据自己数据库中的信息直接返回目标信息, 或者

^① 在 DNS 设计中, 三元组 (name, class, type) 定义了一个 RRset; 而实际应用中 class 的值几乎都为 1 (表示互联网上的资源)。

一个指向它子名字域的引用（包含主机名及其 IP 地址）。当目标信息被返回，或者被证明不存在，整个迭代过程结束。再次强调，整个迭代过程中不断询问同一个问题，由不同的 IP 地址决定谁来回复。图3.1展示了一个解析域名“`www.ndnsim.net`”，资源类型为 TXT 的迭代解析过程，从根域名，大致沿着“`www.ndnsim.net`”的层次迭代。

缓存解析器是 DNS 中一个重要的设计。它把 DNS 数据缓存下来用于应答未来的请求，这极大地保证了 DNS 服务的扩展性，同时也掩盖了暂时性的网络错误，例如某个名字服务器失效，但是它的数据可以存在于缓存解析器中^[97]。同时，由于一个名字域通常有多个名字服务器，缓存解析器需要有主机/路径选择逻辑来选择最佳的一个名字服务器从而最小化查询延时^[98]。

最初的 DNS 设计之后，它在两个维度得到了极大地扩展：资源记录的类型和安全机制。由于 DNS 成为网络中无可争议的最大的、最稳定的分布式数据库，它提供不同类型的数据查询，如电子邮件服务器 (MX RR), PKI (DANE) 等等。到目前为止，IANA 已经批准了 80 多种资源记录类型^[96]。

安全机制的补充是源于现实中越来越多的安全攻击和威胁，这个补充主要是通过 DNSSEC^[33-35] 来实现的。DNSSEC 通过公钥加密的办法来保证数据的有效性：所有的资源记录集合都包含一个签名（这个签名也是一种资源记录，类型为 RRSIG），用户生成这个签名对应的公钥证书（类型为 DNSKEY）也存在相应的内容域里。DNSSEC 采用了层次化的信任模型，父域里存储这子域 DNSKEY 的“指纹信息”(类型为 DS)。跟域需要一个大家都同意的公钥作为全局的信任锚点，并且这个信任锚点需要预装在所有请求者主机上。DNSSEC 的是一种信息为中心的设计，它直接对数据，而不是通信管道，进行签名；这与 NDN 有很多相似之处（表3.2）：

表 3.2 DNSSEC 与 NDN 的安全机制

	DNSSEC	NDN
安全粒度	资源记录集合 RR set	Data 数据包
签名载体	RRSIG RR	数据包中的 Signature-Info 和 SignatureValue
公钥标识符	Key Tag	Key Locator
公钥存储	DNSKEY RR	Certificate 包 (Data 的一种)
安全代理层次	父域中存储 DS 记录	父域中存储验证子域公钥的证书

一个名字域的 DNS 名字服务器分为有主从之分，一个主服务器，一个或多个

从服务器上。数据更新一般是发生在主服务器上^[99]，然后通过同步机制与从服务器上数据进行同步^[100,101]。

RFC 2136^[99] 规定了动态 DNS 更新 (DDNS or DynDNS)，它通过事务签名 (Transaction SIGnature, TSIG) 的机制来保证数据更新的安全。终端用一个密钥用于签名更新报文，该密钥对应的公钥证书存在名字服务器上（类型为 TSIG），使名字服务器能够验证更新报文。由于 DNS 分布式的特性，往往需要几个小时甚至几天，记录的更新才能在所有名字服务器上同步。

除了上述 RFC 定义的标准 DNS 更新机制，还有很多没有规范过的、基于 web 的更新机制。通常这种机制通过 web 做应用的用户验证，然后通过 web 接收到用户的更新请求之后调用本地的更新机制。

DNS 同步机制有两种策略，一种是全部同步 AXFR^[100]，一种是增量同步 IXFR^[101]。

3.1.1.1 nDNS 设计中的挑战

DNS 设计本身体现了信息为中心的思想：1) 对数据进行命名，数据可以直接通过名字标识；2) 对数据进行签名，源于数据的安全。在 DNS 中，信息以 RR 的形式与域名关联，允许多个 RR 与一个名字关联在一起，多个 RR 构成 RRset，RRset 与相关黏合记录 (glue record) 一起来作为应答报文；而在 nDNS 中，信息以 Data 形式与同一个名字关联（这些 Data 分别有不同的版本号），但返回给请求者的只有一个 Data，也并不存在黏合记录。所有的信息都需要包括在一个 Data 报文中。

在 DNS 中，内容的签名是作为一种类型的资源与信息黏合在一起，并且大部分时候是由名字服务器来签名；但在 nDNS 中，完成的资源本身包括内容与签名，而不是把签名作为一个补丁，通常情况下，资源是由名字所有者进行签名，而不是名字服务器。从这个角度看，nDNS 提供给终端用户的是更“完整的”应用生成的数据，因为完整的信息及信息验证所需要的签名都是由名字拥有者提供的。

DNS 中数据通信是基于 IP 之上的，由 IP 来保证将请求和应答报文传输给选定的接收者，所以 1) 需要服务器选择逻辑，即从多个服务器中选择一个作为请求报文目的对端；2) 更重要的是，针对一个请求，名字服务器可以回复它们所知道的最接近目标信息的答案，而不必须是该请求所对应的答案：例如，请求 “www.ndnsim.net/TXT” 到 “.net”的某个服务器 A，该服务器可能会回复 “www.ndnsim.net/TXT” 对应的数据，也可能是回复 “.ndnsim.net/NS”；在这个过程中，由 IP 来保证请求者和 A 之间的信息互通。而在 nDNS 中，1) 请求直接通过名字路由，请求端不需要服务器选择逻辑；2) 基于名字匹配的通信要求服务器必须回复请求对应的答案，而不能

是其它：请求“/net/ndnsim/www/TXT”会被路由到名字域“/net/ndnsim”，而不能是其它域；返回的数据也只能是“/net/ndnsim/www/TXT”。

3.1.2 NDN中现有名字解析系统设计

Alexander Afanasyev于2013年提出NDNS^[94]，基本思想是把DNS迁移到NDN用以解决NDN网络运行中的问题。与NDNS一样，nDNS设计也继承了DNS中的一些基本概念和设计原则，如层次化的命名空间、域名、名字域、资源记录、名字服务器、缓存解析器、终端解析器、迭代查询、递归查询等。但NDNS还有一些重要的问题有待解决，包括：1)设计还不能自举的解决自身系统中的请求路由问题，不符合网络基础设施定位；2)数据验证机制都需要额外的请求来完成，导致整个系统效率较低。本章nDNS改进了名字解析机制，自举地解决了自身请求路由问题，提出了高效和灵活的数据验证机制；并且在系统原型、实验评价部分更加完善。表3.3总结了两个工作的对比。

表3.3 NDNS与nDNS比较

	NDNS	nDNS
设计思路	继承了DNS中很多重要的概念和设计原则	
设计目标	解决NDN网络运行中的已发现的和潜在的挑战	
命名空间	反映了DNS命名空间	
系统定位	可扩展的分布式网络数据库，强调网络存储数据	可扩展的名字解析系统，强调名字解析
迭代解析	假设名字服务器宣告的前缀全局可路由；对于不能路由的“异常”情况，发送额外的请求获取FH(forwarding hint)记录	只假设根服务器前缀全局可路由，引用记录内嵌下一级名字服务器的LINK对象，自举解决解析请求的路由问题
安全模型	层次化的安全模型	允许多信任锚点的安全模型
递归查询	缓存解析器把目标RR（包括其签名）返回给终端，由终端重新获取证书来验证RR	两种安全模型，1) 终端与缓存解析器建立信任关系，通过验证缓存解析器的签名来验证数据；2) 缓存解析器需要把所有相关证书绑定在目标RR中一起返回给终端，由终端来验证RR
系统设计	路由和解多路复用关键字为“DNS”，系统保留资源类型FH、NS、NDNCERT	路由和解多路复用为“NDNS”，系统保留资源类型NS、ID-CERT
系统原型	Python实现，无部署	基于ndn-cxx的实现，部署于NDN Testbed

实验结果	网络缓存效率	1) 在 NDNS ^[94] 缓存实验结果基础上深入分析；2) 在主机/链路选择指标上与传统 DNS 性能对比
------	--------	--

把 DNS 迁移到 NDN 中最大的挑战是适配 NDN 的给予名字匹配的数据通信，基于数据的安全安全模型，本章工作在这两个核心设计上均与 NDNS 有重要差别和明显优势；具体设计、系统实现和部署、实验验证方面也更加深入。

3.2 nDNS 设计

nDNS 继承了 DNS 中的基本概念，如层次化的命名空间、域名、名字域、资源记录 (RR)、名字服务器、缓存解析器、终端解析器、迭代查询、递归查询和相似的安全模型。但是，它们在底层架构的区别，也就是 NDN 与 IP 的区别使得 nDNS 必须创造一些新的设计，尤其是迭代查询机制。

DNS 通过如下的方式来解析一个名字：

1. 缓存解析器通过配置好的根服务器的 IP 地址来启动自顶而下的查找。
2. 所有的名字服务器知道它们子名字域（的名字服务器）的 IP 地址，通过这种方式建立了一个层次化结构允许从根服务器开始的名字解析。
3. 这种层次化的结构通过特许的资源记录, NS RR 和 glue RR 黏合在一起。glue RR 避免了循环依赖的问题，即不需要额外的请求把名字服务器的名字解析成 IP 地址。

nDNS 中的名字解析大致按照如上的方式：

1. 一个基本的假设是，所有的路由器知道如何转发目的是根服务器的请求。
2. 定义一个层次化的结构允许自顶而下的查找。
3. 这种结构通过名字服务器中存储的特殊的资源记录黏合在一起。

就具体设计而言，nDNS 的第一条就与 DNS 根本上不同，因为 NDN 直接依赖名字来路由，整个网络的路由可达性可以直接通过在路由系统中注入名字前缀来实现，例如由根服务器宣告前缀 “/NDNS” ，这可以使得 NDN 网络能够有效地把请求转到最近的根服务器。第二条和第三条相对来说比较接近 DNS 的设计，但同时也有非常大的区别。DNS 名字解析过程中，迭代查询的请求报文都携带同样的问题（也是最终的目标问题），但是不同的目的 IP 地址把这些报文传输到不同的名字服务器。而 nDNS 的请求采用了 NDN Interest 的形式，只能包含对某一个数

据包的请求。因此，nDNS 请求必须显式地请求一个信息，可以是一个引用或最后的目的数据。

为了通过一系列的引用，最终获得目的数据，nDNS 把目标问题分成几个小的问题，迭代式地请求不同的引用，并最后定位目标数据所在名字域(第 3.2.3 章)。如果一个域名 N 不在路由系统中，它的父名字域必须存储一个 LINK 对象把 N 与一组可以全局路由的名字(路由锚点)关联在一起。这个 LINK 对象必须包含在关于 N 的引用中。因为 NDN 网络中支持缓存，一个与 N 关联的 LINK 对象可以在 N 或者 N 的子孙域名的解析过程中发挥作用。

nDNS 也充分利用了 NDN 固有的优势：1) 请求报文的转发是通过网络层对主机/路径的信息来决定的，从而避免了缓存解析器上 end-to-end 的主机/路径选择逻辑；2) 聚合相同的请求，天然支持组播；3) 网络层缓存的支持；4) 内置的数据签名和验证机制。表3.4详细列举了 DNS 和 nDNS 的比较结果：

表 3.4 DNS 与 nDNS 比较

	DNS	nDNS
命名空间	DNS 命名空间	反映了 DNS 的命名空间
扩展性和鲁棒性	数据冗余和应用层缓存	数据冗余，网络层缓存，应用层缓存
主机/路径选择	缓存解析器 end-to-end 选择逻辑	网络层 hop-by-hop 选择
安全机制	DNSSEC 扩展(应用层)	NDN 固有支持(网络层)
数据格式	RRset	NDN Data(第 5.2.4 章)
迭代查找	自上而下，隐式	自上而下，显式

3.2.1 nDNS 命名空间和数据命名

NDN 直接用应用层的数据名来获得数据，今天的应用都是用 DNS 命名空间，因此 nDNS 反映了 DNS 层次化的命名空间。为了更好地区分应用层的名字和存在 nDNS 的数据的名字，nDNS 中名字域的名字是由关联着的域名后添加一个“nDNS”名字部件而成(图3.2)，这同样有利于解决 nDNS 的路由问题。每个 nDNS 名字域都代表一个管理单元，并且可以划分为更小的单元，并把命名空间继续向下划分。一个 nDNS 名字域可以包含多个名字服务器，每个名字服务器都含有该名字域的数据冗余。

nDNS 的层次化结构允许解析端不需要任何的启动配置来解析名字。nDNS 的

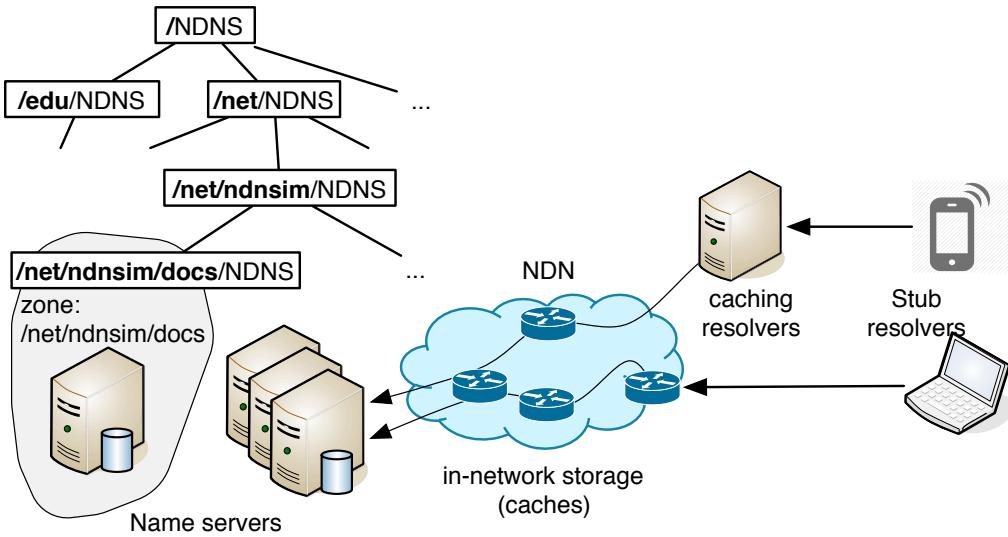


图 3.2 nDNS 设计元素：nDNS 继承了 DNS 的层次化命名空间，名字域，树形资源组织结构，名字服务器，缓存解析器，终端解析器等基本概念；但是不同节点是通过 NDN 网络连接起来，所以 nDNS 的通信需要设配 NDN 通信的要求，包括基于名字匹配的 Interest / Data，网络层缓存和主机/链路选择，基于数据的安全等。

根服务器向路由系统中宣告根名字域的名字（“/NDNS”），这样 NDN 网络就可以有效地转发目的为根服务器的请求^①。而其它名字域的可达性可以通过路由系统（向路由系统中宣告前缀），或者通过命名空间的引用，让父名字域存储和提供子名字域的 LINK 对象。

nDNS RR 与一个域名和类型 (type) 关联在一起，被存放在该域名对应的名字域或其父名字域中，这意味着如果名字域 Z 中存放的一个与域名 N 关联的 RR，则其中 Z 是 N 的前缀或者是其本身，本章用“label”来表示 N 中去掉 Z 的那部分。例如，一个指向名字域 “/net/ndnsim” 的引用被存储在名字域 “/net/NDNS” 中，则 label 是 “/ndnsim”。

为了路由和解多路复用 (demultiplexing) 的目的，本章用名字域的名字，label 和 RR 的类型 (type) 来构造 nDNS 请求 Interest 中的名字 (图3.3)。RR 的名字应该与请求的名字符合最长前缀匹配的规则。为了保证数据的不变形 (immutability) 和存储比较大的 RR 的能力，nDNS 应答数据包的名字中还包含版本号 (version number) 和可选的分段号 (segment number)。

^① nDNS 也可以完全不需要根名字域，如果所有的 TLD 的名字都被宣告在路由系统中。

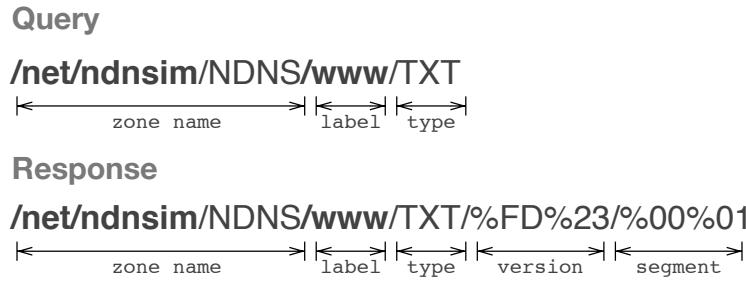


图 3.3 nDNS 迭代查询 Interest 和 Data 名字范例：zone name 部分有利于路由和解多路复用。

3.2.2 数据格式

NDN 以每个 Data 为单位获取数据，每个 Data 包含足够的信息来说明它与哪个名字关联在一起，属于哪个名字域，及其资源类型（图3.4）。每个数据包都在创

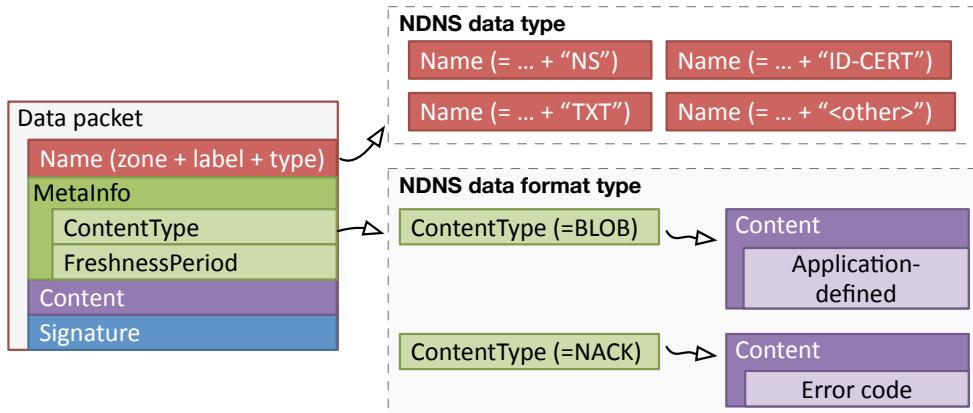


图 3.4 nDNS Data TLV 格式：定义了正常的数据包，以及应用层 NACK

建的时候产生签名，这种基于数据的安全取代了 DNSSEC 里的安全机制，它消除了在应用层创建签名以及相关的复杂性（如需要把“RRSIG”的记录与数据黏合在一起）。NDN 数据包中还包含有 FreshnessPeriod 域来影响路由器缓存该包的时间。表3.5总结了 nDNS 与 DNS 两者数据格式的对比。

出于自身设计的需要，nDNS 保留了两种资源类型，“NS”和“ID-CERT”；“NS”如同 DNS 中的设计一样，使得域名和名字域的托管成为可能。如果对应的名字域不能直接路由，则 NS RR 中需要包含对应的 LINK 对象。ID-CERT RR^[102] 存储了公钥证书资源，从而允许数据请求者可以通过 nDNS 中的公钥来验证存储在 nDNS 数据。nDNS RR 数据包的签名与中包含一个公钥证书的名字，这个公钥证书需要

表3.5 nDNS与DNS数据格式对比

	DNS	NDNS
数据单元	DNS response，包括 identifier, flags, codes, question count, answer record count, authority record count, additional record count 等内容	NDN Data，以名字作为标识符，格式完全由应用决定，无与 question count, answer record count 等对应的信息
应用数据	Answer section 包含一个或者多个 RR	应用或者资源类型决定的 Content 域格式
资源名字	RR 关联的名字	Data 名字的一部分
资源类型	RR 的类型	Data 名字的一部分
资源 class	RR class	未定义
缓存时间	TTL	Data FreshnessPeriod 域 & 名字服务器和缓存解析器端维护的 TTL

存储在 nDNS 中。通过“NS”和“ID-CERT”两种记录，nDNS 自举地解决了自身请求路由、自身数据验证的问题，而不需要依赖第三方系统。

为了系统展示的需要，本章还定义了“TXT”资源类型，它的格式为自由定义格式的文本信息。为了说明某个资源记录不存在，nDNS 定义了应用层的 NACK (图3.4)。NACK 里包含一个错误代码来说明该记录不存在的原因。

除了 nDNS 保留的资源类型，nDNS 可以存储其它任意应用层定义的资源，只要应用本身选定一个类型的名字 (如“MY-INFO”)，并且就该类型 RR 的格式定义达成一致。当一个比较大的内容不能放到一个数据包中时，它被分成多个分段存入 nDNS。

3.2.3 迭代解析

nDNS 请求与 DNS 请求有不同的语义，即 nDNS 请求的语义是返回被请求的这个确定的 RR，而 DNS 请求的语义则是返回请求的 RR 或者是离这个 RR 最近的引用。不同的语义导致了不同的迭代解析方式。nDNS 解析端必须显式地请求对应名字不同层次上的引用，直到最后的目标数据被解析出；而不能像 DNS 那样发送目标请求，让名字服务器返回目标数据或者最接近的引用。nDNS 请求显式的包含请求关联的名字和资源类型，这样热点的数据，如 TLD 的引用，将在网络中广泛缓存，从而极大降低名字服务器的流量压力。

本章用与名字“/net/ndnsim/www”关联的 TXT RR 的解析过程来展示迭代解析

的过程（图3.5）。假设缓存解析器 R_C 中缓存为空，迭代解析的第一步是发送一个名为“/NDNS/net/NS”的 Interest 来获取指向名字域“/net/NDNS”的引用（NS RR）。因为“/NDNS”是全局可路由的，路由器可以把这个请求转发到 nDNS 根服务器。

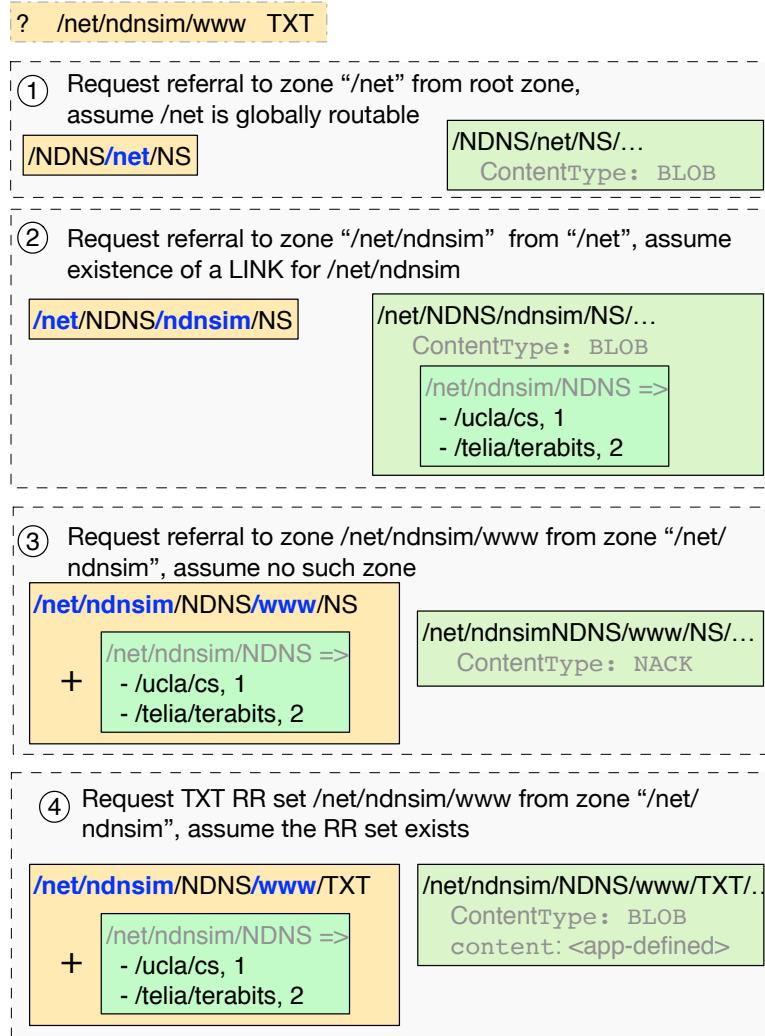


图 3.5 nDNS 名字解析范例：通过显式的自顶而下的名字解析获得数据；NS RR 中内嵌下一级名字服务器对应的 LINK 对象，从而自举地解决自身请求的路由问题。

当 R_C 收到对应的引用时候，根据名字域“/net/NDNS”是否在路由表中，这个引用可能包含不同的信息：a) 不包含特殊的信息，如果“/net/NDNS”本身可以路由；b) 包含 LINK 对象 L，把“/net/NDNS”绑定到一个可以路由的名字上。两种情况下， R_C 都可以继续迭代的过程继续发送请求“/net/NDNS/ndnsim”获取下一个引用，在 b) 情况下，需要把 L 绑定到对应的 Interest 中去；这个的过程一直持续，直到 NACK 被返回；上面的例子中，第三步返回 NACK “/net/ndnsim/NDNS/www/NS/”，这说明没有更进一步的命名空间的托管，目标数据只可能存储在名字域“/net/ndnsim/NDNS”

中；所以可以发送最后的目标问题“/net/ndnsim/NDNS/www/TXT”获取数据。

本章同样需要考虑当一个域名没有自己的名字服务器只能把数据存储在它的父名字域的情况。例如“/net/ndnsim”没有自己的名字服务器，把数据放到名字域“/net/NDNS”中，但“/net/ndnsim/www”可能是有名字服务器。在这种情况下，“/net/NDNS/ndnsim/NS”将返回一个特殊的 NACK（通过 NACK 里的 code 来标识类别），表明请求的域名没有自己的服务器，但是迭代过程需要继续查找下一个引用。缓存解析器在 label 中添加一个名字部件，构造了下一级的名字，形成一个新的请求“/net/NDNS/ndnsim/www/NS”，这个请求依然请求名字域“/net/NDNS”中的数据。

3.2.4 数据验证

nDNS 默认支持层次化的数据验证，所有的资源记录签名对应的证书都存在 nDNS 中；与此同时，证书本身作为资源记录，验证该证书所需要的证书（生成该证书中签名所对应的证书）也需要存在 nDNS 中，这种层次化安全模型的语义从资源记录的名字上显式表达：假设签名名字为 C1 的证书所对应的证书为 C2，则的 C2 作为资源记录所关联的域名应该是 C1 关联域名本身或是其父节点。

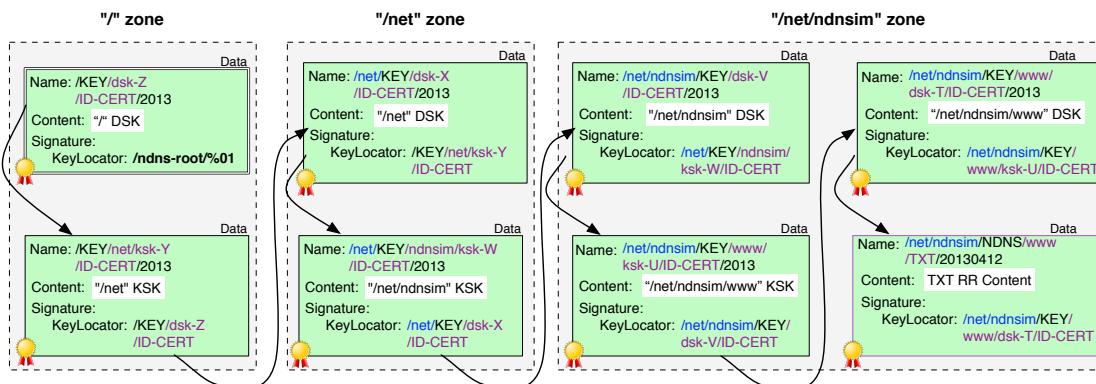


图 3.6 nDNS 层次化安全模型范例：建立从根证书到具体 RR 的信任链，其中根证书作为信任锚点，需要预装在终端用户主机上。

图3.6 展示了一个层次化模型验证名为“/net/ndnsim/NDNS/www/TXT/20130412”的资源记录的信任链。图中显示了三个名字域对应的资源记录，所有的资源记录关联的域名有严格的层次关系。其中根证书（范例中的“/KEY/dsk-Z/ID-CERT/2013”）作为默认的信任锚点，需要通过带外方式原装到终端用户主机上。

图中展示的信任链是从“/KEY/dsk-Z/ID-CERT/2013”开始，需要说明的是，对终端用户而言他可以配置自己能接受的信任锚点；例如对于域名“/net/ndnsim”管理者来说，它可以将自己名字域和其子名字域中数据验证的信任锚点设为证书

“/net/ndnsim/KEY/dsk-V/ID-CERT/2013”，从而缩短信任链长度，简化数据验证。

3.3 数据管理与相关问题讨论

3.3.1 缓存解析器的角色

DNS 缓存解析器的主要作用是通过缓存来解决扩展性问题；NDN 在网络层支持缓存功能，极大地削弱了缓存解析器的重要性。但是网络层缓存是所有的数据流量共享的，而不能保证有足够的缓存空间留给 nDNS 流量，因此，保留缓存解析器这种专门为 nDNS 设置的缓存依然有益。另外，缓存解析器代替终端执行迭代解析操作，这对一些电量有限的设备很有帮助，如越来越多的传感器，越来越普及的可穿戴设备等。缓存解析器同样可以利用 NSEC^[103] 或 NSEC3^[104] 机制来消除对不存在的名字的查询（可能是错误输入，误配置或者攻击所致）。

缓存解析器的安全性在 DNS 中是一个重要的考量，终端解析器要么必须需要信任缓存解析器提供的内容，要么必须亲自执行迭代查询过程来保证安全性。NDN 的基于数据的安全模型让终端用户能够验证 nDNS 中的数据而不用相信缓存解析器。如图3.7 所示，nDNS 允许把证书组^[191] 附在目标数据后，这样终端可以直接通过这些证书来验证对应的 RR，而不用再发额外的请求。与此同时，nDNS 同样允许终端与缓存解析器之间建立信任关系（适合专用的缓存解析器，如家庭、学校、公司内部），这种情况下，递归查询被缓存解析器签名。

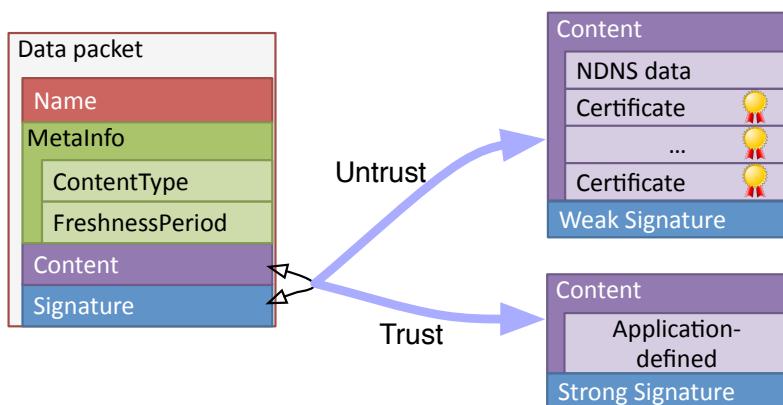


图 3.7 递归查询包的 TLV 格式，它对应两种不同的安全模型

因为 NDN 网络通过名字转发 Interest，nDNS 递归查询请求必须携带缓存解析器宣告到路由系统中的名字。以 Google 的缓存解析器为例，它应该宣告前缀 “/com/google/NDNS-R” 到路由系统中。该前缀、目标信息关联的名字、资源类型和验证模型等信息一起构成了一个递归查询，本章用 “proof” 来表示缓存解析器需要

提供证书组的信任模型，用“non-proof”表示缓存解析器直接签名对应数据的情况（图3.8）。

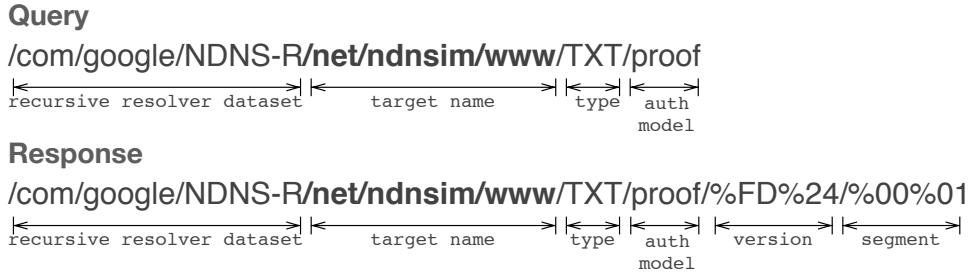


图 3.8 递归查询请求和应答命名范例

DNS 缓存解析器的一个重要功能是主机/链路选择选择，在 NDN 中网络层自动选择最好的主机和路径；通过带状态的转发平面，这种方式能够取得更佳的性能优势，本章在第 3.4.3 章验证了这一点。

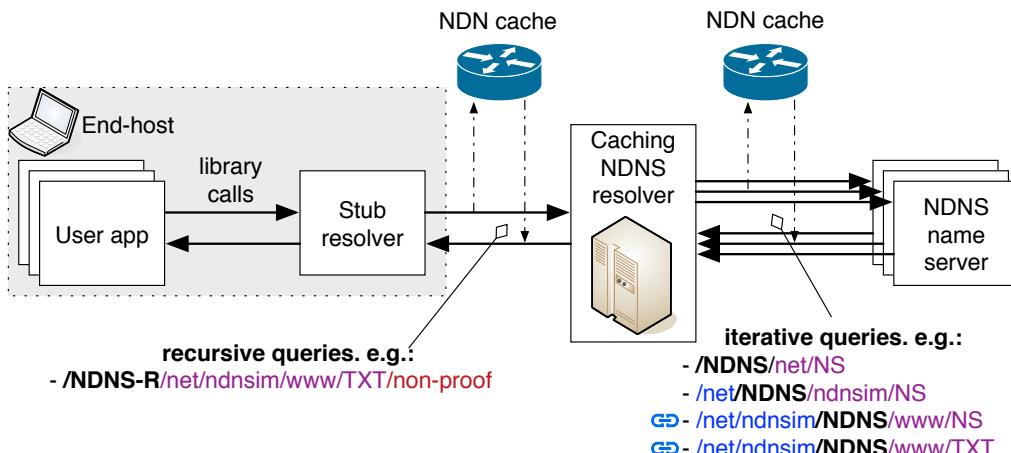


图 3.9 nDNS 中名字解析概览：包含递归查询、迭代查询的整体名字解析过程

图3.9展示了包含缓存解析器和递归查询、迭代查询的整个nDNS查询过程。

3.3.2 名字域更新和同步机制

随着时间的推移，nDNS 中的数据可能需要更新。由于移动设备越来越多时，nDNS 要求所有名字服务器都能处理数据更新。在 nDNS 中，RR 是由真正的数据产生者签名的，而不是由名字服务器，但是签名对应的公钥证书必须存储在 nDNS 中以便数据的验证。一种简单的更新方式是利用 Interest 把 RR 传输到目标服务器。在客户端，域名拥有者先构造 RR，并把 RR 封装在一个或多个 Interest 中，然后把

这些 Interest 发送到名字服务器去；在服务器端，名字服务器验证 RR，并更新名字域。

因为 NDN 网络不能保证每个更新报文都传递到对应名字域的所有名字服务器，所以需要一个名字域同步的方案来保证所有名字服务器中数据是一致的。这是一个典型的 NDN 网络中分布式数据同步的问题，针对这个问题，目前已经多个解决方案，如 ChronoSync^[25], iSync^[105]。在^[106] 中展示了一个基于 ChronoSync 的同步方案，这个纯分布式的方案消除了单点故障和流量集中这两个在集中式方案中常遇到的问题。并且即使网络被分割成不同的子网络，每个子网络里面的节点也可以更新和同步各自的数据；一旦网络回复，所有节点都可以基于最新的状态同步数据。

3.3.3 nDNS 的特殊地位

nDNS 作为 NDN 网络中一个必须时刻在线的基础设施服务，它对网络中数据通信的启动起到了非常重要的作用。只需要假设 “/nDNS” 可路由，nDNS 允许终端获取任何存储在其中的数据；对其它程序来说，nDNS 可以解决路由问题，移动问题和公钥仓库的问题。但是，所有这些“伎俩”只需要 nDNS 具备即可，其它的应用只需要依赖 nDNS，而不再重新具备这样的特点。随着时间的推移，nDNS 可以用于更多的场景，就像 DNS 在 IP 中一样，一开始的作用是域名到主机 IP 的映射，而现在已经定义了 80+ 种不同的资源类型应用于不同的场景。

正因为这种独特而又重要的角色，nDNS 要求具有高度的鲁棒性，并且能抵御网络故障。即使属于某个名字域的一个或多个名字服务器宕机了，依赖其它服务器也能进行名字解析；即使属于某个名字域的所有名字服务器都宕机了终端还可以依赖缓存来获取数据；即使这个名字域的数据不能够从缓存中获得，受到影响的只是这个名字域及其子名字域，而其它部分能够正常工作。

nDNS 是保障网络整体可达性的重要部分，但是对于没有基础设施的，ad-hoc 通信，如车载网络，传感器网络等，可能并不需要 nDNS。在那些场景下往往可以直接通过无线广播信道来获取数据，而不需要名字解析。

3.4 系统原型、部署与实验

3.4.1 系统原型与部署

本章实现了 nDNS 系统原型^[107]，该系统原则包含若干个程序。

- 名字服务器服务程序：能够存储数据数据、应答 nDNS 查询请求的服务。
- 迭代查询客户端程序 dig：能够通过迭代查询的方式，进行名字解析。

- 远程数据更新程序 update：通过发送 Interest，从远端更新资源记录。
- 多个本地 nDNS 管理程序：包括对名字域和本地数据记录的增删改查。

该系统原型被部署在 NDN Testbed^[8] 上，nDNS 系统使用了 NDN Testbed 的根证书作为系统的默认的信任锚点。nDNS 系统还为 NDN Testbed 的参与机构创建了为数不少的名字域，并且把 nDNS 应用于实际的 NDN 程序中去，例如群聊天的程序 ChronoChat^[108]，网络视频播放程序 NDNTube^[36] 等。本章还开发了一个示范程序来展示 nDNS 的通用性和数据可更新性。这个程序叫做 CPUSensor^[109]，它能帮助电脑管理员监控电脑的 CPU 温度，并且把数据以“CPU-INFO”的资源类型存储在 nDNS 中。

由于 NDN Testbed 的规模和网络不可控的因素，本章使用最受欢迎的 NDN 网络模拟器 ndnSIM^[110] 来验证 nDNS 的设计。

鉴于与 DNS 相似的设计已经被 DNS 证明非常成功，本章主要衡量 nDNS 与 DNS 不同之处，也就是那些利用了 NDN 信息为中心优势的设计。本章实验主要包含两个部分：1) nDNS 利用 NDN 的网络层缓存，在不同缓存空间的情况下，nDNS 能够从网络层缓存中获得的便利效果，尤其是估测 nDNS 是否能利用网络层缓存解决系统扩展性问题；2) 主机/链路选择是 DNS 缓存解析器的重要功能，nDNS 通过网络层来解决这个问题，实验比较了两者的性能。

3.4.2 网络层缓存的优势

其中，网络层缓存工作在文章^[94] 中得到充分验证，考虑到缓存机制在 NDN 应用中的通用性，本章简述其结果，在此基础上加以深入分析。

文^[94]从一个商业 ISP 获取了 2010 年 5 月份某一天 10 分钟内向“.com”查询的 DNS 的数据，数据集一共大约包含 3.3×10^6 个请求。该实验使用 Rocketfuel AT&T 的 PoP 拓扑^[111]，它包含 625 节点和 2101 条链路；在模拟中，200 个节点作为缓存解析器根据采样数据发送请求，实验中随机地选择了一个根域名服务器和 13 个“/com”名字服务器。实验中缓存空间的大小范围从 10 个 Data 到 10^5 个 Data，同时分别采用了三种最常见的缓存替换策略，LRU (least-recently used), LFU (least-frequently used) 和随机替换策略。实验结果如图3.10 所示。

根据本人对实验数据的进一步分析，该 DNS 采样数据集这些请求大概包含了 4.2×10^5 个不同的目标数据，理论上说，这组数据最高的缓存命中率可以达到 86.8%。图3.10 展示了缓存的效果。当缓存大小为 10^3 个 Data 时（大约是请求数量的 0.03%），缓存命中率是 22%；缓存命中率上升到接近 50% 当缓存空间是请求数量的 0.3%，这个值增加到 60% 当缓存空间是请求数量的 3%。60% 已经很接近命

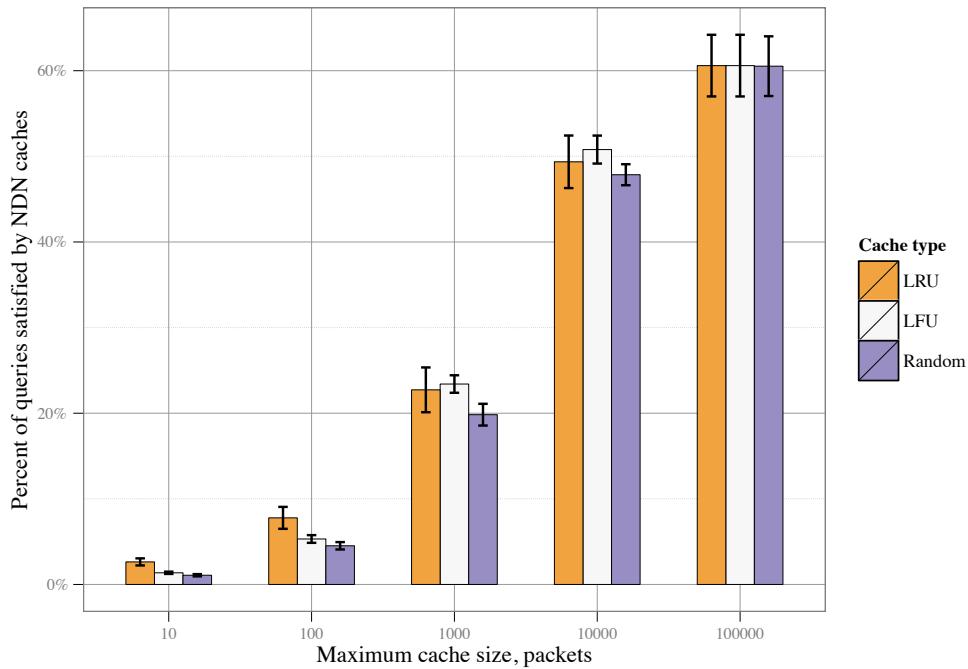


图 3.10 网络层缓存的作用 (本图摘自文^[94]) 缓存命中率高达 60%+ 当缓存空间是 10^5 (请求数量的 3%)，这非常接近理论上限。

中率的理论上限 (86.8%)，这说明 nDNS 可以充分利用网络层缓存带来的优势。

除此之外，本章还强调两点：1) 上述的流量数据集是 DNS 流量，然而在 nDNS 中，考虑到特别流行度名字前缀，如 “/com/youtube”，“/com/google”，往往直接存储在 DFZ FIB 中，而不需要通过 nDNS 迭代查询来解决路由扩展性，再考虑到网络层缓存对热点数据的缓存能力，nDNS 的流量极有可能与 DNS 不同（但现在还没有 nDNS 流量，所以不能实验来验证）；2) 上文实验结果是 nDNS 利用网络层缓存获得的性能，而不是比较 nDNS 与 DNS 的性能；但考虑到 nDNS 同时具有缓存解析器和网络层缓存，nDNS 的扩展性问题应该不比 DNS 严重。

3.4.3 主机/链路选择的优势

每个名字域都可能包含多个名字服务器，在 DNS 中，缓存解析器选择目标名字服务器，这是 end-to-end 风格的；而在 nDNS 中，由 NDN 网络来选择目标主机，这是 hop-by-hop 风格的。下面展示 hop-by-hop 风格的优势。本节使用的拓扑如图3.11(a) 所示，这样的拓扑比较简单，但保留了一般性，通过把多个链路虚拟成一个链路，很多复杂的拓扑可以规约成这样的拓扑。这个拓扑中包含三个属于 TLD 名字域的名字服务器，其中 NS-1 位于缓存解析器的同一个网络中，NS-2 位于缓存解析器的另据网络中，而 NS-3 则位于一个远端的网络中。为避免缓存的干

扰，实验中发送不同的请求，并且发送速率每10秒增加一次。

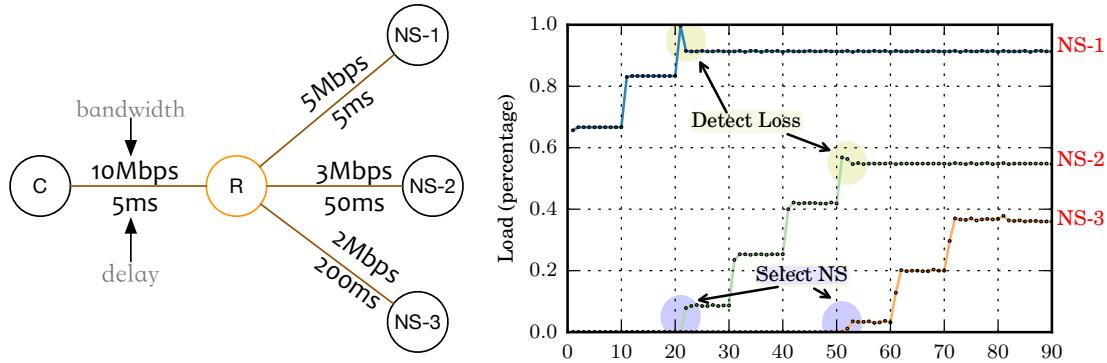


图3.11 nDNS功能验证：(a)拓扑保留了一般性，复杂拓扑可以规约到这个简单的拓扑；(b)显示，在第20、50秒，网络监测到链路丢包，选择了新的链路来转发Interest。

在nDNS中，consumer并不“选择”名字服务器，NDN路由根据默认的转发策略，也就是best route，和当下最优的路径来选择名字服务器。从图3.11(b)中可以发现，当带宽允许的情况下，NS-1永远是第一选择(0-20秒)，而NS-2则是第二选择(20-50秒)。当目前已经选择的路径不能处理当前的流量时候，中间路由器会检测到请求丢失，然后把请求转发到其它可能的名字服务器(第20, 50秒)。当一个请求丢失时，中间路由器迅速地把它重新转发(当有其它可行的带宽时)，而不留给终端应用等待超时之后再处理。

实际上，如图3.12(a)所示，请求的延时时间很靠近从终端到不同名字服务器的RTT(round-trip time)。这个模拟结果说明，NDN网络能够把请求转发到当前最好的名字服务器上去，并且避免了请求丢失后的较长的超时等待时间。

作为对比，本节也实现了一个简单的DNS缓存解析器，它估计到不同名字服务器的RTT，并且通过这个估测的RTT每次选择最优的名字服务器来发送请求。当过了预热阶段之后，这种选择逻辑偏向于选择NS-1，当流量比较小时，它做出的决定是正确的；但是当流量比较大时，其它两个主机/路径不能被充分利用，它并不能取得很好的效果；所以本节把超时事件也作为另外的名字服务器选择的因素，来平衡不同名字服务器的流量压力，结果如图3.12(b)所示。

总体来说，这种end-to-end的主机/路径选择逻辑缺乏网络层信息，而不能充分利用所有可能的资源(带宽，数据缓存等)，并且需要更长的时间来应对拥塞。这导致90%+的请求会选择NS-1(图3.12(c))，在nDNS里这个值为65%+，更为合适。

并且，成功获得请求的数量只有nDNS情形下的55%(图3.12(d))，更加说明

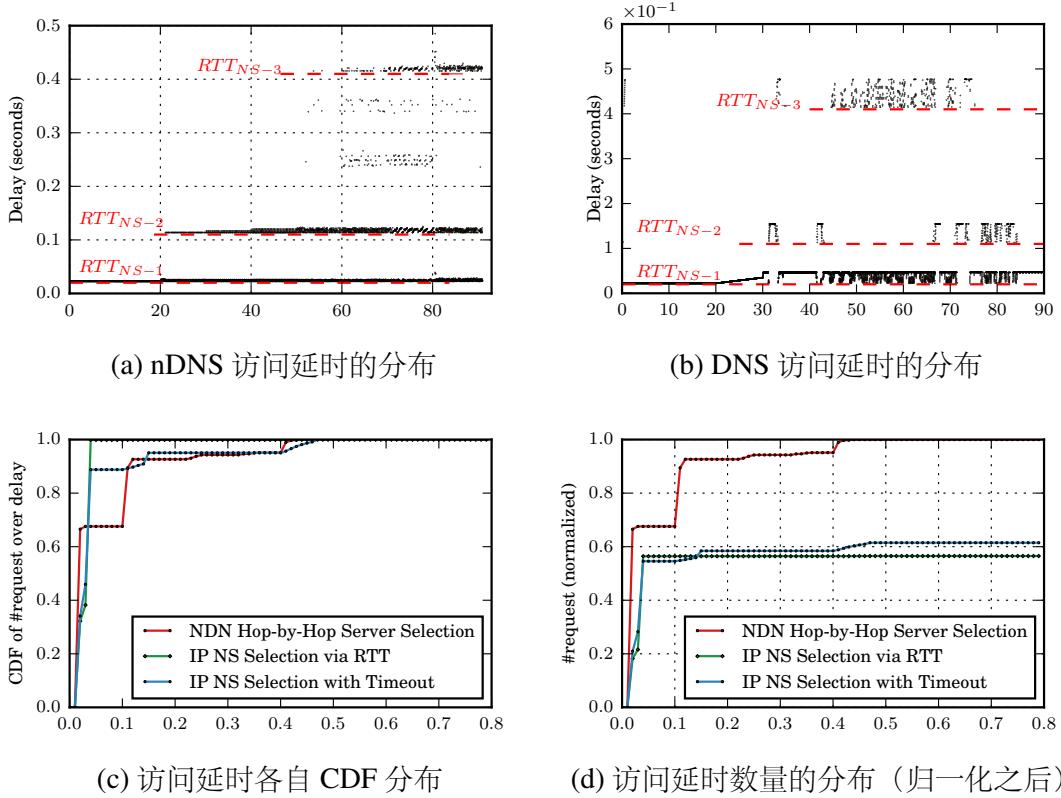


图 3.12 nDNS 与 DNS 访问延时对比：(a) 和(b) 对比说明，nDNS 通过网络选择能更好的选择服务器；(c) 显示 DNS 中 90% 的请求是由 NS-1 回复的，而在 nDNS 中只有 65%，说明 nDNS 更充分地利用了冗余链路；(d) 显示，就服务能力而言，DNS 服务能力只有 nDNS 的 60%，这是充分利用冗余链路的结果。

了 nDNS 机制的优势。

3.5 nDNS 总结

本章指出了 NDN 网络运行已经发现的一些严峻问题，并受到 DNS 启发，为 NDN 量身设计了一个名字解析系统 nDNS。本章指出名字解析系统对运行网络的重要性，它能切实解决网络运行中遇到的诸多问题，它作为网络固定和稳定的基础设施服务，可以存储路由锚点、信任锚点和其它信息，可以作为权威数据中心来解决网络中存在的动态性（动态到静态数据映射）、扩展性（大量数据到有限数据的映射）、安全性（待验证内容到可信任内容的映射）问题：

- 实际运行的网络永远存在动态性和扩展性，最常见的就是移动性和路由扩展性，解决之道就是把一个存在动态性的名字绑定到一个非动态的名字上，例如 SNAMP 方案把不能全局路由的名字映射到可以全局路由的前缀上，把过时的 LINK 对象更称到最新的 LINK 对象；故而网络中需要一个名字解析系统来实现这样的映射。

- 网络空间中的成员一般需要占有一定的公共资源，例如名字空间、公私钥等，这些资源的分配得到所有成员的认可的前提下，安全的通信才能够进行。所有网络中需要有一个基础设施来提供这些资源分配的记录以及相应验证所需要的的数据。
- 对于一个分布式网络中程序而言，一些应用或者服务器需要向外暴露或者宣告一些信息，如 IP 地址、地理位置 (DNS 中资源类型为 LOC)、负责人 (DNS 中资源类型为 RP)，使得它们的客户能够根据这些信息来启动对应的操作。考虑到信息的动态性和多样性，人工的配置并不可取，也需要一个网络存储映射系统来保存这些信息并提供访问。相比于 DNS 中 80+ 多种资源类型，NDN 资源类型种类还很有限，但各种不同的资源都可以存储于 nDNS 中，DNS 中的资源类型都可以迁移到 nDNS 中。

值得一提的是 MobilityFirst^[92,93] 中的映射系统 GNS 也有重要的意义，MobilityFirst 项目中映射系统是网络层最核心的部件，nDNS 则只是网络的基础设施；这意味着 NDN 允许应用通过自定义的方式实现通信的目标，这一点在自组织网络或者边缘网络中具有极其重要的意义，因为在这样的网络环境下：1) 可能不能保证全局网络存储映射系统的可用性，这种情况下应用还需要能够独立地运行；2) 广播信道往往用于这样的场景（车载网络、传感器网络等），网络的动态性、资源的获取，这些问题都可以通过广播来解决，而不需要比较复杂的映射系统。

与此同时，本章积累了如何把一个传统的基于 IP 的系统迁移到 NDN 中的一些经验，希望这些经验有助于未来更多的应用迁移：

- 命名和命名空间是首要考虑的问题，NDN 的很多优势来自于把数据安全、转发与名字本身关联在一起（例如消除 DNSSEC 或 TLS/SSL 这样的安全机制）。但同时这种关联也引入了一些挑战，例如在 DNS 里，请求的目的地与请求携带的问题（也就是应用层的语义）是完全独立的，所以递归查询和一系列的迭代查询都可以携带相同的问题；而问题的回复也可以是多样化的，可以是问题的答案，也可以是进一步的指向该含有答案的主机的引用；在 nDNS 里，名字路由和匹配要求迭代查询报文携带不同的问题。
- 与命名相关，直接使用应用层的名字来做路由能够带来很多性能上的优势，例如这样的设计能获得天然的组播和网络层缓存的支持，有效降低名字服务器的负载；而且逐条的路径选择，快速的失败恢复机制能够消除 DNS 需要的复杂的主机选择逻辑。
- NDN 中每一片数据都要进行签名为数据安全提供了坚实的基础；这种源于数据的安全模型在时间和空间上都分离了数据的产生与使用，对移动场景、

组织与网络、异构网络间通信带来极大的便利，同时也很好地匹配了大规模数据分发的机制（如缓存支持，多数据源等）。

同时本人也意识到，关于网络运行和 NDN 应用开发的工作尚有未尽之处，例如除了本节指出的网络运行挑战之外，更多的问题需要在更深入的运行、实验和研究中去发现；而关于 NDN 中应用的安全和信任机制的设计光有 nDNS 作为证书仓库是远远不够的，有待进一步研究。

第4章 ACM：NDN 通信效率建模

如第 1.3.2 章所说，NDN 通信中 consumer 没有确定的通信对端导致现有网络传输效率模型不能描述 NDN 通信，而每个数据包中较大的开销又使得合适的数据包尺寸对性能有明显影响，因此本章的目标是建立 NDN 的网络通信效率模型，并用于优化 NDN 中数据包尺寸。

首先，本章需要一个“指标”来描述通信效率，考虑到每个数据包的开销比较大，一个“可能”的指标是载荷比 (payload ratio)：

$$\text{PayloadRatio} = \frac{\text{payload}}{\text{payload} + \text{overhead}}$$

载荷比可以衡量一个数据包中有效部分的比值，然而这种衡量方法是一种“静态”的描述，没有考虑网络传输中丢包、重传的情况。对网络传输而言，只有在整个数据包成功传输的情况下，数据包中的载荷才是有效数据，而被丢弃的数据包应当作为开销的一部分；推广到更一般化的场景，即不是单一的数据包，而是有在大量数据包传输（含重传）的情况下，所有吞吐量 (throughput) 中的有效吞吐量 (goodput) 的比例能反映传输效率，本章定义这个比值为 G2T (goodput to throughput) 来衡量网络通信效率：

$$\begin{aligned} G2T &= \frac{\text{Goodput}}{\text{Throughput}} \\ &= \frac{\text{payload}}{(\text{payload} + \text{overhead}) / (1 - \text{ChunkLossRate})} \\ &= \text{PayloadRatio} \times (1 - \text{ChunkLossRate}) \end{aligned} \tag{4-1}$$

举一个例子来说明载荷比与 G2T 的区别，以及网络传输中数据包的尺寸对传输效率的影响。如图4.1 所示，假设有 21K 字节应用层数据需要传输，每个 Data 的开销是 0.5K 字节；再假设 NDN 下一层链路的数据帧最多能封装 1.5K 数据 (即 MTU = 1.5K)，且链路数据帧丢失率固定为 2%。使用如图4.1所示 A, B 两种方法，用不同尺寸的数据包来传输数据。表4.1 显示了两种方案 (A 和 B) 传输性能，并且给出了最优的方案 (7.5K x3) 与两者进行比较。

上述两个方案中，方案 A 特点是通过小的包尺寸 ($1.5\text{K} = \text{MTU}$) 获得最小的丢包率 (2%)，方案 B 的特点是通过大的包尺寸 (21.5K) 获得最大的载荷比 (97.7%)，根据公式 (4-1)，显然减小丢包率、增大载荷比都有助于提高网络传输效率 G2T；

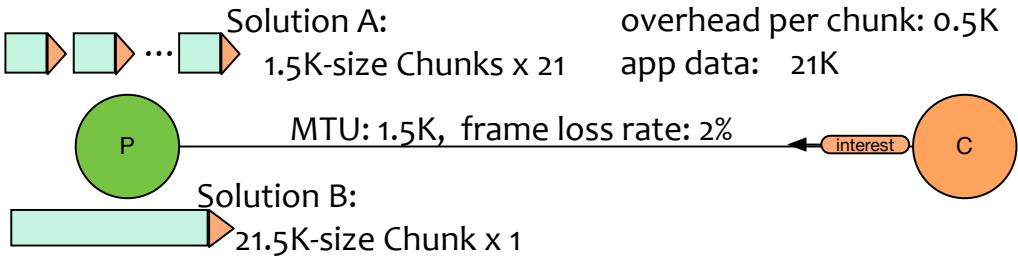


图 4.1 两种不同的方案来传输 21K 数据。方案一把数据分别封装在 21 个数据包里，传输时候不需要分片，一共需要 21 个数据帧；方案二把数据封装在一个数据包里，形成一个 21.5K 的到的数据包，在传输时候需要形成 15 个分片，共需要 15 个数据帧，任何一帧的丢失都导致整个数据包重传；然而太大或太小的数据包都不能最大化传输效率，最好的包尺寸是 3 个 7.5K 的数据包。

表 4.1 三种方案的性能比较

	A: 1.5K x 21	B: 21.5K x 1	Best: 7.5Kx3
载荷比	66.7%	97.7%	93.3%
丢包率	2%	26.1%	9.6%
G2T	65.3%	72.1%	84.4%
网络层流量（数学期望）	32.1K	29.1K	24.9K

然而方案 A、B 都在优化一个因子的时候，忽略了另外一个因子，并不能最大化 G2T，A 和 B 的 G2T 分别是 65.3%、72.1%。实际上，最好的包尺寸是 7.5K。在这种情况下，G2T 达到 84.4%，意味着传输这 21K 应用层数据，在网络层的流量（的数学期望）是 24.9K，比 A 和 B 的网络层流量要少很多（32.1K、29.1K）。

上面的例子可以说明两点：1) 在网络传输中，G2T 更能反映传输一定数据所需要的开销大小，它比载荷比更好地反映网络传输的效率。2) 更进一步，在数据包开销不可忽略的情况下，特别大或特别小的包尺寸都不能获得好的传输效率。如何调整数据包大小使得 G2T 最大化？这个问题是本章研究主要问题和贡献所在。根据公式 (4-1)，载荷比与载荷正比，丢包率与载荷正比，故存在最佳的载荷大小能最大化 G2T。

在本章中首先以最大化 G2T 为目标，考虑到 NDN 中数据获取没有通信对端的情况，从丢包率与包大小的角度出发建立模型，该模型能够推算出不同包大小的传输效率；然后本章用连续时间的马尔科夫链来分析链路丢帧率的影响。利用本章的模型，能推导出最优包尺寸从而最大化 G2T；更进一步，本章提出自适应的包尺寸模式 (Adaptive Chunk Mode, ACM) 应用于大规模数据分发，把模型结果

应用于实际应用中去，还开发了一个利用 ACM 的应用，一个基于 NDN 的在线视频播放器 nPlayer，用于验证 ACM 机制的有效性。

本章余下部分将按如下结构组织：首先介绍本章相关背景知识，包括 NDN 链路层技术，应用层数据单元 (Application Data Unit, ADU)，分段 (fragmentation) 和分片 (segmentation)（第 4.1 章）；然后从简单模型开始建立 NDN 网络传输效率的数学模型，再松弛假设，并用马尔科夫链来分析丢帧率影响（第 4.2 章）；在该模型的基础上，设计了 ACM 机制（第 4.3 章）；然后讨论相关问题，包括 ACM 机制对缓存和请求聚合的影响、访问延时等（第 4.4 章）；之后介绍系统原型与模型验证实验（第 4.5 章）；最后总结本章内容（第 4.6 章）。

4.1 相关背景：NDN 链路技术 & 应用数据单元、分段与分片

4.1.1 NDN 的链路技术

NDN 可以运行任何提供包交换的技术之上，包括传统意义上的链路层技术，如 Ethernet、WiFi，网络层技术，如 IP，和传输层技术如 UDP, TCP；并且 NDN 团队专门为 NDN 设计了链路层协议 NDNLP^[112]。这些技术都可以作为 NDN 的“链路”^① 技术，来支撑两个 NDN 节点之间的报文传输。这些技术根据服务可靠性，可以分为两种：1) 可靠链路服务，如 TCP, WiFi, NDNLP，它们的特点是协议本身含有重传恢复机制，数据帧丢失可以依靠协议本身来恢复，并且对 NDN 是透明的；2) best-effort 链路服务，如 Ethernet, IP, UDP，它们的特点是一个数据帧的丢失将造成整个 NDN 层数据包的丢失，并且需要 NDN 来处理。

提供可靠的传输服务的链路技术具有一定的优势：因为在这种情况下，每个数据帧的丢失不会影响到整个数据包，同时它自己处理了丢包能简化 NDN 转发。但可靠性服务是有代价的，相比于 best-effort 协议，可靠协议依赖时钟和状态来实现分片监控、超时重传等功能，从而带来隐形的开销。更为严重的是，它可能会降低 NDN 中自适应的请求转发。。NDN 中的转发策略非常智能，在拥塞时候，丢包将触发 NDN 路由器迅速调整不同链路的优先级，并立即重新发送 Interest 到备选链路上从而从数据传输失败中快速恢复。可靠链路服务自己处理了丢包事件，并向 NDN 策略层（图2.4）屏蔽了网络不健康的信号，使得 NDN 的转发及时调用快速失败恢复，并且也不能观测到准确的链路状态，从而可能影响到后续一系列的数据传输。

NDN 更适合哪种类型的链路技术尚有待研究，影响因素可能包括丢帧率、数

^① 为避免混乱，后文直接成支撑 NDN 的技术为链路，而不区分具体是 TCP、UDP、IP、Ethernet、WiFi；链路层的数据传输单元为“数据帧”(frame)，NDN 层的数据传输单元为“数据包”(packet)。

据包尺寸、可选路径、转发策略等多个因素；但 NDN 本身可以运行在两类技术之上，所以本章将分对两类技术分别建模，两类技术在模型上的差异将非常的明显；同时本章建立的模型具有一般性，适合上文列举的所有的技术。在当前阶段，NDN 部署一般情况是 NDN over UDP/TCP over IP over Ethernet/WiFi；而现在 IP 网络基础设施上实际上的 MTU 大约是 1500 字节，因为超过这个值的报文往往被中间件如 NAT (Network Address Translation) 和防火墙丢弃；在这种部署条件下，NDN 链路层的 MTU 是 1472/1460 字节；另外，当前网络中典型的 end-to-end IP 包丢包率大约在 1% 到 2% 之间^[113,114]；考虑到 NDN 通过网络中的缓存能够降低转发跳数，NDN 中的某条确定的数据获取路径^①上的丢包率应该低于 2%。

4.1.2 NDN 中应用数据单元、分段和分片

NDN 以应用数据为中心，将应用数据单元 (Application Data Unit, ADU)，如 HTTP response，封装在 Data 里。ADU 尺寸应该只与应用需要相关，而无需让它产生的数据包受到 MTU 的限制。对于较大的 ADU，NDN 在网络层面可以进行分段 (segmentation)，每个分段都是一个独立的数据包，需要一个独立的 Interest 来请求才能返回该数据。

对于尺寸大于链路 MTU 的数据包，需要进行分片才能在链路上传输。这与传输大于 MTU 的 IP 数据包遇到的问题是一样的。现在 IP 采用的分片机制有两种：1) IPv4 采用逐跳路由器上分片，接收端组装 (Hop-by-Hop Fragmentation with End-Host Reassembly) 方法；2) IPv6 采用发送端分片，接收端组装 (End-to-End Fragmentation and Reassembly) 的方法。但这两种分片机制都无法迁移到 NDN 中，因为 1) Interest 包需要包含完整的信息，如果路由器接收了不完整的请求信息，它将无法匹配数据；2) 路由器需要完整的数据包来应答 PIT 表中的记录；3) 缓存的单位应该是整个数据包；4) 考虑到数据安全与验证的问题，需要完整的数据包的支持。因此，NDN 团队设计了在每一跳链路的发送端进行分片，而在该链路的接收端立即进行组装，也就是所谓的“逐跳的分片和组装”(Hop-by-Hop Fragmentation and Reassembly, HBH-FR)^[115]。HBH-FR 能够很好地满足 NDN 网络层操作原语的需要。

从机制上说，NDN 可以对每一个 ADU 进行分段，使的每个分段都不大于网络中链路上大部分 MTU（如当前网络 MTU，1500 字节），从而避免数据包在传输过程中需要分片的问题。但显然这种方法效率很低，一方面这将导致很多的 Data，增加平均开销（载荷比小）；另一方面发送和转发 Interest 的数量可能几倍地增长，考虑到路由器上内容匹配、PIT 查找、前缀查找的操作，这将极大增加路由器

^① NDN 中没有端到端的概念，本章以 Interest 转发的路径为单位，在每个具体路径上来衡量丢包率。

负担。在这种情况下，按多大的尺寸来分片 ADU 能够最优化传输效率成为关键所在。现在 NDN 应用根据经验值，往往使用 4096 字节作为每个 Data 的默认载荷尺寸，本章通过建立模型来回答这个问题，改变 4096 字节作为默认载荷尺寸的现状。如果 ADU 比较大，本章建立的模型可以指导 ADU 进行分段；例如在线视频应用中视频数据非常大，视频数据被切分变成不同的分段来进行传输。如果 ADU 较小，在必要的前提下，多个 ADU 可以压缩成一个 Data 来满足最大化 G2T 的要求，因此，本章的模型有比较强的应用前景。

4.2 数学建模

在本节将针对不同类型的链路技术进行建模，首先规范建模中所用的符号如表4.2所示。

表 4.2 变量符号

符号	意义
Δ	数据包中开销的尺寸
M, M_{ij}	第 j 条传输路径上第 i 个链路 ($link_{ij}$) 的 MTU
Ω, Ω_{ij}	$link_{ij}$ 的链路丢帧率
p	数据包中负载尺寸
m	一个数据包分片的数量
ρ	数据包丢包率
P	能最大化 G2T 的最优的负载尺寸

对于给定的应用和网络， Δ, M 几乎都是固定的，先假设：1) 所有的链路都提供 best-effort 服务；2) Ω 也是确定的，至少有比较稳定的平均值。由这两点假设，本节先建立了一个简单的通信效率模型，，然后将在后文松弛这两个假设。 m 和 ρ 是可以从已有变量推导出的中间变量， p 是模型中的变量， $(P + \Delta)$ 是最优化的包尺寸。由于 Δ 是固定的，本节将用模型变量 p 来代替包尺寸来做分析，这样更为直观。

如图4.2 所示，consumer (C) 想要获得目标数据，由于 NDN 通信允许多路径，多数据源，网络缓存，其可能路径有 $\langle P_1 - C \rangle$, $\langle P_1 - R_1 - C \rangle$, $\langle P_2 - R_2 - C \rangle$ 和 $\langle R_1 - C \rangle$, $\langle R_2 - C \rangle$ 。在本节剩下的部分将以图4.2 为例子，由易到难建立通信效率 G2T 的模型：首先建立通过一跳获得数据的效率模型 (图4.2 中的路径 $\langle P_1 - C \rangle$,

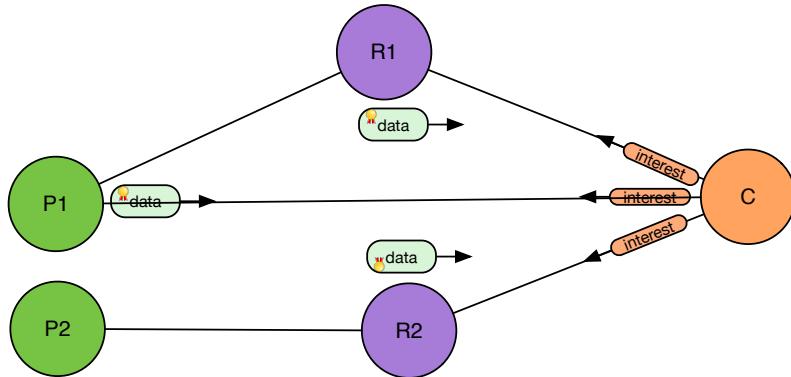


图 4.2 拓扑范例: C 想要从 NDN 网络中获得由 P1 或 P2 生成的目标数据。由于 NDN 通信允许多路径, 多数据源, 网络缓存, C 可能通过路径 $\langle P_1 - C \rangle$, $\langle P_1 - R_1 - C \rangle$, $\langle P_2 - R_2 - C \rangle$ 或 $\langle R_1 - C \rangle$, $\langle R_2 - C \rangle$ 获得数据, 注意到 C 可以直接从路由器 $R_1 \& R_2$ 中获得数据。这体现了 NDN 通信中 consumer 没有确定的对端,

$\langle R_1 - C \rangle$, $\langle R_2 - C \rangle$; 然后建立多跳通信效率模型 ($\langle P_1 - R_1 - C \rangle$, $\langle P_2 - R_2 - C \rangle$), 最后考虑所有的可能的通信路径, 包括多数据源、网络缓存和请求聚合的情况, 建立模型。建立模型的过程反映了 NDN 与 TCP/IP 本质上的不同: 在 TCP/IP 中有确定的发送方和接收方, 中间路径也往往由路由协议唯一确定; 而在 NDN 中, 数据可以从多个节点、多个路径获得, 且没有确定的通信对端和传输路径。本节最后两小节先研究了异构链路通信的情况, 即通信中包含可靠链路和 best-effort 链路(第 4.2.4 章); 然后分析当丢包率 Ω 不是固定, 而是符合 Gilbert Loss Model^[116] 的情况, 并用连续时间的马尔科夫链来分析这个问题(第 4.2.5 章)。

4.2.1 单跳获取数据

考虑一个尺寸为 $(\Delta + p)$ 的数据包, 传输该数据包需要的数据帧数量为

$$m = \frac{\Delta + p}{M}, \quad (4-2)$$

更准确地说, 应该是

$$m = \lceil \frac{\Delta + p}{M} \rceil. \quad (4-3)$$

为简单起见, 本章在推导中用公式 (4-2), 将所得结果 p 根据公式 (4-3) 进行修正, 使得 m 为一个整数。

链路上任何一帧丢失都将导致该数据包传输失败, 因此, 该数据包丢包率为

$$\rho = 1 - (1 - \Omega)^m. \quad (4-4)$$

更进一步，G2T 为

$$\mathbf{G2T} = \frac{p * (1 - \Omega)^{\frac{p+\Delta}{M}}}{(\Delta + p)}. \quad (4-5)$$

在公式 (4-5) 中， Ω, Δ, M 视为常数， p 作为变量，根据这个公式，可以得出能最大化 G2T 的 p 值（即表4.2 中的 P ）

$$P = \frac{\Delta \ln(1 - \Omega) + \sqrt{(\Delta \ln(1 - \Omega))^2 - 4\Delta M \ln(1 - \Omega)}}{-2 \ln(1 - \Omega)}. \quad (4-6)$$

根据公式 (4-6) 得到的值需要进行修正，这个修正在工程上意味着填满每一个数据帧， P 的可能取值为 MTU 的整数倍，即 $P = MTU \times i, i \in \mathbb{N}$ ，则

$$P = \left\{ \lfloor \frac{p}{MTU} \rfloor \times MTU, \lceil \frac{p}{MTU} \rceil \times MTU \right\} \Big|_{max\{G2T\}}. \quad (4-7)$$

下面对现有模型展开深入的分析。

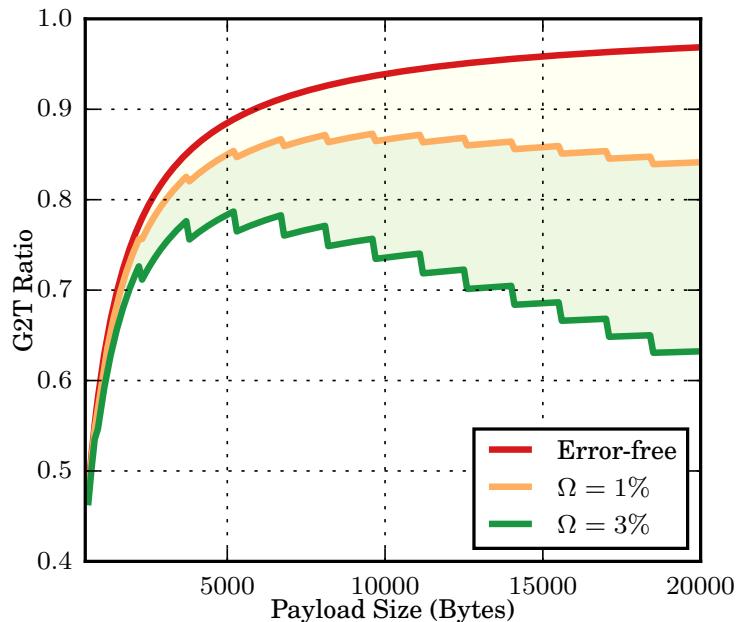


图 4.3 当前典型 NDN 部署环境下的 G2T，即 NDN over UDP over IP over Ethernet。在有丢包率的情况下，G2T 是包尺寸的凸函数，即存在某个包尺寸能最大化 G2T。上图中 $\Delta = 650, M = 1472$ 。

图4.3 展示了不同包尺寸的 G2T 值，当网络没有丢帧时（红色曲线，丢帧率为

0), G2T 是关于 p 的单调递增函数; 但网络中一旦有丢帧的情况, G2T 曲线则有峰值。橙色曲线 ($\Omega = 1\%$) 的峰值为 (8832, 86.8%)。从图4.3 还可以总结出 G2T 变化范围较大, 当 $\Omega = 1\%$ 而载荷范围从 500 到 20000 字节时, G2T 的变化范围是从 47% 上升到 88%, 这表明通过调整包尺寸, 可以获得比较大的性能优化。

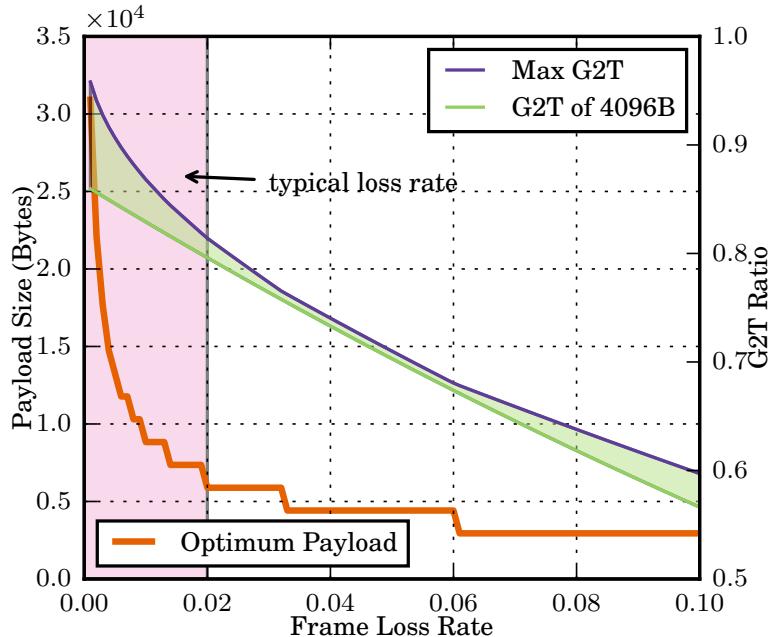


图 4.4 X 轴为丢帧率, 左边 Y 轴为最优化的载荷尺寸, 右边 Y 轴为对应的最大化的 G2T。丢帧率变化范围为 0.1% 到 11%, 丢帧率典型值应该低于 2%, 在此范围内最优化载荷尺寸抖动较大, 当前常用的载荷尺寸 4096 字节对应的 G2T 与最大化的 G2T 差距明显, 有较大的优化空间。上图中 $\Delta = 650, M = 1472$ 。

图4.4 展示了当丢帧率变化范围为 0.1% 到 10% 最优的载荷尺寸与对应的最大化 G2T (紫色曲线) 和载荷为 4096 字节情况下的 G2T (绿色曲线)。可以发现, 当丢帧率从 0.1% 增加到 2% 时, 最优的载荷尺寸下降非常迅速; 这样的丢帧率范围包含了当前网络端到端的丢包率范围, 说明了有比较强的需求要调整包尺寸。

4.2.2 多跳获得数据

大部分情况下, 数据需要经过多个中间节点才能从它原来的位置传输到 consumer, 假设这条传输路径上一共有 N 条链路, Ω_i, M_i 分别是第 i 跳的丢帧率和 MTU。这种情况下, 丢包率为

$$\rho = 1 - \prod_{i=1}^{i=N} (1 - \Omega_i)^{\frac{p+\Delta}{M_i}}, \quad (4-8)$$

于是推导出

$$G2T = \frac{p}{p + \Delta} \cdot \prod_{i=1}^{i=N} (1 - \Omega_i)^{\frac{p+\Delta}{M_i}},$$

则 P

$$P = \frac{-E\Delta + \sqrt{E^2\Delta^2 + 4E\Delta}}{2E}, \quad (4-9)$$

其中 E 为

$$E = - \sum_{i=1}^{i=N} \frac{\ln(1 - \Omega_i)}{M_i}. \quad (4-10)$$

把公式(4-6)与公式(4-9)和公式(4-10)对比， P 的表达式非常相似；其实公式(4-6)完全可以用公式(4-9)来表示，因为单跳其实是多跳的一个特殊情况。 E 的表达式公式(4-10)标明不同链路上丢包的累积效果，因为任何一跳上的任何一个丢包都导致数据传输的失败。

4.2.3 多路径传输，支持多数据源，网络缓存和请求聚合

从 consumer 的角度看，它们可以从不同路径上获得数据（包括中间路由器），假设一共有 K 条链路（考虑到多数据源、中间节点缓存、多路径等所有情况），第 j 条路径由 N_j 个链路组成，并且该路径承担的流量在 consumer 数据获取的总流量中的比例是 w_j ，则

$$G2T = \frac{p}{p + \Delta} \cdot \sum_{j=1}^{j=K} \left[w_j \cdot \prod_{i=1}^{i=N_j} (1 - \Omega_{ij})^{\frac{p+\Delta}{M_{ij}}} \right], \quad (4-11)$$

其中

$$\sum_{j=1}^{j=K} w_j = 1.$$

通过下面的式子，可以推导出 P 的数值解

$$0 = \sum_{j=1}^{j=K} \left\{ w_j D_j^{p+\Delta} [\ln D_j p^2 + \ln D_j \Delta p + \Delta] \right\}, \quad (4-12)$$

其中 D_j 为

$$D_j = \prod_{i=1}^{i=N_j} (1 - \Omega_{ij})^{\frac{1}{M_{ij}}}.$$

如果所有的 MTU 都是相同的（如 1500 字节），推导出来的包尺寸经过修正后与单跳情况得到的数值可以归入同一组数值集合中。

上面的分析同样适合 consumer 从多数据源和网络缓存中获得目标数据的情况。从 consumer 的角度看，它并不关心 Interest 遇到名字匹配的 Data 的节点，这个节点可以是任意 producer，也可以是中间路由器，本文称这个节点为“碰撞点”。对一次具体的数据获取而言，Interest 经过的从 consumer 到碰撞点是一条具体的传输路径，而碰撞点是哪个数据源或者是中间路由器无关，因此上面的模型可以描述多数据源和网络缓存的情况。针对请求聚合的情况，某个具体 Interest 依然有一条确切的路径，无论是该 Interest 聚合后被丢弃（此时碰撞点视为聚合点），还是该 Interest 聚合后进一步转发（别的 Interest 被丢弃），因此，上面的模型也可以描述请求聚合的情况。总的来说，多数据源，网络缓存和请求聚合都可以视为是多路径传输的一种情况，从 consumer 的角度看没有本质的差异。

4.2.4 松弛假设：异构链路

在真实的网络中，可能有多种异构链路，包括可靠链路和 best-effort 链路。一条传输路径就可能是由多个异构链路组成的。目前本章的模型是建立在链路是 best-effort 情况下，现在松弛这个假设。对于可靠链路，丢帧事件直接由链路本身恢复，所以 G2T 非常直接：

$$G2T = \frac{p \cdot (1 - \Omega)}{p + \Delta}. \quad (4-13)$$

图4.5 展示了 UDP, TCP 和无错以太网信道三者 G2T 比较。TCP 和无错以太网的 G2T 值都是随载荷尺寸单调递增的^①。但是从 consumer 的角度看，丢包事件直接被信道本身恢复，而不能启动 NDN 层快速失败恢复，导致该丢包事件不能为 consumer “检测” 到。因此，在 consumer 能够测量到的 G2T 依然可以用公式 (4-11) 描述。

但理论上依然可以用数学模型描述异构链路下准确的 G2T，假设第 j 条路径

^① 此处没有考虑传输超时带来的影响，在真正的 TCP/WiFi 链路上，超时超过阈值，将导致传输失败。

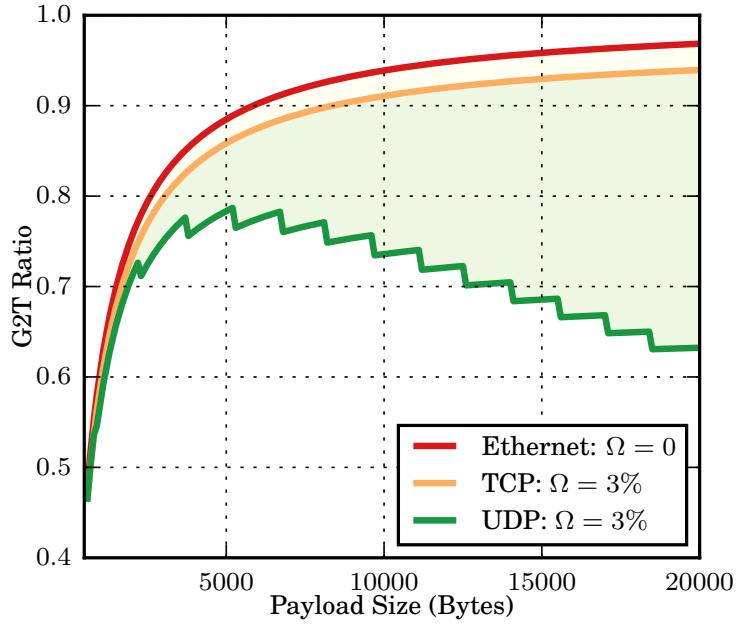


图 4.5 不同链路服务下的 G2T。完美链路（丢帧率为 0）和可靠传输链路下，G2T 是包尺寸的递增函数。三种情况下的 Δ 取值分别是 1500 (Ethernet)、1460 (TCP)、1472 (UDP) 字节。

上一共有 T_j 个可靠链路（有 N_j 个 best-effort 链路，这一假设保持不变），则

$$G2T = \frac{p}{p + \Delta} \cdot \sum_{j=1}^{j=K} \left[w_j \cdot \prod_{i=1}^{i=N_j} (1 - \Omega_{ij})^{\frac{p+\Delta}{M_{ij}}} \cdot \prod_{i=1}^{i=T_j} (1 - \Omega_{ij}) \right]. \quad (4-14)$$

公式(4-14)与公式(4-11)非常相似，此处把公式(4-14)中的 w_j 与 $\prod_{i=1}^{i=T_j} (1 - \Omega_{ij})$ 合并在一起，则两个方程有完全一样的形式：

$$G2T = \frac{p}{p + \Delta} \cdot \sum_{j=1}^{j=K} \left[w'_j \cdot \prod_{i=1}^{i=N_j} (1 - \Omega_{ij})^{\frac{p+\Delta}{M_{ij}}} \right], \quad (4-15)$$

其中 w'_j 为

$$w'_j = w_j \cdot \prod_{i=1}^{i=T_j} (1 - \Omega_{ij}).$$

因此，最优的包尺寸可以用过公式(4-12)来计算。需要强调的是，公式(4-15)描述了准确的 G2T，而公式(4-11)描述了在 consumer 能够测量到的 G2T。

4.2.5 松弛假设：丢帧率 (Ω) 不是常数

在之前的简化模型中，本章假设丢帧率是一个常数（或者至少有稳定的平均值），现在对这一假设进行松弛。丢帧率为常数意味着每帧丢失概率相同且独立，而是实际中丢帧事件往往不是独立的，一般来说，前面一个帧丢弃的情况下后一帧丢弃的概率非常大 (burst loss)，目前 Gilbert Loss Model^[116] 被广泛用于描述这种情况。Gilbert Loss Model 可以用两状态连续时间的马尔科夫链 (two-state continuous time Markov chain) 来表达，为了进一步的分析，本节先定义表4.3 中的符号。

表 4.3 马尔科夫链分析中用到的符号

符号	意义
$\chi_{ij}(t) \in \{G, B\}$	表示在时刻 t, 链路 $link_{ij}$ 的状态; G 代表 Good, B 代表 Bad
π_{ij}^G, π_{ij}^B	链路 $link_{ij}$ 好和坏的静态概率 (stationary probabilities)
ξ_{ij}^G, ξ_{ij}^B	从 B 到 G, 从 G 到 B 的转移概率 (transition probability)
$f_{ij}^{s,t}(\tau) (s, t \in \{G, B\})$	链路 $link_{ij}$ 上在时间 τ 内从状态 s 转到状态 t 的转移概率
$\tau_{ij}^{n,n+1}$	链路 $link_{ij}$ 第 n 个与 (n+1) 个帧的传输的时间间隔
c_{ij}	m 元组, 表示链路 $link_{ij}$ 传输 m 帧时对应每个帧传输的状态
$c_{ij}^n \in \{G, B\}$	c_{ij} 这个 m 元组中第 n 个元素
$0 \leq L(c_{ij}) \leq m$	组合 c_{ij} 组合在传输时候丢帧数量
$\mathbb{P}(c_{ij})$	组合 c_{ij} 传输失败的概率

$\chi_{ij}(t) \in \{G, B\}$ (G 代表 Good, B 代表 Bad) 表示在时刻 t, 链路 $link_{ij}$ (第 j 条路径的第 i 条链路) 的状态；如果 $\chi_{ij}(t) = G$, 则该帧被成功传输，反之则没有； π_{ij}^G 与 π_{ij}^B 分别表示链路好和坏的静态概率 (stationary probabilities)。 ξ_{ij}^G and ξ_{ij}^B 表示从 B 到 G, 从 G 到 B 的转移概率 (transition probability)。在这种情况下，两个独立的变量可以确定一个连续时间马尔科夫链的丢帧模型：(1) 信道丢失概率 π_{ij}^B ；(2) 平均丢失长度 $1/\xi_{ij}^G$ 。可以得到：

$$\pi_{ij}^G = \frac{\xi_{ij}^G}{\xi_{ij}^G + \xi_{ij}^B},$$

$$\pi_{ij}^B = \frac{\xi_{ij}^B}{\xi_{ij}^G + \xi_{ij}^B}.$$

c_{ij} 是一个 m 元组，表示链路 $link_{ij}$ 传输一个包含 m 个帧的 Data 时对应每个帧传输的状态， $c_{ij}^n \in \{G, B\}$ 是 m 元组中的第 n 个状态。通过考虑 c_{ij} 的所有可能组

合，得到丢帧率的数学期望：

$$\Omega_{ij} = \frac{1}{m} \sum_{\forall c_{ij}} [L(c_{ij}) \cdot \mathbb{P}(c_{ij})], \quad (4-16)$$

其中 $L(c_{ij})$ ($0 \leq L(c_{ij}) \leq m$) 表示丢帧的个数， $\mathbb{P}(c_{ij})$ 表示组合 c_{ij} 传输为失败的概率。进一步， $L(c_{ij})$ 可以表示为

$$L(c_{ij}) = \sum_{n=1}^m 1_{\{c_{ij}^n = B\}}. \quad (4-17)$$

用 $f_{ij}^{s,t}(\tau)$ ($s, t \in \{G, B\}$) 表示链路 $link_{ij}$ 上在时间 τ 内从状态 s 转到状态 t 的转移概率，可以得到

$$f_{ij}^{s,t}(\tau) = \mathbb{P}\{\chi_{ij}(\tau) = t | \chi_{ij}(0) = s\}. \quad (4-18)$$

根据经典的连续时间马尔科夫链瞬态行为 (transient behavior)，得到时间 τ 内的转移矩阵

$$\begin{bmatrix} f_{ij}^{G,G}(\tau) & f_{ij}^{G,B}(\tau) \\ f_{ij}^{B,G}(\tau) & f_{ij}^{B,B}(\tau) \end{bmatrix} = \begin{bmatrix} \pi_{ij}^G + \pi_{ij}^B \cdot \kappa_{ij} & \pi_{ij}^B - \pi_{ij}^B \cdot \kappa_{ij} \\ \pi_{ij}^G - \pi_{ij}^G \cdot \kappa_{ij} & \pi_{ij}^B + \pi_{ij}^G \cdot \kappa_{ij} \end{bmatrix},$$

其中 $\kappa_{ij} = \exp[-(\xi_{ij}^B + \xi_{ij}^G) \cdot \tau]$ 。用 $\tau_{ij}^{n,n+1}$ 第 n 个与 $(n+1)$ 个帧的传输间隔，可以得到

$$\mathbb{P}(c_{ij}) = \pi_{ij}^{c_{ij}^1} \cdot \prod_{n=1}^{m-1} \left[f_{ij}^{c_{ij}^n, c_{ij}^{n+1}}(\tau_{ij}^{n,n+1}) \right]. \quad (4-19)$$

最后，推导出

$$\Omega_{ij} = \frac{1}{m} \sum_{\forall c_{ij}} \left\{ L(c_{ij}) \cdot \pi_{ij}^{c_{ij}^1} \cdot \prod_{n=1}^{m-1} \left[f_{ij}^{c_{ij}^n, c_{ij}^{n+1}}(\tau_{ij}^{n,n+1}) \right] \right\}. \quad (4-20)$$

链路 $link_{ij}$ 的准确丢帧率可以通过公式 (4-20) 表达，当这个随机过程趋于稳态时，由公式 (4-20) 的结果就由信道丢失率和平均丢帧长度决定^[117]。通过替换公式 (4-15) 中的 Ω_{ij} ，准确的 G2T 可以由公式 (4-15) 表示。

但是我们也不应该小看 Ω 被当做固定参数的简化模型，实际上在工程中简化模型更便于应用决定最优的数据包尺寸。

4.3 ACM 机制

到目前为止，本章建立了一个模型能够推导出最优的包尺寸从而最大化传输效率指标 G2T。这一节根据该模型设计了 ACM (Adaptive Chunk Mode) 机制，它使得应用可以动态地改变传输数据包的尺寸，从而提高效率。出于两个原因，本章用简化模型的公式 (4-6) 和公式 (4-9) 来决定最优包尺寸，而不用更精确的模型：1) 作为一个工程方法，简化模型已经能够推导出一个接近于最优包尺寸的值；2) 更重要的是，在工程中通常没有精确模型需要的所有参数。

ACM 可以用于不同的场景下，例如点到点应用和大规模数据分发场景。对于点到点的应用，可以测量丢包率，并且实时调整最优包尺寸。而对于大规模数据分发的情况，CSP 用广泛分布在世界各地的数据冗余备份来服务数量庞大的用户群。这种情况下，为每一个客户都定制最佳的包尺寸会极大地增加服务器的压力，可行的办法是把数据分成几类不同的尺寸预存在服务器上，而让客户端根据自身情况选择获取最优的尺寸。鉴于根据模型推导出来的最优结果是离散的，这样的做法也可以满足模型的需要。

因为 NDN 的通信是由请求者驱动的，是 consumer 来决定它需要的内容是什么；所以 ACM 机制让 consumer 去估计当前丢包率并计算最佳包尺寸，再通过 Interest 显式地表达请求数据包的尺寸。

4.3.1 ACM 命名和交互机制

数据命名是 NDN 应用首要的设计元素，ACM 利用了数据名中的两个名字部件：包尺寸和分段号，而其它部分依然由应用自己去决定。NDN 有两种不同的分段标识 (segment marker)^[118]。第一种 “0x00” 标识顺序编号 (sequence number) 为分段号，第二种 “0xFB” 标识字节偏移量 (byte offset) 为分段号；ACM 创建了一种新的标识，尺寸标识 “0xAC”，标识请求数据包的尺寸。当顺序编号为分段号时，载荷的内容数据起始偏移量为 $\text{ChunkSize} \times \text{SequenceNumber}$ ，名字中的其它部分可以承担应用层面的含义，图4.6 展示了两个 ACM 命名的例子。

ACM 遵守 NDN 中的通信规则，consumer 必须发送正确的数据名以获得相应的数据。在应用启动阶段由于缺乏足够的网络层信息，consumer 根据预设的默认尺寸请求数据包；经过几轮 Interest / Data 交换后，consumer 可以通过下一小节介绍的包尺寸决定机制确定最优的尺寸。

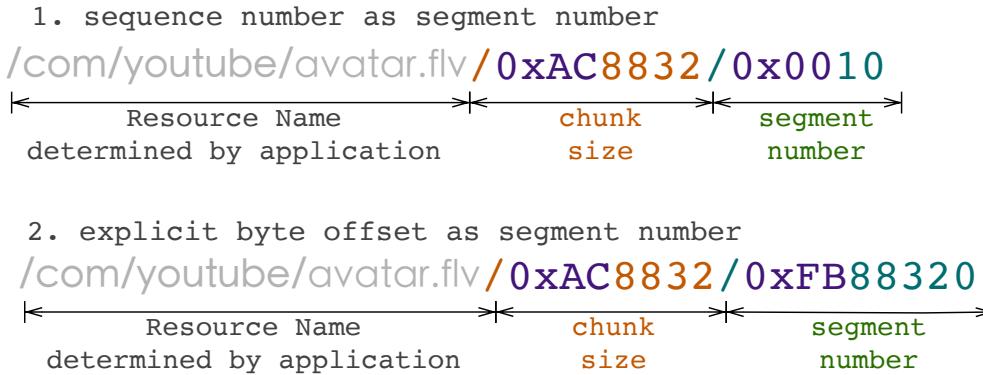


图 4.6 ACM 命名机制范例：该名字标识了 Youtube 服务器上 avatar.flv 这个视频从 88320 字节这个文件偏移量之后的 8832 字节数据。“0xAC”是 ACM 创建的尺寸标识。

4.3.2 包尺寸决定机制

在 ACM 机制中，consumer 可以用两种不同的办法来确定最优的包尺寸：主动探测和丢帧率估计。因为最优的尺寸是离散的，所以可以通过发送不同尺寸的请求并测量 G2T 来确定哪一个尺寸效果最优。这种方法非常简单，当 NDN 转发非常复杂时候可以使用这种方法。主动探测需要周期性的进行，以适应动态的网络环境。

丢帧率 (Ω) 是模型中用于确定最优尺寸的重要参数，通过丢帧率估计来确定最优包尺寸适合于网络环境变化比较剧烈和频繁的场景中。对于单跳情况下的通信，可以用公式 (4-4) 来确定丢帧率；因为 consumer 不能区分出数据包传输的路径，所以本章用公式 (4-8) 辅以一定程度的修正来确定丢帧率。由于网络中存在的快速失败恢复机制，这种估计的丢帧率会比实际中的要小，因此本章用下面展示的方法予以修正。根据公式 (4-4)，推导出两种不等式

$$\Omega_{lower} = \frac{ChunkLoss\#}{RequestedPacket\#} = \frac{1 - (1 - \Omega)^m}{m} \leq \Omega \quad (4-21)$$

$$\Omega_{upper} = \frac{ChunkLoss\#}{RequestedChunk\#} = 1 - (1 - \Omega)^m \geq \Omega \quad (4-22)$$

其中 Ω_{lower} , Ω_{upper} 分别是丢帧率的下限和上限；通过 Ω_{lower} 和 Ω_{upper} ，可以得到 Ω 的范围，此处用线性关系来估计 Ω :

$$\Omega = \alpha \cdot \Omega_{lower} + (1 - \alpha) \cdot \Omega_{upper}, \quad (4-23)$$

其中 $\alpha \in [0, 1]$ 。这样，丢帧率可以通过公式 (4-23) 得到修正。根据目前已有的实

验结果，本章在应用中选择 $1/8$ 作为 α 的值。

4.4 相关讨论

4.4.1 动态变化的尺寸影响

网络层缓存和 Interest 聚合对于大规模数据分发有非常重要的意义，也是 NDN 优势之一；但它们的性能依赖于数据请求的分布：越多相同的请求，缓存/聚合性能越好。ACM 要求 consumer 根据网络状态请求不同尺寸的数据包，导致同一个应用文件对应多种不同的 Interest 集合。这将降低发送相同请求的可能性，从而降低缓存和聚合性能。

但基于两个原因，有理由认为这种副作用的影响有限：1) 最优尺寸是离散且数量非常有限的（相比于热点内容的 consumer 数量来说）；2) 相同或者相邻网络里的 consumer，更可能会使用相同的包尺寸，故而不影响缓存/聚合的效果。而且，应用可以在应用层对不同尺寸的数量予以限制，如允许 3 种尺寸，从而将这种负面影响降低到可以接受的程度。

4.4.2 访问延时

本章的模型是以最小的带宽代价为优化目标，而没有把访问延时纳入考量范畴。但对于固定带宽的信道而言，最小的带宽消耗也意味着最小的传输延时。对于一个带宽为 B 的链路，传输尺寸为 p 的 ADU 的传输延时的简单模型为

$$\text{delay} = \frac{p}{G2T * B}. \quad (4-24)$$

由公式 (4-24) 可知在单个链路上的延时与 G2T 反比；也就是说增大 G2T 同时也减低了内容的访问延时（不考虑其它因素）。这个结论对非实时的应用，例如 ftp, video-on-demand (有 buffer) 有重大价值，它表明可以同时最小化总体延时且最大化传输效率。

但对于实时应用，例如实时视频，问题变得复杂。根据公式 (4-24)，减小每个 ADU 的尺寸可以有效降低延时（这也导致 ADU 的数量更多，以及更长的总的传输延时）。因此，实时应用会在应用层尽量减小每个 ADU 的尺寸，例如把每个视频/音频的采样都作为一个 ADU。这样的结果是，每个单个 ADU 很小，传输时间很短，但传输效率不高。传输时延与效率两者的平衡需要应用自己来衡量。

4.4.3 Interest 的带宽消耗

NDN 是采用 Interest / Data 交换的形式来获得数据，通常 Interest 尺寸较小（一般 Interest 几十个字节），本章的模型没有把 Interest 消耗的带宽纳入模型中。由于 Interest / Data 的交换是按照对称性路径进行的，所以在需要准确计算时，可以把 Interest 的尺寸归入到数据包开销中，不用修改现有模型，而获得更为准确的结果。在这种情况下，Interest 每一比特的丢失也导致传输失败，这与 Data 中任意比特丢失是完全等价的，这也是符合 Interest / Data 交换通信机制。

4.5 系统原型与模型验证

4.5.1 ACM 原型与应用

本节实现了一个支持 ACM 机制的 NDN 功能库 `ndnflow`^[119]，它提供 API 可以用于 NDN 应用开发；并且基于 `ndnflow` 开发了一个在线视频播放器 `nPlayer`^[37]，它的界面如图4.7所示。`nPlayer` 可以同时播放两路视频，并且把数据获取的实时信息动态地展示在底部的图片中。图4.8展示了某次运行中两路视频包尺寸和传输比特速率的变化，其中一路采用 ACM 决定包尺寸（绿色曲线），另一路使用 4096 字节作为负载尺寸情况下（蓝色曲线），绿线比特率很快降为 0，是因为已经完成了视频文件传输。

本节用 `ndnflow` 和 `nPlayer` 在真实的网络上验证 ACM 的优势（第 4.5.2 章）；但由于真实网络环境难以控制，除 `ndnflow` 和 `nPlayer` 之外，本节还使用 `ndnSIM`^[110] 来进行模拟验证本章的模型（第 4.5.3 章）。由于实际系统中用到的 NDN 协议栈 NDN Forwarder^[12,13] 和 `ndnSIM` 都把 9000 字节作为 Data 尺寸的上限，本节在实验中的包尺寸将不超过这个限制。

4.5.2 ACM 机制有效性

为了验证 ACM 机制是否最大化 G2T，本节在真实的网络上通过 ACM 库 `ndnflow` 来做实验。首先本节比较了几个不同的包尺寸决定机制：1) 固定包大小，分别为 1K、4096、8.5K 字节；2) 动态调整包大小，分别为 ACM、ACM -1K、ACM +1K，其中 ACM 是指通过 ACM 机制来决定包尺寸；而 ACM -1K 是指在 ACM 决定的尺寸基础上减去 1K 字节；ACM +1K 则是在 ACM 决定的尺寸上加上 1K 字节。六种不同的情况最后的 G2T 如图4.9(a) 所示，显然固定包大小机制性能最差，ACM 效率最高，而改变 ACM 机制决定的尺寸 (ACM +1K, ACM -1K) 都降低了效率，由此可见，ACM 的确获得最好的效果，实验中，ACM 机制的 G2T 比 4096 字

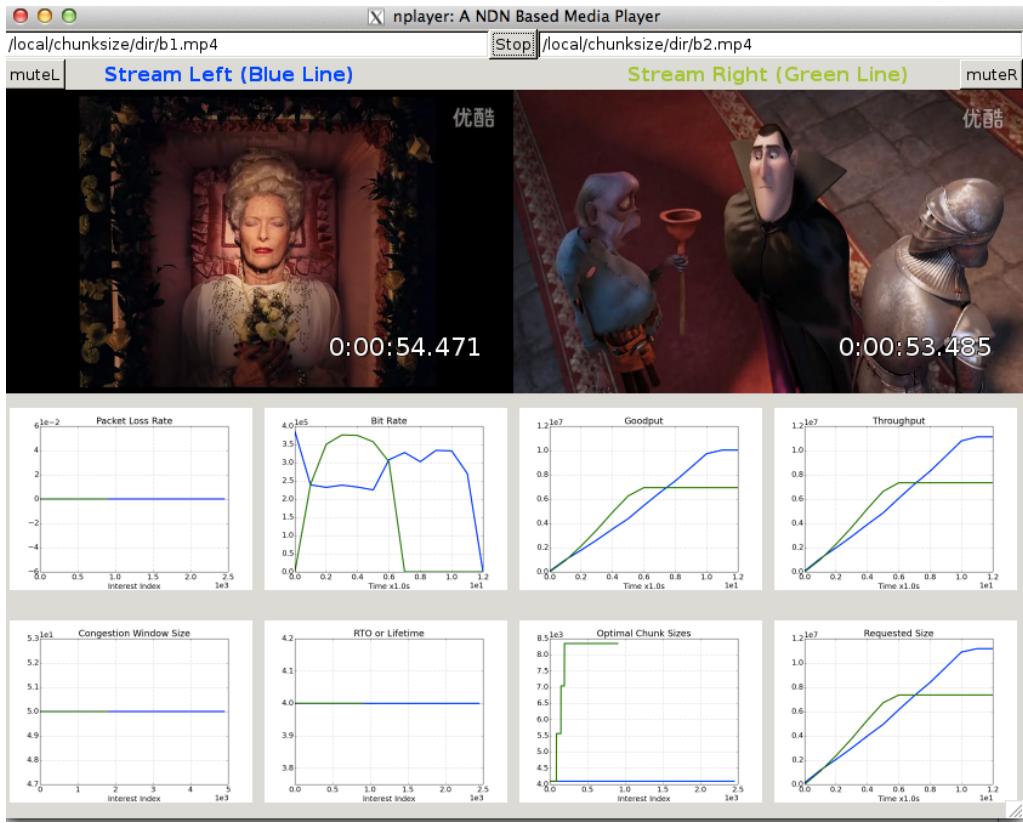


图 4.7 nPlayer，基于 NDN 的在线视频播放器，它可以同时播放两路网络视频，并且把数据获取的实时信息动态地展示在底部的图片中。

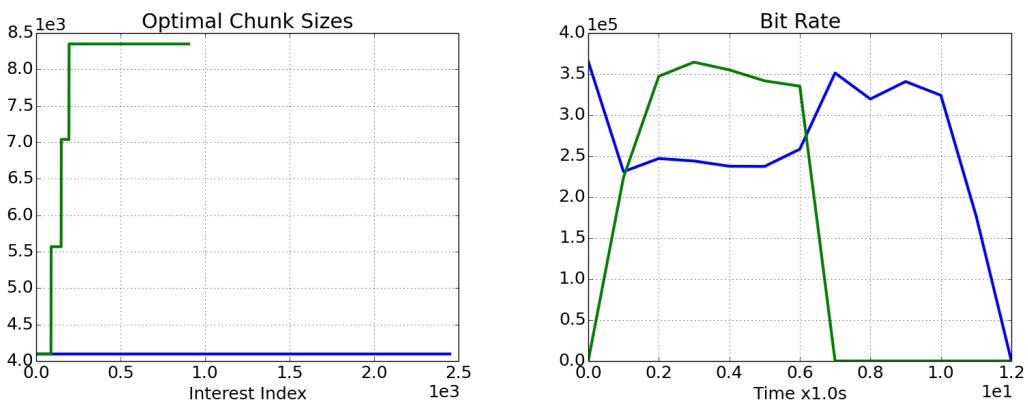


图 4.8 nPlayer 运行中的网络传输信息，其中绿色线标识使用了 ACM 机制的视频获取，蓝色的线使用 4096 字节作为默认的载荷尺寸。绿线比特率很快降为 0，是因为已经完成了视频文件传输。

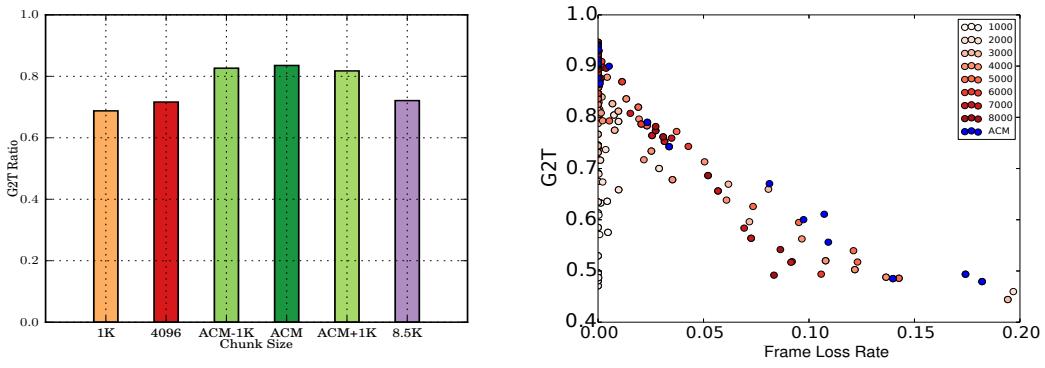


图 4.9 ACM 与其它包尺寸决定策略效率比较: (a) 表明 ACM G2T 最高, 增大或者减小 ACM 决定的尺寸 (ACM+1K, ACM-1K) 都会减小 G2T, 固定尺寸 4096 (常用尺寸), 1K, 8.5K 效率都低于 ACM ; (b) 表明 ACM 适应于各种复杂的网络环境, 都很取得很好的效果。

节的 G2T 高 12%。这组实验是在相同的网络环境下, 在比较短的时间内完成, 本节假设它们是在相对一致的环境下完成。

更进一步, 本节在各种不同的网络环境下进行实验, 来验证 ACM 的有效性。本节在不同网络环境, 不同时间分别来测量 G2T。作为对比, 本节使用了非常广泛的包尺寸范围 (1K 到 8K 字节), 实验结果如图4.9(b) 所示。总体而言, 在各种不同的网络环境下, ACM 总能获得最高的效率。

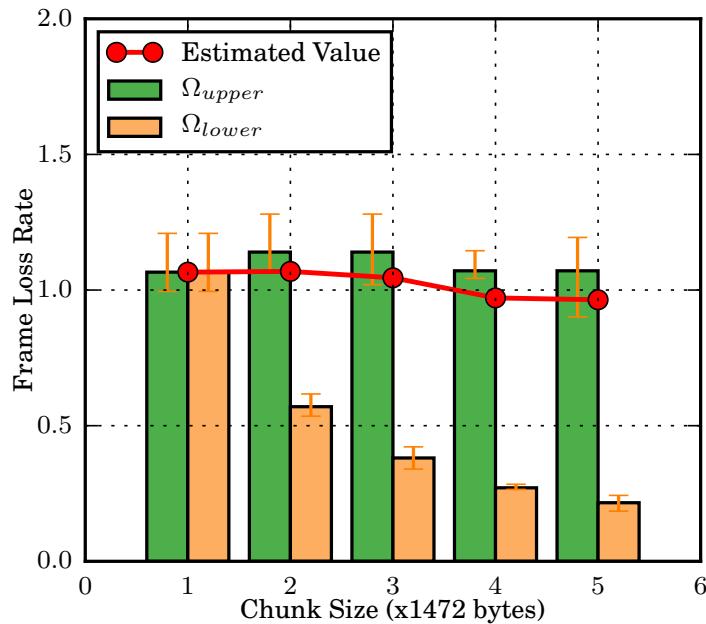


图 4.10 丢包率测量: 包尺寸为 MTU 时候, 丢帧率和丢包率等价, 从而可以作为基准值, 根据实验结果, 选取 $\alpha = 1/8$ 取得了较为稳定的丢包率 (红色曲线)。

丢帧率是 ACM 机制的重要参数，本节也评估了第 4.3 章提出的丢帧率估计机制，以及公式(4-23)中的 α 。本节包尺寸变换范围为 1xMTU(1472 字节)到 5xMTU。当尺寸为 1xMTU 时，丢包率与丢帧率是等价的(没有分片)，由于本节在较短的时间内完成了实验，1xMTU 对应的丢帧率可以最为基准线。本节测量了不同包尺寸下的 Ω_{lower} 和 Ω_{upper} 如图4.10 所示。在实验中， Ω_{upper} 相对稳定且靠近基准值，而 Ω_{lower} 比基准值低且变化剧烈，因此测量值应该靠近 Ω_{upper} 。基于此，本章选择了 1/8 作为 α 的值；在这种情况下， Ω 估计值如图4.10 中红线所示。该实验重复多次，图中了平均值/最大值/最小值。需要说明的是，目前 α 值是基于目前有限次数的实验结果所得，为了更佳的估计机制和更准确的 α ，需要更多的工作。

4.5.3 模型验证

本节展示一系列的模拟结果来验证本章建立的模型，包括简化模型和准确模型。模型验证将按照建模过程中的顺序，以图4.2为实验拓扑，分单跳、多跳、多路径的场景分别验证；然后松弛假设，模拟异构链路的情况和 Gilbert Loss Model 的场景。在实验中，每个数据包的开销(Δ , 包括 Interest 开销)为 650 字节。本节测试在不同场景下用不同包尺寸进行数据传输，实验测量值与理论值的一致程度。实验的结果用一些图来展示，横轴表示载荷尺寸，在这些图中绿色长条表示从 consumer 端测量得到的 G2T，对于每种场景，实验都运行了 20 次，绿色长条的高度展示了平均值，并且图中用浅红色短线展示了最大值和最小值。红色曲线表示根据本章的模型计算出的对应载荷尺寸的 G2T 理论值。

图4.11 展示了通过一跳内容获得数据的效率模型，丢帧率分别是 1%、3%、5%；图4.12展示了通过多跳获得内容的效率模型；图4.13展示了多条路径获得内容的效率模型；图4.13(a) - 图4.13(f) 是针对同一个数据源的多路径情况；图4.13(g) - 图4.13(i) 是针对多个数据源的多路径情况；图4.13(d) - 图4.13(f) 是针对不同流量比例情况。图4.14展示在异构链路上的获取内容的效率模型；图4.15展示了当丢包率符合 Gilbert Loss Model 时候的效率模型。实验结果与理论数据一致程度非常高，在实验中，实验数据与理论数据的相对误差^①是 0.9%，标准方差是 0.00012。

上述模型验证的结果是基于 NDN 模拟器 ndnSIM^[110,120]，虽然模拟器本质上依赖于数学计算，而不同于真正的网络测试；但 ndnSIM 中实现了 NDN 层、链路层相关协议，并且在系统模拟中引入了随机性(随机数)，它的实验结果具有一定普遍性，这也是 ndnSIM 广泛使用的原因。更进一步，本节将在后文依托于真实的网络，来验证应用了该模型的 ACM 机制的效率。

^① 相对误差定义为： $\frac{|ExperimentalValue-TheoreticalValue|}{TheoreticalValue}$ 。

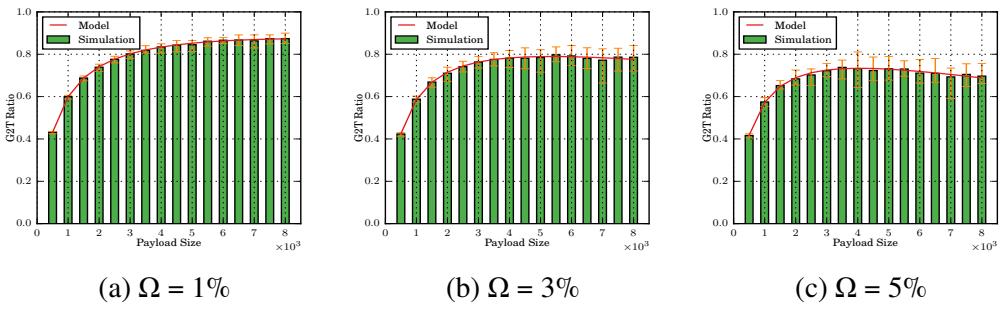


图 4.11 单跳获取数据（路径 $<P_1 - C>$, $<R_1 - C>$, $<R_2 - C>$ ）：丢帧率分别为 1%, 3%, 5%。

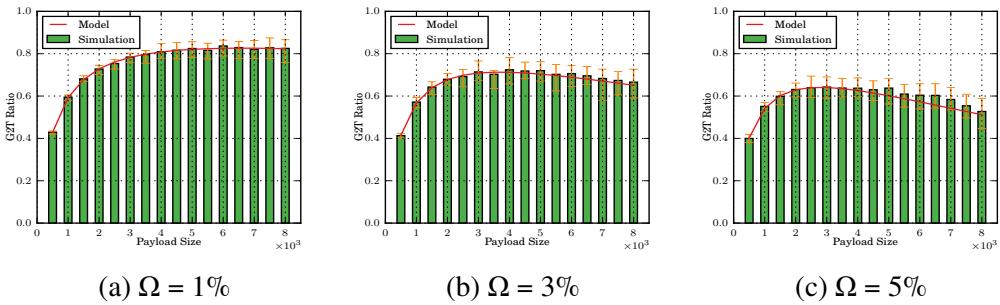


图 4.12 多跳获取数据（路径 $<P_1 - R_1 - C>$, $<P_2 - R_2 - C>$ ）：链路丢帧率分别为 1%, 3%, 5%。

4.6 建模总结

NDN 把通信的焦点从 where 转移到 what，是一个迥异于 IP 的网络体系结构设计，它支持多路径、多数据源、网络层缓存，因此数据获取时，consumer 没有确定的通信对端，使得传统的 IP end-to-end 服务模型及对应的传输效率模型不再适用。与此同时，信息为中心设计要求 NDN 的数据包具有幂等性，为了达到这个目的，数据包需要包含一系列的能够实现网络层操作，如数据匹配、缓存、安全，所需要的信息，这远比 TCP/IP 包头更为复杂，并因此导致了网络传输中不可忽略的开销。本章从这两点出发，建立了 NDN 网络通信效率的模型，并提出具体的机制，用该模型来优化网络传输的效率。本章的结论如下：

- 由于以信息为中心的要求，在 NDN 中，单个数据包开销将达成几百个字节而不可以为网络传输所忽略，这是在 TCP/IP 中不曾出现过的新问题。在这样的情况下，本章采用 G2T (Goodput-to-Throughput) 这个指标来衡量网络传输的效率；相比于载荷比，G2T 反映了总体吞吐量中有效数据的比例。
- 本章建立的模型准确的描述了 NDN 网络通信的效率，实验值与理论值的误差仅 0.9%。根据该模型设计的 ACM 机制，能切实提高应用的效率，实验中比常用的载荷尺寸提高了 12%。本章的模型和最优的包尺寸被翔实的模拟和实验证明。

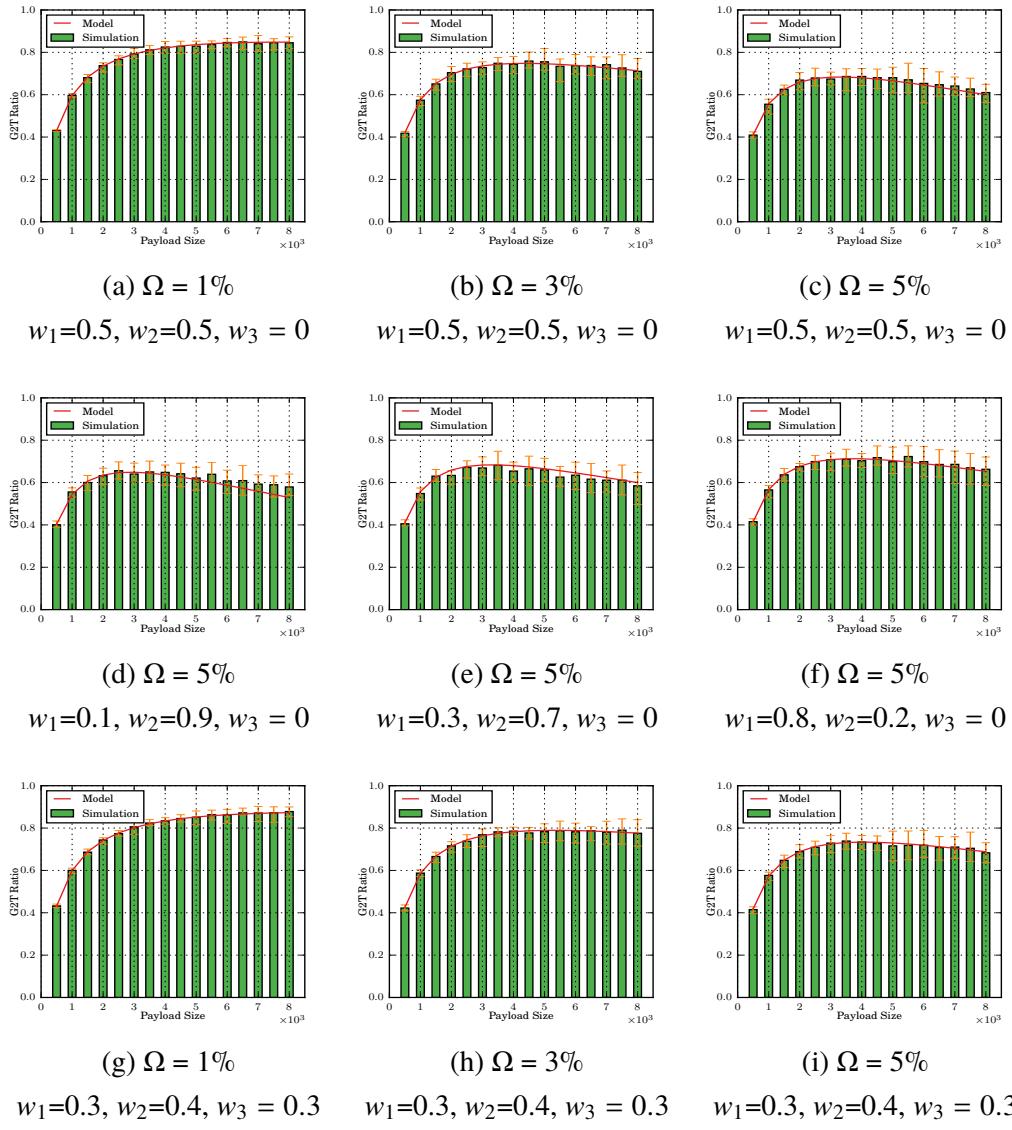


图 4.13 多路径获取数据： w_1 指代路径 $\langle P_1 - R_1 - C \rangle$ 上的流量比例， w_2 指代路径 $\langle P_1 - C \rangle$ 上的流量比例， w_3 指代路径 $\langle P_2 - R_2 - C \rangle$ 上的流量比例。(a) - (f) 是针对同一个数据源的多路径情况；(g) - (i) 是针对多个数据源的多路径情况；(d) - (f) 是针对不同流量比例情况。

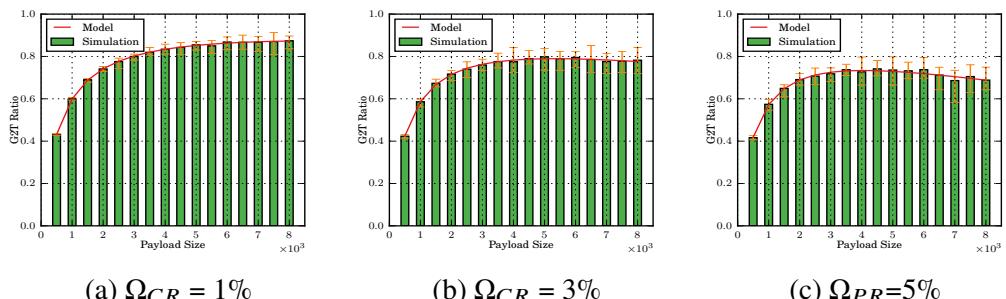


图 4.14 通过异构链路构成的路径获取数据（路径 $\langle P_1 - R_1 - C \rangle, \langle P_2 - R_2 - C \rangle$ ）

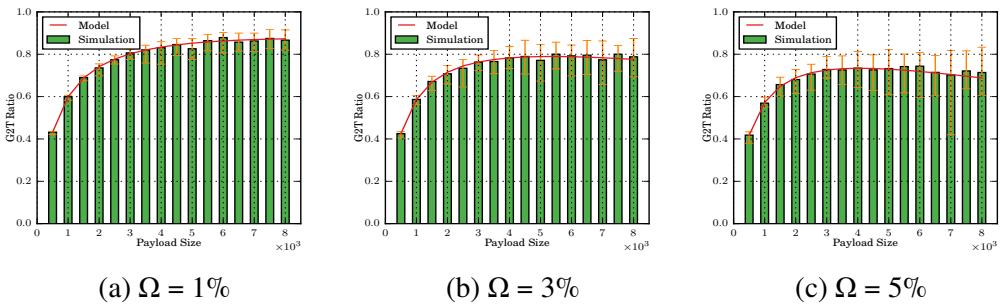


图 4.15 丢帧率符合 Gilbert Loss Model

本章的模型同样适用于其它 ICN 网络体系结构，信息为中心的设计导致的每个包中较大的开销，都可以通过本章的模型予以优化。对于非 ICN 的应用，在涉及到对数据块进行切分（无论是应用层、网络层还是链路层），并且每个切分有较大开销的场景时，也可以使用本章模型予以优化。

最后，本章工作主要以数据包尺寸、链路丢包率作为主要影响传输效率的因素，而在实际 NDN 网络中，路由器缓存大小、Interest 包超时时间、每个路由器转发策略都是影响传输效率的因素，本章将这些因素的总体影响通过链路丢帧率来做宏观的描述，而没有进一步地分析每种因素的具体影响（本章包含链路丢帧率在时间维度的具体影响模型）；更精致准确的模型依赖于更进一步、更深入的研究。

第 5 章 nCDN: NDN 演进部署方案

如第 1.3.3 章所述, NDN 是一个 clean-slate 的设计, 尽管它在体系结构上融合了很多当前非常精巧的设计, 满足当前互联网数据分发的需求, 但是它并不兼容当前 IP 网络, 也不支持数量巨大、广泛应用的各种 IP 网络应用。不兼容性在 TCP/IP 与 NDN 之间形成巨大的鸿沟: 1) 绝大部分资源都在 IP 网络上, NDN 网络中没有数据资源, 不能激励终端用户安装和使用 NDN; 2) 传统的 TCP/IP 的应用也无法利用 NDN 在数据分发方面带来的优势, 不能激励 CSP 部署使用 NDN。实际上, 即便 NDN 能够在未来吸引大量的终端用户和 CSP, IP 可能也将在较长时间内存在, 两者的互通互联也将成为一个潜在的需求。

CDN 是当前网络上最重要的内容分发平台, 大量的资源存储在 CDN 上, 但由于 CDN 与 TCP/IP 在根本目标不一致, 当前 CDN 有很多问题; 然而 NDN 在很多方面与 CDN 非常相似, 并且有很多的优势。因此, 本章提出以 CDN 作为突破口, 提出 nCDN, 它带来的优势有: 1) 能够将大量的数据资源迁移到 NDN 网络, 潜在的促进了 NDN 的使用与部署; 2) 充分发挥 NDN 的优势, 改善当前 CDN 中的各种弊病; 3) 提供了 NDN 部署的渐进方案, 传统的 TCP/IP 应用可以无缝接入, 新的 NDN 应用可以充分发挥自身的优势。

本章余下部分将按如下结构组织: 首先本章介绍相关背景, 包括 CDN 框架及其存在的问题, HTTP/HTTPS 设计及 CDN 对 HTTP/HTTPS 的加速支持(第 5.1 章); 然后介绍 nCDN 的设计(第 5.2 章); 再讨论相关问题, 包括实时数据流支持、安全与部署激励(第 5.3 章); 之后介绍系统原型、部署与实验(第 5.4 章), 最后总结本章(第 5.5 章)。

5.1 相关背景: CDN & HTTP/HTTPS

5.1.1 CDN

CDN 通过把数据部署在网络中分布广泛且靠近用户的代理服务器上, 来为内容商(CSP)提供诸如访问加速, 弹性扩展以及安全防御等服务。如图 5.1 所示, CDN 把分布于各地、各运营商的代理服务器连接起来, 形成一个 overlay 网络, 并在其中加入了很多高级的功能, 如网络状态监测, 服务器/链路打分, 网络传输优化, 以及内容到主机的映射等等。除了商业 CDN 服务提供商, 很多 ISP 也部署了 CDN, 减少其与 provider 和 peer 网络之间的流量。本章介绍 CDN 最主要的功能,

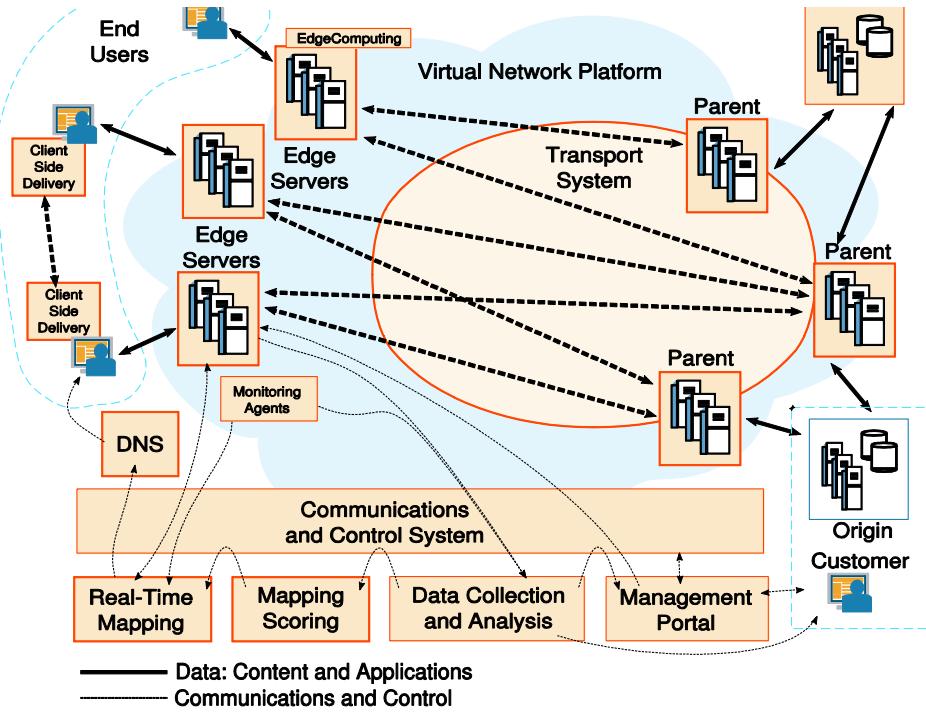


图 5.1 传统 CDN 体系结构(本图摘自^[121]对 Akamai CDN 网络的描述): 其中 Edge Server 是分布在靠近终端用户的代理服务, 网络中的通信和控制系统非常复杂, 依赖于监控系统、映射系统等。

即通过 HTTP/HTTPS 提供数据分发服务。

CDN 中同样需要对数据进行命名, 传统的 CDN 采用了平坦的名字来区分不同的内容, 一些元数据 (metadata), 如数据大小、过期时间等, 则与名字一起来描述对应的内容。在 CDN 中, 存储数据的服务器被称为代理服务器, 可以据此将 CDN 中的通信分为三种: 前端通信, 指代理服务器与终端用户 (consumer) 之间的通信; 后端通信, 指代理服务器与 CSP (源 producer) 之间的通信; CDN 内部通信, 指不同代理服务器之间的通信。

作为基于 TCP/IP 设计的大规模数据分发的 overlay 系统, 从某种程度上说, CDN 具备了直接对命名数据进行操作的设计元素, 如请求路由、数据缓存等; 它与基于信息为中心设计的 NDN 有很多相似点, 表5.1 列举了两者的比较结果。

5.1.1.1 前端通信

在 CDN 中前端通信的主要功能是选择合适的代理服务器来应答终端客户的数据请求。一个数据可能在 CDN 平台上有一个冗余备份, 但由于 IP end-to-end 语义的限制, 只能返回一个代理服务器的 IP 来应答请求, 这种实现数据名到 IP 映射到算法和机制被称为“请求路由”(request routing)。请求路由算法的输入是一系列优

表 5.1 CDN 与 NDN 对比

	CDN	NDN
体系层次	中间件(应用层的实现, 工作在应用层和传输层之间)	网络层
命名	平坦	层次化
请求路由	名字到 IP 映射 (end-to-end)	名字路由 & 带状态转发 (hop-by-hop)
流量探测	分布式网络实时监测	本地对称数据流监测
缓存	应用层	网络层
失败恢复	终端(超时机制)	中间路由器(NACK 机制)

化因素的集合, 包括拓扑位置、丢包率、延时、负载、流量种类等等^[121], 输出是某个代理服务器的 IP 地址。常用的请求路由机制有基于 DNS 的方法^[122] 和 GSLB (Global Server Load Balancing)^[123]. 其中基于 DNS 的方法是目前业界的主流方法, 该方法利用 DNS 中的 CNAME 记录, 把 CSP 的域名链接到 CDN 的域名。例如, 假设一个 CSP 的名字为 “alice.com” 被链接到了 CDN “carol.com” 上, 则该 CSP 的别名就变成 “alice.carol.com”, 当终端客户通过 DNS 解析 “www.alice.com”的时候, 他最终会访问 CDN 的名字服务器, 从而返回 CDN 控制的、由请求路由算法所决定的代理服务器的 IP。

为了能够选择最好的代理服务器, CDN 中需要部署一个分布式的实时网络监测系统来探测网络状态和代理服务器的负载。为了保证选择决策的实时性, DNS 返回给终端的请求路由决策 (IP 地址) 往往带有比较短的 TTL (time-to-live) (例如 5 分钟), 因此这种 DNS 数据在缓存解析器中只能缓存较短时间, 这样的设计有违 DNS 系统利用缓存应答大量请求、抵御链路/节点失效从而提高可用性的设计原则, 使得 CDN 的名字服务器成为整个系统的关键部件, 容易造成单点故障, 因此需要采用备份服务器等手段来保障可靠性。

前端通信的安全性令人担忧。考虑到 CDN 天然的“中间人”(Man-in-the-Middle, MITM) 身份, 当前端通信有安全验证时, 如采用 HTTPS 时, CDN 往往需要采用一些巧妙的方法来“欺骗”终端用户, 使他们相信自己是在和所期望的 CSP 通信, 这带来了非常严重的安全隐患 (第 5.1.2 章)。

5.1.1.2 CDN 内部通信

CDN 内部通信机制构建了一个高性能的内容传输网络。由于代理服务器受控于 CDN, 一系列性能优化方法被用来提升内部通信的性能, 如路径优化, 丢包检测, 传输层优化, 应用层组播^[121]。内部通信往往服务于实时数据流, 或者应用于

请求不能被缓存的数据以及缓存缺失 (cache miss) 的场景。

在内部通信中, CDN 数据分发这一目标, 既与 IP end-to-end 的语义不匹配, 也增加了系统的复杂度。例如, 当缓存缺失发生时, 要么需要分布式的数据与地址索引表来找到最近的存储该数据的代理服务器位置^[124], 要么就只能从源 producer 处拉取数据。尤其是当 CDN 发生变化 (如内容部署发生变化、新的节点加入或退出) 的时候, 数据与地址索引表需要更新, 这种更新信息需要散布到整个 CDN 网络上^[125], 极大地增加了系统管理的复杂度。

5.1.1.3 后端通信

后端通信应用于数据部署 (data out-sourcing), 它有两种模式: 1) 把数据从源 producer 预存到代理服务器上, 这种情况下, 一般采用的是由源 producer 主动把数据推送到代理服务器上的“推”模式; 2) 当请求 CDN 上没有预存或缓存的数据时, 要从源 producer 上返回数据, 这种情况下, 往往采用的是由代理服务器从源 producer 或其它代理服务器^①上获取数据的“拉”模式。为了保障后端通信的安全, 一般需要使用 HTTPS 的方式进行通信, 然而在实际环境中, 后端通信的安全威胁非常严重^[48]: 调研的 CDN, 包括 CloudFront, CloudFlare, CDN77, CDN.net, Incapsula 几乎都没有阻断不安全的后端通信。

5.1.1.4 CDN Software Developing Kit (SDK)

为了让终端客户和 CSP 能更好地利用 CDN 上的冗余数据和高性能的 CDN 网络, 一般来说, CDN 都会为 CSP 提供 SDK。这些 SDK 允许 CSP 编写程序管理数据分发、建造自己的 overlay 网络、优化请求路由、自定义安全配置等; 同时 CSP 也可以基于 SDK 开发客户端程序提供给终端用户, 使得终端客户更容易访问 CDN 上的数据, 并通过 CDN 网络连接服务器或者其它终端客户。一些应用, 如实时流媒体分发、在线游戏等大量使用 CDN 提供的 SDK。因此, 通过 SDK 可以优化前端通信, 也可以优化后端通信。

5.1.2 HTTP/HTTPS & CDN 如何支持 HTTP/HTTPS

5.1.2.1 HTTP

HTTP/HTTPS 是 World-Wide Web (WWW) 的基础, CDN 服务主要用于加速 WWW 流量。

^① 严格地讲, 从其代理服务器上拉数据是属于 CDN 内部通信, 一些研究类型的 CDN, 如 Coral^[124] 用分布式哈希表的一个变种实现了这种方式。

HTTP 定义了两种消息，即请求 (HTTP request) 和应答 (HTTP response)。除了 HTTP 方法 (如 GET、POST 等) 外，请求报文中还定义了一系列字段来指定该请求对应的操作对象 (内容)：1) 数据标识，如 host 头，一般用域名和端口号来指代，如 “www.google.com:80”，resource 头，一般指该 host 上对应的内容，如 “/images/logo.png”，还有可选的报文头来定义内容中字节偏移量，如 range 头；2) 用于内容协商的报文头，如 accept-language, accept-encoding，这种报文头，它们往往对应着多个值；3) 其它一些注释信息，如 user-agent 等。

应答报文则包括：1) 状态域，标定在对应内容上执行对应操作的结果；2) 报文头域，包含对内容缓存信息的说明 (cache-control 头)，内容部分说明 (content-language, content-range 头)，以及设置 cookie (set-cookie 头)；3) 报文信息部分，通常是被请求的数据。

通过上面的介绍可以发现，HTTP 是通过请求/应答的方式，提供操作对命名数据的原语，这是典型的信息为中心设计。同时，HTTP 虽然没有定义缓存机制，但定义了数据缓存相关信息，网络中间件 (如 CDN, Web 缓存) 可以根据这些信息来缓存/替换相应的内容。

5.1.2.2 HTTPS

由于 HTTP 和 TCP/IP 都没有定义安全机制，考虑到安全需求，研究者发明了 HTTPS，即先在 TCP 之上通过 TLS/SSL 维护一个安全的通信会话，然后在这样的会话上传输 HTTP 协议，从而保障通信安全。

TLS(Transport Layer Security) 和 SSL(Secure Sockets Layer) 是旨在提供 end-to-end 通信安全的协议。它利用了 X.509 证书和公钥基础设施 (public key infrastructure, PKI)^①。非对称加密的方式被 TLS/SSL 用于认证通信对端，并且协商一个对称性的会话钥匙 (session key)；然后这个会话钥匙被用于该会话中双方的数据认证和加密。这种方法实现了数据机密性 (confidentiality)、完整性 (integrity)、真实性 (authenticity) 和可认证性 (authentication)。

为了支持 TLS/SSL 中的通信对端认证，一般需要把 CA (certificate authority) 的根证书预装在终端客户机上作为信任锚点。现在每个操作系统中都装有几百个 CA 的根证书。任何一个 CA 都可以为任何一个域名颁发证书使它通过验证，这意味着两个不同的 CA，可能分别为域名 D 颁发了证书；而这两个证书中可能是属于不同组织的。假设持有证书的机构一个是域名 D 所有者，另一个则是第三方，则这个第三方具备了劫持域名 D 的能力，因为这两个证书都可以通过验证，这就是在

^① 此处 X.509 系统指的是由 IETF PKIX 工作组标准化的 X.509 公钥基础设施，而不是由 ITU-T 标准化的。

HTTPS 中出现的证书滥用的情况。现实中存在为数不少的域名都受到证书滥用的影响，导致了严重的安全威胁。研究者进一步提出 DANE (DNS-based Authentication of Named Entities)^[126] 来改善这种情况。DANE 允许域名所有者自己决定验证对应域名需要的 CA 或者信任锚点，这样的信息以 TLSA 这种类型的资源记录存在 DNS 中；同时 DANE 依赖 DNSSEC 来保障终端用户安全地从 DNS 中获得 TLSA 记录。

HTTP/HTTPS 定义了请求、应答两种报文，直接对命名数据进行操作；虽然它是应用层协议，但是也显示出很多与 NDN 相似的地方，表5.2 比较了两者的异同。

表 5.2 NDN 与 HTTP/HTTPS 对比

	NDN	HTTP/HTTPS
体系层次	网络层	应用层
命名	层次化	层次化
消息类型	Interest & Data	request & response
交互机制	拉模式	拉模式
安全机制	基于数据的安全	无/基于管道的安全
缓存支持	网络内缓存	外部缓存/无

5.1.2.3 CDN 如何支持 HTTP/HTTPS

CDN 可以很轻易地加速 HTTP 的流量，因为 HTTP 通过域名和相对路径来描述操作数据，其本身是与存储节点无关的；HTTP 甚至提供了数据缓存描述的信息，允许第三方节点缓存数据。

然而，对于增加了主机验证的 HTTPS，CDN 作为中间人的天然角色则与主机验证形成根本性冲突。由于 TLS/SSL 是在主机验证之后，建立安全连接；所以 CDN 必须通过某种机制使得终端客户“相信”自己是在与 CSP 通信，而不是某些中间人。目前有两种常用的工程方法来解决这个问题^[48]：1) 私钥分享 (private key sharing)，这种方法要求 CSP 向 CDN 分享私钥及相关证书，这从根本上违背了密码学设立公私钥的原则；2) 分享证书 (shared certificate)，这是滥发证书的一种“合理”利用，例如某个域名 D 的内容托管在 CDN C 上，让 CA 颁发一个对 C 和 D 都有效的证书，从而通过 TLS/SSL 主机认证。分享证书的方法创建了一个新的证书，因此也就失去了原本的证书（也是真正的证书）中携带的信息，例如对安全提示信息等。更严重的是，这两种方法都使得 CSP 很大程度上失去了对数据托管证书撤销的控制，当 CSP 转到另外的 CDN 服务时，之前的被分享的私钥或分享的证

书都被原来的 CDN 掌握，成为重大的安全隐患。

文^[48]提出一种基于 DANE 的方法来解决 CDN 在支持 HTTPS 中中间人问题，这种方法不“欺骗”终端用户，而是通过 DANE 把数据代理的“事实”告诉终端用户：域名所有者在 TSLA 资源记录中存储 CSP 和 CDN 的证书，这样终端客户就可以同时认证 CSP 和 CDN 的主机，并且识别出两者之间的数据托管关系。用这种方法，就无需分享私钥，或者利用新的含有“谎言”的语义的证书，从而规避众多安全风险；并且，由于 TSLA 记录是由域名所有者（也就是 CSP），所以 CSP 可以独立撤销或修改这种数据托管关系，而无需受制于人。这种方法依赖 DANE 和 DNSSEC 的支持，同时也需要终端客户能够识别这种新的 TSLA 格式。

5.2 nCDN 的设计

5.2.1 nCDN 系统概述: 把 NDN 支撑 CDN

如同传统 CDN 一样，nCDN 是 IP 的 overlay 系统（图5.2）；CDN 在 IP 与冗余数据中间建立了一套中间件，这套中间件可以利用分布式的实时网络监控系统来对不同的路径和数据源“打分”，用一定的算法在不同服务器之间实现负载均衡，通过 DNS 实现名字到选定 IP 的映射。nCDN 则把 NDN 放在冗余数据和 IP 之间作为一个薄层（shim layer）取代了这一套中间系统，用带状态的转发来对链路进行打分，用网络层路径选择取代了请求路由，而用名字路由来取代名字到 IP 映射。为了保证 nCDN 对终端的透明，nCDN 依然无缝地支持传统设备和新的支持 NDN 的设备。

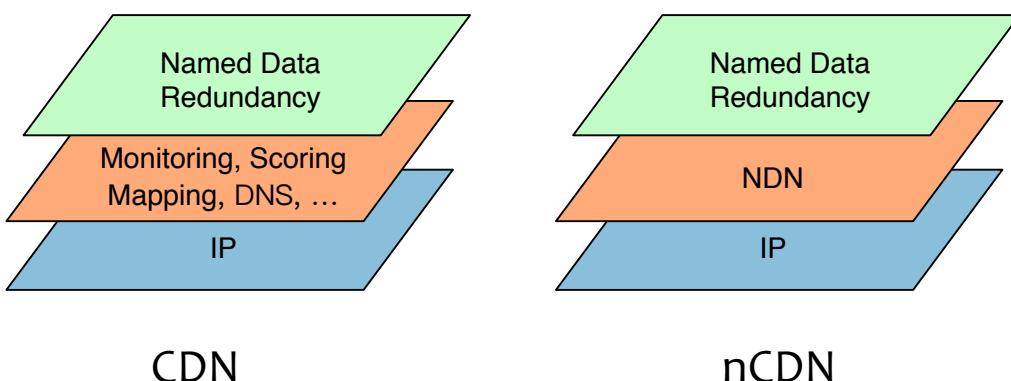


图 5.2 nCDN v.s. CDN : nCDN 用 NDN 取代了 CDN 中复杂的中间件

nCDN 里的代理服务器有双重角色：NDN 路由器和数据服务器。作为 NDN 路由器，每个代理服务器需要运行 NDN 协议栈程序，如 NFD，所有的代理服务器 TCP/UDP 建立链路，并通过路由协议形成了一个 overlay 结构。作为数据服务器，

每个代理服务器运行一个程序来服务各种数据请求，并通过路由协议向 overlay 结构中宣告它能提供的数据的前缀，这样服务器上数据就可以通过 overlay 结构中的任何一个节点访问到。但 nCDN 里还有几个亟须解决的关键问题：

- NDN 的应用稀少，现在最主流的 HTTP/HTTPS 应用是运行在 IP 之上的，传统的 HTTP 服务器（后端）和浏览器（前端）与 NDN 不兼容。
- 安全挑战：CDN 中间人角色不变，依然不能适配 给予管道的安全模型；另外当前 CDN 具体运行中有令人糟糕的工程办法，如公钥分享，不安全的后端通信，CA 证书的滥颁发等问题亟须处理。
- 运行挑战：NDN 网络中还本身还有一些问题，例如路由扩展性，不准确的路由，数据移动性支持等等，也需要解决。

前两个挑战是源于把 IP-based 应用和给予管道的安全模型适应于 NDN 这种以信息为中心的 NDN 设计。本章用 HTTP-NDN Gateway 实现 HTTP 和 NDN 流量相互翻译，HTTP-NDN Gateway 必须是 IP&NDN 双栈的，并且能够理解 HTTP/HTTPS (第 5.2.2 章)。HTTP-NDN Gateway 是 nCDN 关键部件，为了避免单点故障，可以在多个、甚至所有代理服务器上安装 HTTP-NDN Gateway 程序来提高可用性和鲁棒性。为了解决安全问题，本章采用第 5.1.2.3 章介绍的基于 DANE 的方法^[48]，它把 CDN 这种代理的关系存在 DNS 中，而不用“欺骗”终端。同时，nCDN 也提供包含 NDN 开发包的 SDK，鼓励用户终端和 CSP 利用 SDK 开发程序；这样在 NDN 基于数据的安全模型中，中间人角色将不会是安全的障碍，从而彻底地解决用 CDN 做大规模数据分发中面临的安全问题。第三个挑战是 NDN 自身的挑战，本章利用 nDNS 和 LINK 对象来解决一系列网络运行中的挑战，并支持实时视频(第 5.3.1 章)，用网络层 NACK 来解决不准确路由的挑战(第 5.2.3 章)。

图5.3 展示了 nCDN 系统框架，前端和后端通信都利用 HTTP-NDN Gateway 来支持传统 IP 的终端和 CSP 服务器；nCDN 内部网络是纯 NDN 网络。因为 NDN 能够直接通过名字取得内容，所以没有必要在终端选择服务来应答某个请求，而只需要把请求发送到一个代理服务器 S 上，而其它请求路由的功能就通过名字路由来实现，S 被称为访问点 (access point, AP)。对于 NDN 的应用，任何一个代理服务器都是 AP；而对于传统 IP 应用，HTTP-NDN Gateway 是 AP；如果 HTTP-NDN Gateway 在所有代理服务器安装，则所有代理服务器都成为通用的 AP。

为了给应用提供一个 AP，nCDN 依然需要利用 DNS，但与 CDN 里的 DNS 不同，这个 DNS 不是实现名字到 IP 映射，而仅仅是提供一个 AP 的 IP 地址；这样的映射是非常稳定的，它对应的解析结构可以缓存较长时间 (TTL 较大)。

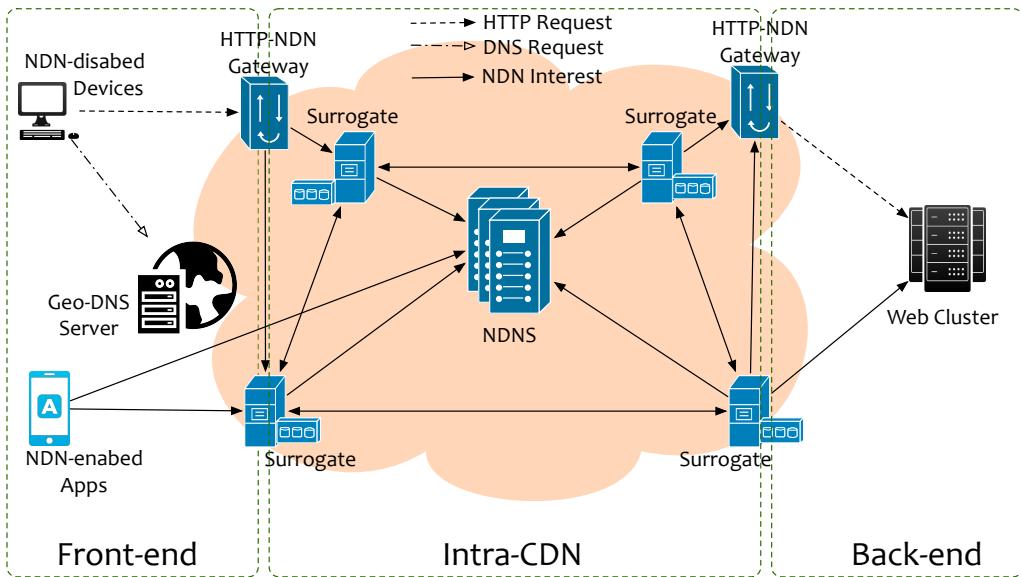


图 5.3 nCDN 框架：也分前端、后端和 nCDN 内部通信，前端通信被极大简化，只需要提供请求接入 nCDN 的入口，而不需要完成请求路由的功能；HTTP-NDN Gateway 是系统中关键部件。

5.2.2 HTTP-NDN Gateway: NDN 与 HTTP 之间的翻译

支持 NDN 终端设备和 CSP 的服务器可以无缝地与代理服务器交互；但是传统的浏览器和网页服务器则需要 HTTP-NDN Gateway 做翻译。

HTTP 和 NDN 的翻译有两种方式：1) bit-to-bit 翻译，2) 语义层次的翻译。bit-to-bit 翻译是把 HTTP 消息 (request & response) 逐比特封装在 NDN 数据包，这种方法简单，而且非常准确地翻译了 HTTP 的语义。但这样一来，Interest 名字中可能含有较多无用的 HTTP request 的包头信息，导致请求的聚合和缓存重利用会受到负面影响；更糟糕的是，这样的名字可能非常大，因为它包含 HTTP request 中所有的比特。语义层次翻译则是定义了 HTTP 在 NDN 协议上的机制，更接近于一个 NDN HTTP 协议。这种方法更加信息为中心。但是这种翻译与 HTTP 语义可能不能完全匹配，例如 HTTP request 可能让一个参数包含多个值，如 accept-language。

在实际中，很多 HTTP 数据，尤其是多媒体数据，往往可以通过较少的信息来描述，这样用语义层次的翻译即可实现，并实现性能优化；但对于一些特殊的 HTTP 请求和数据，用 bit-to-bit 翻译则更加准确。基于这样的观察，本章采取了混合式的翻译方法，一些必须的信息，如 HTTP request 中的 domain, method 和 resource 来鉴别目标数据，同时也允许一些可选域用于 bit-to-bit 的翻译，命名例子如图5.4所示。命名例子中，一个“HTTP”的名字部件放在 domain 与 method 之间，用于路由和多路复用。

在 HTTP-NDN Gateway 具体实现中有两个问题：1) 一些 HTTP response 太大



图 5.4 HTTP-NDN Gateway 翻译命名: 采用混合翻译的方式更好地支持 HTTP 与 NDN 之间的翻译, 兼顾性能优化和准确性。

不能封装到一个 Data 中; 2) 一些 HTTP request 从一个明确的偏移量请求一个大文件一部分。这两个问题都涉及到分段号, HTTP-NDN Gateway 还负责智能地翻译偏移量和分段号, 并且在有必要的时候负责获取多个分段并组装成完整的 HTTP response。

虽然 CDN 主要是用于加速 HTTP 流量 (网页、视频等), 但不排除有其它的应用协议, 在这种情况下, 也需要针对这些应用协议予以翻译。

5.2.3 nCDN 内部通信

本小节介绍 nCDN 如何把 NDN 的优势利用起来实现大规模数据分发。

5.2.3.1 名字路由实现数据可达性

NDN 名字路由天然地支持多路径、多数据源, 因此 nCDN 直接通过名字路由 hop-by-hop 地把请求“引导”到选定的代理服务器; 而不需要像传统 CDN 一样, 依赖于复杂的请求路由算法和机制, end-to-end 地链接终端和代理服务器来应答 HTTP 请求。路由协议让代理服务器和内容的放置变得更加灵活, 无论代理服务器放在哪里, 或者内容放在哪个服务器上都能很方便的接入到网络中; 而且代理服务器和内容的动态变化, 如服务器宕机, 或者内容迁移都可以很方便地处理。

如第 2.2.3 章所指出的, 一个代理服务器在没有前缀 N 下的所有内容情况下, 依然可以宣告 N。因此, 一个请求可能被路由到一个目标数据不存在代理服务器上, nCDN 中将出现这种“不准确路由”的情况。另一方面, 为了应对不准确路由, 代理服务器将返回网络层 NACK, 中间路由器在接到网络层 NACK 之后, 将尝试其它路径和其它数据源。

CSP 的服务器需要宣告更短的前缀, 例如 “/com/youtube”, 这样不能被缓存的数据最后路由到对应的 CSP 服务器上。对于不支持 NDN 的 CSP 服务器, 与它连接的 AP 将代替它负责宣告对应的前缀。

5.2.3.2 自适应转发实现高效负载均衡

虽然 NDN 路由中支持多条路径到冗余数据，但是通过洪泛的方式把请求同时发送到不同路径显然造成带宽浪费。在实际中，一个请求应该根据一些因素，如访问延时、带宽价格等因素选择最佳链路。由于网络状态持续变化，最优的路径应该是被实时的选择。NDN 自适应的转发完全符合了这种需求。

网络层 NACK 在数据获取失败的时候快速回复，造成数据失败的原因可能包括不准确路由、拥塞、链路或主机失效，当一个中间代理服务器 S 不能取得目标数据时，S 向它的上一跳代理服务器 S2 返回一个网络层 NACK，S2 则尝试其它链路，进行快速失败回复；这样的过程可以一直持续，直到目标数据被返回，或者 NACK 被终端收到。同时，这样的机制也修正了 nCDN 中的不准确路由。

与传统 CDN 中分布式实时网络监测系统和 GLSB 实现的流量控制、负载均衡机制相比，nCDN 的设计有非常明显的优势：1) nCDN 利用了 NDN 在网络层 Interest / Data 对本地每一条链路实时探测，相比于全局的分布式网络监控，可以更加准确和及时地应对网络变化；2) nCDN 利用了 NDN 中的 hop-by-hop 流量控制，每一个链路节点能够独立地、根据路由信息、转发策略、链路历史状态和当前最新状态等迅速做出选择，相比与 end-to-end 终端调控方式，控制的链路、路径更加丰富，控制方法更加灵活；3) 与 hop-by-hop 流量控制相关，当出现拥塞、节点或链路失效等情况时，hop-by-hop 的方式允许中间节点迅速的将请求转发到其它备选链路，而无需等终端在请求超时之后，通过 hop-by-hop 的方式进行故障恢复。本文在第 5.4.4 章实验中对 nCDN 和传统基于 GLSB 的 CDN 进行了性能比较，实验结果验证了上述分析。

5.2.4 数据格式

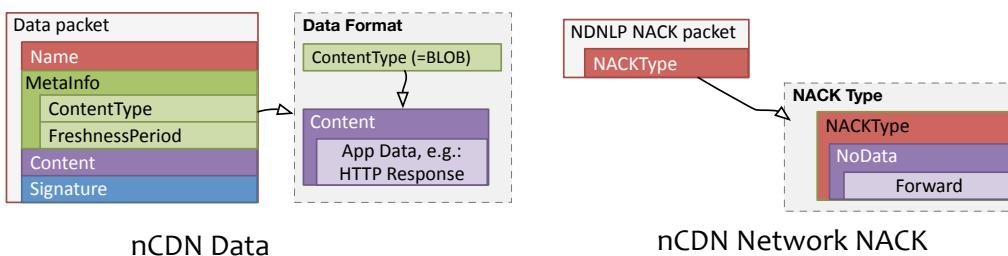


图 5.5 nCDN 数据 TLV 格式：普通数据和网络层 NACK

nCDN 代理服务器之间的通信是纯 NDN 形式，nCDN 运行应用定义自己的应用层数据格式(图5.5)，应用层内容被封装在 Data Content 域里。nCDN 数据包定义

的格式天然地与 NDN 应用兼容，而里面的应用层内容则由应用层来识别和利用，这样可以兼容 IP 的应用，也简化了 NDN 与 HTTP 的翻译。图示这样的 Data 可以由 producer 提前生成，预存到对应的代理服务器上，从而避免私钥分享的问题；如果需要由代理服务器实时生成数据，则用基于 DANE 的方案来解决数据验证的问题。

本节同样定义了网络层 NACK (network NACK)^[89,90]，网络层 NACK 与 nDNS 中应用层 NACK 作用完全不同。应用层 NACK 本质上是一个 Data，是 producer 返回给 consumer 的应答；而网络层 NACK 则具有 hop-by-hop 语义，当一个节点无法取得 Interest I 对应的数据时，它将一个网络层 NACK 回复给 I 的接收端口对应的节点 N，节点 N 接收到该 NACK 后，就迅速启动快速失败恢复机制，从其它链路获取数据；如果 N 其它链路也无法获得数据，则 N 在将网络层 NACK 回复它接收 I 的上一跳。这是一种 hop-by-hop 的流量控制方法。网络层 NACK 概念是为自适应转发设计的^[89]，格式定义在 NDNPv2^[90] 中。在 nCDN，网络层 NACK 和自适应转发修正了不准确路由，提高了数据传输效率，搭建了高效的 nCDN 网络。

5.3 nCDN 相关问题讨论

5.3.1 实时数据流

加速实施数据流是当前网络常见的场景，这种应用可以分为两类，一种是重大的活动直播，如世界杯、奥林匹克运动会等，特点是单一 producer，超大规模 consumer；另一种多人实时数据流，如多人视频、多人在线游戏，特点是多个成员构成一个通信组，每个成员同时是 producer 和 consumer，他们产生的数据需要在通信组里同步。NDN 支持组播，能很好地适配第一种场景，而且 NDN 网络中的快速恢复机制，以及网络层缓存可以非常好地处理针对组播中丢包的情况。针对第二种场景，多源数据同步是 NDN 中典型用例，一些现有的研究，如 ChronoSync^[25], iSync^[105] 解决了这种场景下的需求，成为对 NDN 网络中的重要补充。同时需要说明的是，针对第二种场景，组成员的可能不能直接通过他们的名字来路由，因此需要 SNAMP^[5] 的机制和 nDNS 系统来保证路由可达性。

5.3.2 数据部署和安全性

nCDN 同时允许预存确定的数据集到确定的代理服务器，同时也支持网络层由数据访问局部性决定的缓存。对于数据预存，将数据集部署到不同的代理服务器是 CDN 中常见的需求。在 NDN 名字路由的帮助下，nCDN 可以从最近的内容

存储节点获得内容，如果 nCDN 所有代理服务器上没有对应的内容，nCDN 可以通过名字路由从源 producer 处获得内容。

对于基于 NDN 的通信，用基于数据的安全的方式保障通信的安全，对应的证书存储在 nDNS 中；然而，为了兼容传统的 HTTPS 应用，基于 DANE 的方案被采用，通过 DNS 来表达内容代理授权的含义，而不需“欺骗”终端。在实际中，HTTPS 通信每次会话都需要协商一个会话钥匙用于前面和加密对应的内容，这种操作在 AP 上完成。

5.3.3 部署激励

CSP 一般通过两种方式利用 CDN，一种是名字部件自己的 CDN，例如苹果、微软、谷歌这类大公司，另外一种是购买第三方提供的 CDN 服务。于 CDN 平台需要的存储、带宽相比，CDN 服务非常贵。例如，亚马逊提供的 S3 云服务（包括存储和带宽）价格是亚马逊在它的 CDN 服务平台 CloudFront 上提供相同存储空间和带宽下的 CDN 服务的价格的一半。^①

搭建一个传统的 CDN 并不容易，其中名字到 IP 映射、分布式网络监控系统、数据传输优化等都增加了系统复杂性。相比之下，搭建一个 nCDN 将非常简单，nCDN 将很多主要的功能都转移到 NDN 网络，不需要复杂的配置；数据和节点管理简单灵活；并且具有更好的性能。对于 CSP 来说，nCDN 是一个更好的选择。

5.4 系统原型、部署与实验

本人开发了 nCDN 原型系统，本节简单介绍了 nCDN 原型系统涉及到软件套装，并用程序截图界面来说明系统可用性。由于缺乏足够的实际代理服务器和 CDN trace 数据，所以本节用 ndnSIM 进行更多的模拟实验，来验证设计可用性与性能。

5.4.1 原型与部署

nCDN 被部署在一个有 8 个骨干节点的尝试床上（图 5.6），这些节点分布在不同的城市、不同的 ISP，相互之间通过 IPv4 或 IPv6 的 UDP 链路连接起来，节点上安装了 nCDN 需要的一系列软件套装：

^① 本节是这样计算价格的：假设平均数据对象的大小是 2K 字节，提供亚马逊服务价格计算器 According to Amazon <http://calculator.s3.amazonaws.com/index.html> 的计算：S3 平台上 100T 字节的存储，在终端和云平台、云和源 producer、以及云平台服务器内部各 10T 字节的流量的总价格是 4427.09 美元每月；而在 CloudFront 上，10T 字节的流量（流量分布为 30% 在美国，20% 欧洲，而香港/菲律宾/韩国/新加坡/台湾、日本、南美、澳大利亚、印度各 10%）从 CDN 到终端，加上一个证书验证的支持总价格是 9213.08 美元每月；上述结果是根据 2016 年 4 月 11 日价格计算。

1. 代理服务器上安装 NDN 协议栈实现 NFD^[12], NDN 路由协议 NLSR^[14] 和一个 nCDN 后台程序能够访问本地数据仓库并应答对应请求。
2. HTTP-NDN Gateway 的实现，一个早期版本曾经展示过^[127]。
3. nDNS，名字解析系统，存储 nCDN 中需要的 LINK 对象，证书等信息。
4. Location-aware DNS: Bind and Geodns^[128]，根据 MaxMind's^[129] 提供的地理位置数据，用不同的 DNS response 来应答来自不同地域的相同 DNS request。
5. 一些 NDN 程序，包括一个在线视频播放器 nPlayer 和一个网站，能够通过安装了 NDN 插件 NDN.js 的浏览器访问。
6. 一个 nCDN 上的资源管理系统 RMS (resource management system)^[130]。

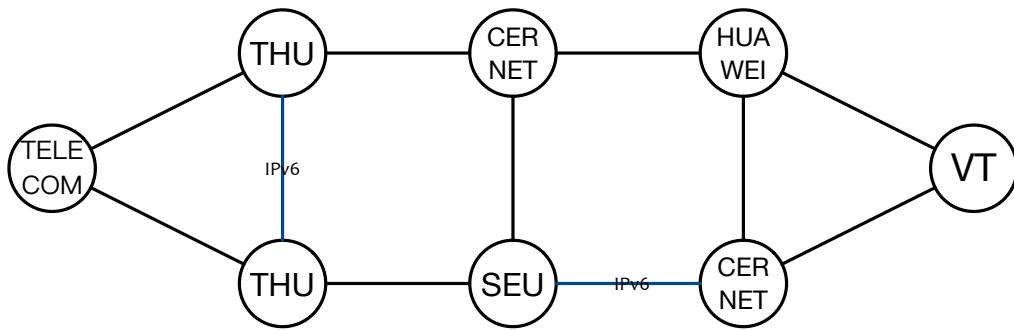


图 5.6 nCDN 部署测试床：该测试床包含不同城市、ISP 的节点，既有 IPv4 链路也有 IPv6 链路。

上述的 nCDN 平台支持浏览器（包括桌面端和手机端），例如 Internet Explorer, Chrome, Safari, and Firefox，同时也能加速视频播放。对应的客户端既可以是传统 IP 应用，也可以是支持 NDN 的应用。

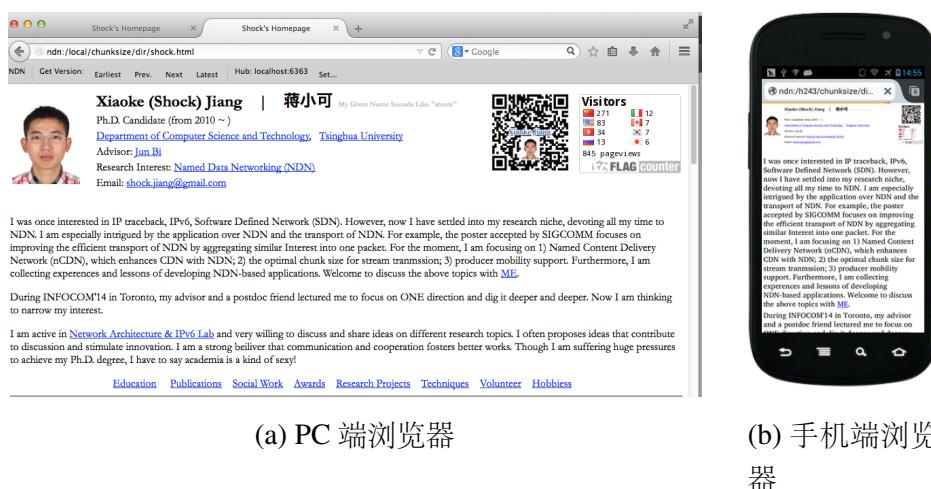
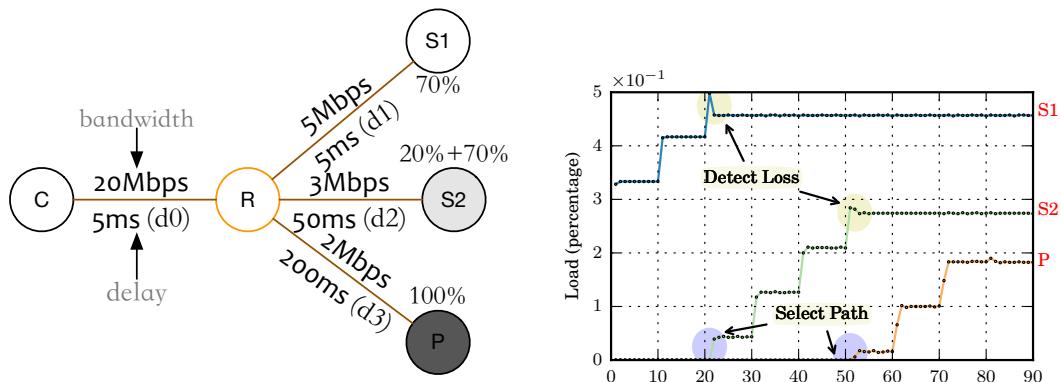


图 5.7 nCDN 网页，同时也可以加速视频应用，如 nPlayer。

图5.7(a)和图5.7(b)展示了两个应用的界面，另外图4.7展示的NDN视频播放器nPlayer也可以通过nCDN测试床来加速。在本章搭建的平台中，nCDN能有效加速数据获取，最好的情况下，访问延时最优情况下降低了99.5%，此时数据通过缓存直接满足时。这个结论对于传统IP应用和NDN应用都适用。

5.4.2 路径选择

本节首先验证网络路径选择和快速失败重传的优势，本节依然用一个简单的拓扑(图5.8(a))来进行实行，这个拓扑与第3.4.3章里的图3.11(a)相似，但节点含义不同，后续实验中，部署的内容也不同。从consumer C的角度看，有2个代理服务器S1&S2和一个CSP producer P，该图中还显示了对应链路的带宽和传输时延。S1与consumer在同一网络中，S2在邻居AS中，而P离consumer比较远。为了避免缓存给统计结果带来的干扰，实验中每次发送不同的请求。发送请求的频率每10秒钟增加一次。路由器R上默认的转发策略是best route。



(a) 简单拓扑：与图3.11(a)相似，但意义不同
(b) 主机/链路选择：与图3.11(b)意义不同

图5.8 功能性验证：nCDN与nDNS一样，都充分利用了NDN网络层主机/链路选择，不赘述其它优点。

如图5.8(b)所示，S1永远是第一选择(0-20秒)，S2是第二选择(20-50秒)，只有当流量压力超出正在使用的链路带宽之后，转发平面检测到丢包，并且转发一部分请求到下一个最优的链路(第20, 50秒)。这个结论与第3.4.3章的结论是一样的，其它相似实验结果(如缓存等)不再赘述。

5.4.3 不准确路由

针对图5.8(a)里的拓扑，实验中改动其中内容分布的情况来验证不准确路由是如何被NACK修正的。假设S1, S2和P宣告了同样的一个前缀，但是S1和S2作

为代理服务器并不存储该前缀下的所有内容，其中 S1 存储了其中的 70% 内容，S2 存储了这 70% 以及另外的 20% 内容。P 作为 producer，存储有所有的内容。由于 S1 是最优先的链路，只要链路允许，70% 的请求应该从 S1 获得内容；同样的 20% 的内容从 S2 中获得；而 P 服务了 10% 的内容。

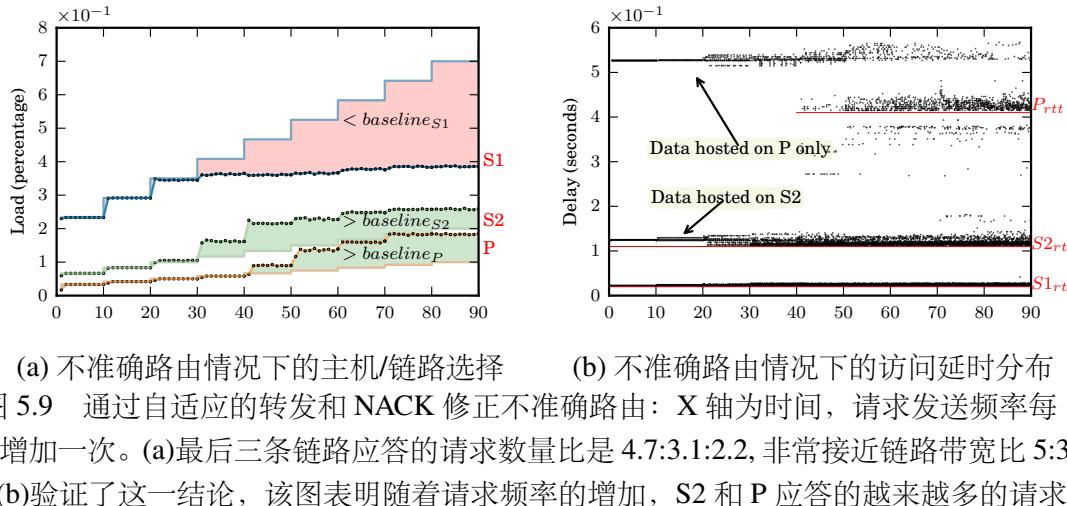
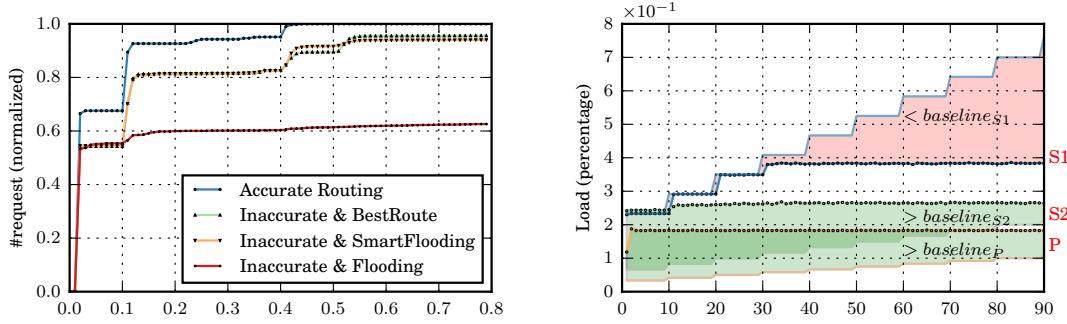


图 5.9 通过自适应的转发和 NACK 修正不准确路由：X 轴为时间，请求发送频率每 10 秒增加一次。(a) 最后三条链路应答的请求数量比是 4.7:3.1:2.2，非常接近链路带宽比 5:3:2。
(b) 验证了这一结论，该图表明随着请求频率的增加，S2 和 P 应答的越来越多的请求。

图 5.9(a) 显示了流量按内容比例服务的基准线 (70%, 20%, 10%)，和在实验中观测到的真实的不同主机应答的请求。绿色的区域是超出基准线的部分，而红色的区域是少于基准线的部分。从图中可以观察到，在一开始流量压力比较轻的时候，实验数据与基准线符合得很好 (0-30 秒)，表明请求是被最合适的服务器应答。然而当链路 R-S1 被完全占用之后 (第 30 秒后)，S1 应答的请求数少于基准线，而 S2 应答的数量开始高于基准线；而在 R-S2 被完全占用之后 (第 40 秒后)，P 应答的请求开始多于基准线。在这个阶段，自适应的转发平面会把请求转移到其它可选链路，所以，不准确的路由通过中间路由器上的快速失败恢复得到修正，即便是当流量压力较大的情况下，依然可以工作。到最后由 S1, S2 和 P 应答的请求比例是 4.7:3.1:2.2，这与三个关键链路 (R-S1, R-S2, R-P) 的带宽比 5:3:2 非常接近，说明自适应的转发能够充分利用带宽。图 5.9(b) 展示了在这种情况下的访问延时，从一开始 S2, P 就开始应答请求，随着请求频率的增加，S2, P 应答了越来越多的请求。

本节进一步测试了不同的转发策略对不准确路由的影响。实验中尝试了最优路由 (best route), 洪泛 (flooding) 和智能洪泛 (smart flooding)^①，对于这些不准确路由的情况，它们的性能上限是准确路由的性能。图 5.10(a) 展示了不同转发策略之下请求数量 (归一化之后) 在传输延时维度上的分布；需要说明的是，只有从数

^① 当一个链路返回 NACK 时，将对应的 Interest 洪泛到其它所有可能的链路。



(a) 不同的转发策略在不准确路由延时
 (b) 洪泛转发策略下不同服务器的负载
 图 5.10 不同转发策略的影响: (a) 可见不准确路由下合适的转发策略 (best route & smart flooding) 都几乎能取得性能 (服务能力) 和延时 (准确路由情况下的值) 上限, 比洪泛策略好很多; (b) 解析了其中原因, 洪泛策略一开始就占满 R-S2 & R-P, 智能依靠最大链路 R-S1 获取更多的数据, 并不能充分利用冗余链路。

据源获得数据请求才被统计。准确路由要比不准确路由服务能力更强, 因为更多的请求获得数据; 在洪泛策略下, 只有 63% 的请求返回数据, 而最佳路由和智能洪泛都几乎达到了性能上限 (96% & 94%)。对于准确路由的情况, 将近 70% 的请求是由 S1 来应答的, 少于 5% 是由 P 应答的; 而在最优路由和智能洪泛策略下, 有 60% 的请求是由 S1 应答, 14% 由 P 应答。图 5.10(a) 说明, 即便不为 nCDN 定制转发策略, 仅仅依靠比较常用的策略, 如最优路由, 智能洪泛, 服务能力和延时也能达到将近上限值 (准确路由情况下的值); 网络修正不准确路由的代价非常小, 是可以接受的。洪泛策略之所以达不到好的效果, 是因为洪泛浪费了大量的带宽, 图 5.10(b) 验证了这一结果, 在洪泛转发策略下, 连接 S2 和 P 的链路 (R-S2, R-P) 在一开始就被占满, 请求者只有依靠 S1 获取更多的数据; 而当连接 S1 的链路 (R-S1) 也被占满之后, 即便增加请求发送速率, 也不能获取更多数据。

5.4.4 与 CDN 性能比较

本章节对 nCDN 和传统 CDN 的性能作比较, 该比较包括丢包数量, 服务能力、访问延时、处理链路或主机失效这些关键性指标。本节依然采用 nDNS 实验 (第 3.4.2 章) 中的 Rocketfuel AT&T PoP 拓扑^[111], 实验中用到的数据是根据全球 CDN 领导者公司 Akamai 的实际 trace 数据得到的分布模型^[65]。实验中请求发送速率在 40 到 200 个每秒, 200 个每秒对于当前拓扑设定而言, 是较重的流量压力。

传统 CDN 拥有 GSLB, 它能决定最近的通信对端来服务请求。如果一个代理服务器没有返回对应的数据, GSLB 将选择另外一个代理服务器来服务。

图 5.11(a) 展示了在不同请求发送速率下在 consumer 端检测到的请求丢失情况。在这一点而言, nCDN 比较比 CDN 要好很多, 这是因为路由器上的快速失败

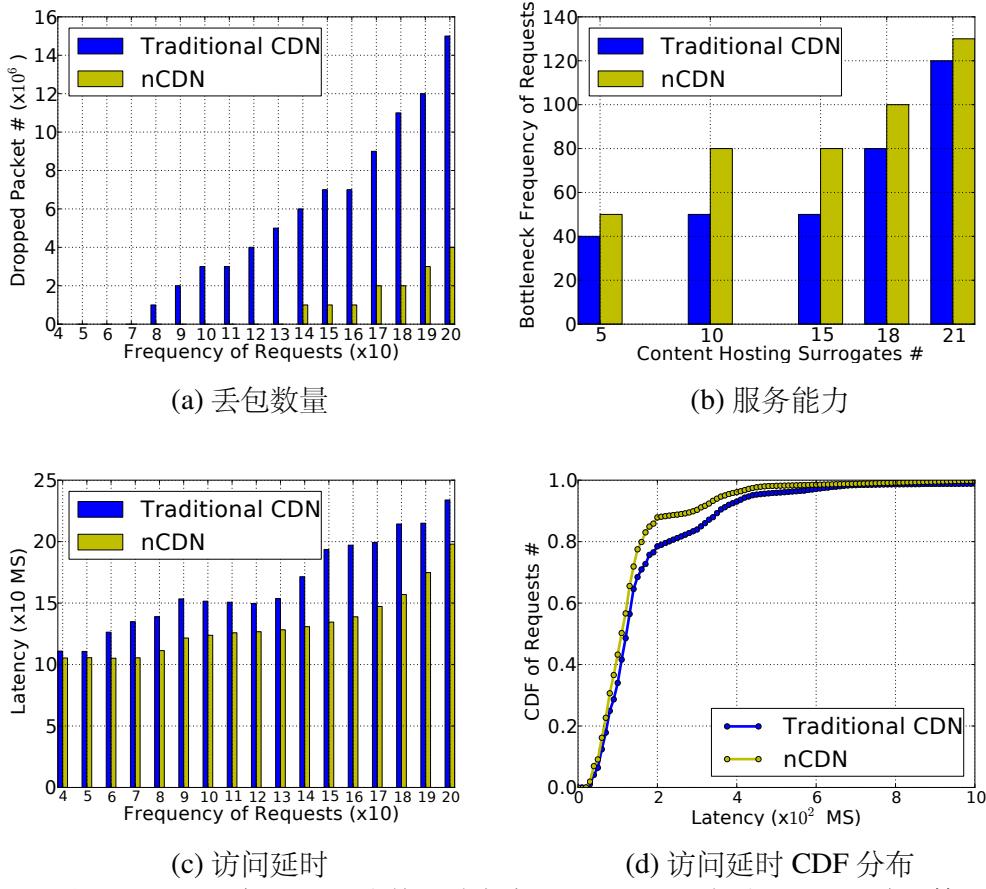


图 5.11 CDN 与 nCDN 比较：从各方面看，nCDN 都比 CDN 更胜一筹。

回复能够处理丢包，而无需终端来处理。图5.11(b)展示了 nCDN 和 CDN 服务能力^①的对比，而 nCDN 的服务能力略高一筹。这两个之所以差距不大，是因为服务能力受到很多其它因素的限制，如关键链路的带宽。而且，模拟中用到的 GSLB 可以准确和实时地调整流量负载，然而实际中用到的 GSLB 还面临着调控延时等问题，所以模拟中的 CDN 效果是真实 CDN 的上限。

图5.11(c) 和 图5.11(d) 测量了数据访问延时。图5.11(c) 显示了不同负载状态下的平均延时，其中 nCDN 的延时显著低于 CDN 的延时。图5.11(d) 则显示了在请求发送速率为 100 个每秒时的延时的累积分布情况，在 nCDN 情况下，延时低于 200ms 的请求比例高达 90%，而在 CDN 情况下，这个值下降到 80%。如图中所示，总体而言，相比与传统 CDN，nCDN 将服务用户数量的瓶颈提高 60%（网络流量非过载的情况下），平均延时缩短 20%；是实际应用中，利用 nCDN 加速可以将访问延时减少 99.5%。

图5.12 衡量了当发生链路、主机失效情况下，请求丢失的情况，面对链路和

^① 服务能力通过请求发送频率的瓶颈来衡量，测量方法是逐渐增大请求发送频率，直到请求丢失的频率达到 10 个每秒时，这个时候的请求发送频率定义为服务能力。

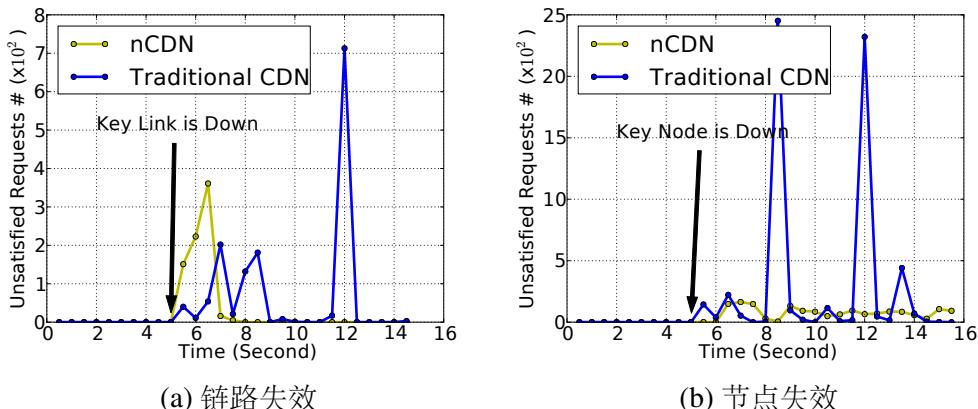


图 5.12 CDN 与 nCDN 处理链路、主机失效：面对链路和节点时效，nCDN 能够利用 NDN 很快及时恢复，造成的影响有限（凸起很小）；然而在 CDN 中，这种失效需要较长时间才能反应过来，造成了较多请求丢失。

节点时效，nCDN 能够利用 NDN 及时恢复，造成的影响有限（凸起很小）；然而在 CDN 中，这种失效需要较长时间才能反应过来，造成了较多请求丢失。结合图5.11和图5.12，可以总结出 nCDN 充分发挥了 NDN 的优势，在各个方面表现都比基于 IP 的 CDN 要好。

5.5 nCDN 总结

本文工作有两方面重要意义，一方面是从 CDN 的角度看，本章提出了一个设计更简单，性能更高，服务更安全的 CDN 框架；另一方面本章把 NDN 应用于当前网络基础设施，提供了一个 NDN 渐进部署的方案，潜在地还向 NDN 网路中引入了大量的资源。nCDN 的特点总结如下：

- 传统的 CDN 设计中的重点难点是把 TCP/IP 这种以连接两个节点为目的的设计适配于通过内容冗余来达到大规模数据分发的目的，这种适配通过全局网络内容的实时监控，DNS 动态映射，以及私钥/证书共享等方法得以实现，但留下了很多的问题，例如 end-to-end 的流量均衡的模式不能有效利用所有的冗余链路和数据，全局网络监测带来的系统复杂性和实时性问题，私钥分享的 hacking 方法带来潜在的安全漏洞。而 nCDN 则充分利用 NDN 名字转发，hop-by-hop 链路/主机选择，快速失败回复，基于数据的安全，天然的缓存支持等信息为中心的设计，既简化了系统设计，又充分利用了资源（数据资源，带宽资源等）提高了数据分发效率，充分体现了 NDN 在数据分发方面的优势。经过本文的实验验证，相比于传统 CDN 中对数据 end-to-end 的控制，nCDN 中继承了 NDN hop-by-hop 的控制，在 QoS 各个方面都取得了更好的效果。

- 数据获取的过程极大地改变了。传统 CDN 依赖前端通信来建立终端网络用户与选定的代理主机通信；然而在 nCDN 中，前端通信仅仅提供了一个接入点，而不需要符合的网络监测和负载均衡的机制；nCDN 消除了 name to IP 映射造成的瓶颈；而 nCDN 内部通信成为数据发现和获取的关键。
- 数据安全得到改善，对于 nCDN 内部通信完全依赖 NDN 固有的基于数据的安全来保证安全性，而对于传统设备与接入点的通信，nCDN 利用 DANE 机制将数据分发服务代理的语义放入对应的 DNS 记录中，从而避免了私钥分享等问题。

第6章 论文总结

当前互联网承担的角色和它面临的挑战已经超出了半个世纪前的互联网先驱们的想象。它打破了四维时空中距离的限制，使整个人类社会变得“平坦”，因此互联网也被形容为人类社会的“第五维”。面对第五维上无所不在的数据分发的需求，研究者提出很多 patch 或 overlay 方案来改造 IP；而 NDN 则从另外一个角度出发，先忽略与 IP 的兼容性，直指最终目标，提出了信息为中心的颠覆式解决方案。NDN 将很多已有设计元素通过操作命名数据数据创造性地组合起来，从而打破 end-to-end 管道通信的限制，能充分利用冗余数据和链路来加快数据分发速度，增加数据分发能力。

NDN 的目标符合互联网数据分发的潮流，它的设计反映了互联网技术发展的趋势；很欣喜地看到一些 NDN 相关的应用和网络设备逐渐出现，如建筑管理系统 BMS^[20]，大型强子对装机和气候检测系统中数据收集系统^[21,22]，以及思科公司在 2016 年世界移动通信大会上展示的兼容 ICN/NDN 的 5G 设备^[131]。但从目前 NDN 发展来看，将它用于实际网络还面临着一系列的挑战，本文针对网络运行问题、模型优化问题和兼容部署问题，分别提出 nDNS、ACM 和 nCDN 尝试解决，并取得了预期的效果。本文研究贡献总结如下：

- 本文设计、实现和部署了 NDN 的名字解析系统 nDNS，用以解决一些 NDN 网络中目前已经发现的和潜在地挑战。nDNS 自举地解决了系统本身的路由可达性，提供了灵活高效的数据验证机制，能向其它应用提高路由和数据验证所需要的信息，从而引导它们开始通信；nDNS 已实际部署于 NDN Testbed，成为 NDN 网络中的基础设施服务。在此过程中，本文还积累了把传统 IP 应用迁移到 NDN 网络上的经验，有助于未来更多应用的迁移。
- 本文首次（尽本人所知）指出信息为中心设计导致每个数据包中的开销高达几百个字节，而不能在传输中忽略；有鉴于此，本文建立了 NDN 网络通信效率的模型来优化通信，该模型非常准确，实验与理论误差仅 0.9%；并且本文提出 ACM 机制把该模型应用于实际，在本文实验中 ACM 机制把通信效率提高了 12%。
- 本文首次（尽本人所知）将 NDN 和 CDN 结合起来，提出、设计、实现和部署了 nCDN，用 NDN 来改造当前 CDN。nCDN 既简化了 CDN 的设计，提高了 CDN 的性能，改善了 CDN 的安全；又潜在的把 CDN 中数据引入 NDN 里，增强了 NDN 的部署激励，并支持 NDN 的渐进部署。本文的 nCDN 平

台可以加速传统 IP 和 NDN 应用的网页、视频的内容获取；实验结果显示，相比与传统 CDN，nCDN 将服务用户数量的瓶颈提高 60%（网络流量非过载的情况下），平均延时缩短 20%；在实际应用中，利用 nCDN 加速可以将访问延时减少 99.5%。

本文中的 nDNS，ACM 和 nCDN 三个工作在不同层次发挥作用：nDNS 首先解决了自身的功能问题，这使得大规模 NDN 网络得以运行；ACM 则解决了性能问题，通过建立网络通信的数学模型，来实现性能优化；而 nCDN 则是应用的问题，激励终端用户和 CSP 使用 NDN，允许 NDN 渐进部署。在实际方案部署和使用中，三者也可以互相协作。总体上说，本文通过 nDNS、ACM 和 nCDN 提供了 NDN 应用于实际内容分发的方案，成为 NDN 体系结构的重要补充。

相应地，本文在三个方案具体设计中充分地发挥了 NDN 的优势，例如用网络层缓存、hop-by-hop 网络主机/链路选择、基于数据的安全等，NDN 中固有的支持，来简化应用的设计和实现，这样设计者只需要关注应用层面的语义和功能，而把数据获取的效率、规模和安全等传统应用中的设计难点交由 NDN 来处理。

更进一步，作为“可用性”问题研究，本文不光从理论的高度上总结了可用性的挑战背后的原因以及相应的对策，并且从工程上实现和部署了本文提出的解决方案，提供了一系列可运行软件和系统，如名字解析系统 nDNS、ACM 库 ndnflow，在线视频播放器 nPlayer、CDN 资源管理系统 RMS 等，让相关挑战切实解决，让相应方案落地“可用”。

真正使 NDN 普遍应用（仅技术一方面而言）有非常多的挑战，本文较好地解决了其中一部分关键的问题（功能、性能、兼容性），立体式地提供了 NDN 应用于实际内容分发的方案，成为 NDN 体系结构的重要补充；其它的挑战，例如其它潜在的网络运行问题，网络通信中延时的学术模型，及更多的因素对通信效率的影响等，有待于更多、更深入的研究。

最后，虽然 NDN 总体研究取得了非常好的进展，但 NDN 最后能否取代 IP 尚未可知。不管结果如何，可以预见的是，这种以信息为中心的设计思想是一个火种，它开辟了一种全新的，迥异与 TCP/IP 的视角来看待网络通信，开拓出网络体系研究的新视野，具有非常强的思想价值和研究意义；即便它最后不能部署在网络层，它的基本思想也极有可能被很多应用层、或者网络中间件吸收，从而促进整个互联网的变革。

参考文献

- [1] Cisco visual networking index: Forecast and methodology, 2014–2019. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white-paper_c11-481360.pdf.
- [2] InterDigital I. Advanced networks. <http://www.interdigital.com/download/551aec840de2c0b78b00001c>, 2015.
- [3] News R. Information centric networking (icn): key to 5g, qoe. <http://www.rcrwireless.com/20151202/carriers/information-centric-network-key-to-5g-qoe-tag17>.
- [4] Postel J, et al. RFC 791: Internet Protocol. 1981..
- [5] Afanasyev A, Yi C, Wang L, et al. Snamp: Secure namespace mapping to scale ndn forwarding. 18th IEEE Global Internet Symposium, 2015.
- [6] Zhu Z, Afanasyev A, Zhang L. A new perspective on mobility support. <http://named-data.net/wp-content/uploads/TRmobility.pdf>, 2013.
- [7] Zhang Y, Zhang H, Zhang L. Kite: A mobility support scheme for ndn. Proceedings of the 1st international conference on Information-centric networking. ACM, 2014. 179–180.
- [8] NDN Testbed. <http://named-data.net/ndn-testbed/>.
- [9] Jacobson V, Smetters D, Thornton J, et al. Networking named content. Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies. ACM, 2009.
- [10] Concepts that helped lead to the ndn design. http://named-data.net/publications/education/#Reading_List_of_NDN_Publications.
- [11] NDN Homepage. <http://named-data.net/>.
- [12] NFD: NDN Forwarding Daemon. <http://named-data.net/doc/NFD/current/>.
- [13] NDNx: Software Router of NDN. <https://github.com/named-data/ndnx/blob/master/README>.
- [14] Hoque A, Amin S O, Alyyan A, et al. Nlsr: named-data link state routing protocol. Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking. ACM, 2013. 15–20.
- [15] ndn-cxx Project. <http://named-data.net/doc/ndn-cxx/current/>.
- [16] Moiseenko I, Zhang L. Consumer-producer api for named data networking. <http://named-data.net/wp-content/uploads/2014/02/TR17-consumer-producer-API.pdf>.
- [17] Pyndn. <https://github.com/cawka/PyNDN>.
- [18] Shang W, Thompson J, Cherkaoui M, et al. Ndn.js: A javascript client library for named data networking..
- [19] jndn: A named data networking client library for java. <https://github.com/named-data/jndn>.
- [20] Shang W, Ding Q, Marianantoni A, et al. Securing building management systems using named data networking. Network, IEEE, 2014, 28(3):50–56.

- [21] Olschanowsky C, Shannigrahi S, Papadopoulos C. Supporting climate research using named data networking. Local & Metropolitan Area Networks (LANMAN), 2014 IEEE 20th International Workshop on. IEEE, 2014. 1–6.
- [22] Jacobson V, Burke J, Zhang L, et al. Named data networking next phase (ndn-np) project may 2014-april 2015 annual report..
- [23] Burke J, Gasti P, Nathan N, et al. Securing instrumented environments over content-centric networking: the case of lighting control. arXiv preprint arXiv:1208.1336, 2012..
- [24] Zhu Z, Wang S, Yang X, et al. Act: audio conference tool over named data networking. ACM ICN, 2011, 11.
- [25] Zhu Z, Afanasyev A. Let's chronosync: Decentralized dataset state synchronization in named data networking. Proceedings of the 21st IEEE International Conference on Network Protocols (ICNP 2013), 2013.
- [26] Wang L, Wakikawa R, Kuntz R, et al. Data naming in vehicle-to-vehicle communications. Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on. IEEE, 2012. 328–333.
- [27] Internet Corporation For Assigned Names and Numbers. ICANN. [Http://www.icann.org](http://www.icann.org).
- [28] Kutscher D, Eum S, Pentikousis K, et al. ICN Research Challenges. Internet-Draft draft-irtf-icnrg-challenges-06, Internet Engineering Task Force, March, 2016. <https://tools.ietf.org/html/draft-irtf-icnrg-challenges-06>. Work in Progress.
- [29] Narayanan A, Oran D. Ndn and ip routing can it scale. Proposed Information-Centric Networking Research Group (ICNRG), Side meeting at IETF-82, Taipei, 2011.
- [30] Zhu Z, Afanasyev A, Zhang L. A new perspective on mobility support. Named-Data Networking Project, Tech. Rep, 2013..
- [31] Mockapetris P. Rfc 1035: Domain names, implementation and specification, november 1987. <http://www.ietf.org/rfc/rfc1035.txt>, 1987..
- [32] Mockapetris P. Rfc 1034: Domain names: concepts and facilities. Status: Standard, 2003..
- [33] Arends R, Austein R, Larson M, et al. Resource records for the dns security extensions. Technical report, RFC 4034, March, 2005.
- [34] Arends R, Austein R, Larson M, et al. Protocol modifications for the dns security extensions. Technical report, RFC 4035, March, 2005.
- [35] Arends R, Austein R, Larson M, et al. Dns security introduction and requirements. Technical report, RFC 4033, March, 2005.
- [36] Wang L, Moiseenko I, Zhang L. Ndnlive and ndntube: Live and prerecorded video streaming over ndn. Technical report, NDN, Tech. Rep. 31, 2015.
- [37] nPlayer: NDN-based Video Player. <https://github.com/shockjiang/nplayer.git>.
- [38] You W, Mathieu B, Simon G. How to make content-centric networks interwork with cdn networks. Network of the Future (NOF), 2013 Fourth International Conference on the. IEEE, 2013. 1–5.
- [39] Leiner B M, Cerf V G, Clark D D, et al. A brief history of the internet. ACM SIGCOMM Computer Communication Review, 2009, 39(5):22–31.

- [40] Cert V, Kahn R. A protocol for packet network intercommunication. *IEEE Transactions on Communications*, 1974, 22(5):637–648.
- [41] Fenner B, Handley M, Holbrook H, et al. Rfc 4601: Protocol independent multicast-sparse mode (pim-sm): Protocol specification. 2006..
- [42] Ford A, Raiciu C, Handley M, et al. Rfc 6182, architectural guidelines for multipath tcp development. 2011..
- [43] Hopps C E. Analysis of an equal-cost multi-path algorithm. 2000..
- [44] Steward R. Rfc4960: Stream control transmission protocol, 2007.
- [45] Kohler E, Handley M, Floyd S, et al. Datagram congestion control protocol (dccp). 2006..
- [46] Ford B, Iyengar J R. Breaking up the transport logjam. *HotNets*, 2008. 85–90.
- [47] Hosseini M, Ahmed D T, Shirmohammadi S, et al. A survey of application-layer multicast protocols. *Communications Surveys & Tutorials, IEEE*, 2007, 9(3):58–74.
- [48] Liang J, Jiang J, Duan H, et al. When https meets cdn: A case of authentication in delegated service. *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE, 2014. 67–82.
- [49] Dunaytsev R, Moltchanov D, Koucheryavy Y, et al. A survey of p2p traffic management approaches: best practices and future directions. *Journal of Internet Engineering*, 2012, 5(1):318–330.
- [50] Koponen T, Chawla M, Chun B, et al. A data-oriented (and beyond) network architecture. *ACM SIGCOMM Computer Communication Review*, volume 37. ACM, 2007. 181–192.
- [51] Fotiou N, Nikander P, Trossen D, et al. Developing information networking further: From psirp to pursuit. *Broadband Communications, Networks, and Systems*. Springer, 2012: 1–13.
- [52] Lagutin D, Visala K, Tarkoma S. Publish/subscribe for internet: Psirp perspective. *Future Internet Assembly*, 2010, 84.
- [53] DANEWITZ C. Netinf: An information-centric design for the future internet. *Proc. 3rd GI/ITG KuVS Workshop on The Future Internet*, 2009.
- [54] Ahlgren B, D'Ambrosio M, DANEWITZ C, et al. Second netinf architecture description. 4WARD EU FP7 Project, Deliverable D-6.2 v2. 0, 2010..
- [55] Niebert N, Baucke S, El-Khayat I, et al. The way 4ward to the creation of a future internet. *Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on*. IEEE, 2008. 1–5.
- [56] Edwall T. Scalable & adaptive internet solutions (sail), 2011.
- [57] Zhang L, Estrin D, Burke J, et al. Named data networking (ndn) project. Technical report, Tech. report ndn-0001, PARC, 2010.
- [58] Ahlgren B, DANEWITZ C, Imbrinda C, et al. A survey of information-centric networking. *Communications Magazine, IEEE*, 2012, 50(7):26–36.
- [59] Ahlgren B, DANEWITZ C, Imbrinda C, et al. A Survey of Information-Centric Networking (Draft). In: Ahlgren B, Karl H, Kutscher D, et al., (eds.). *Information-Centric Networking*, Dagstuhl, Germany: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2011.
- [60] Xylomenos G, Ververidis C N, Siris V A, et al. A survey of information-centric networking research. *Communications Surveys & Tutorials, IEEE*, 2014, 16(2):1024–1049.

- [61] Tyson G, Sastry N, Rimac I, et al. A survey of mobility in information-centric networks: challenges and research directions. Proceedings of the 1st ACM workshop on Emerging Name-Oriented Mobile Networking Design-Architecture, Algorithms, and Applications. ACM, 2012. 1–6.
- [62] Pan J, Paul S, Jain R. A survey of the research on future internet architectures. Communications Magazine, IEEE, 2011, 49(7):26–36.
- [63] Smetters D, Jacobson V. Securing network content. Technical report, Citeseer, 2009.
- [64] Sun Y, Fayaz S K, Guo Y, et al. Trace-driven analysis of icn caching algorithms on video-on-demand workloads. Proceedings of the 10th ACM International Conference on emerging Networking Experiments and Technologies. ACM, 2014. 363–376.
- [65] Fayazbakhsh S K, Lin Y, Tootoonchian A, et al. Less pain, most of the gain: Incrementally deployable icn. ACM SIGCOMM Computer Communication Review, volume 43. ACM, 2013. 147–158.
- [66] Imbrenda C, Muscariello L, Rossi D. Analyzing cacheable traffic in isp access networks for micro cdn applications via content-centric networking. Proceedings of the 1st international conference on Information-centric networking. ACM, 2014. 57–66.
- [67] Jelenković P R. Asymptotic approximation of the move-to-front search cost distribution and least-recently used caching fault probabilities. Annals of Applied Probability, 1999. 430–464.
- [68] Wang S, Bi J, Wu J. Collaborative caching based on hash-routing for information-centric networking. ACM SIGCOMM Computer Communication Review, volume 43. ACM, 2013. 535–536.
- [69] Saino L, Psaras I, Pavlou G. Hash-routing schemes for information centric networking. Proceedings of the 3rd ACM SIGCOMM workshop on Information-centric networking. ACM, 2013. 27–32.
- [70] Jiang A, Bruck J. Optimal content placement for en-route web caching. Network Computing and Applications, 2003. NCA 2003. Second IEEE International Symposium on. IEEE, 2003. 9–16.
- [71] Tang X, Chanson S T. Coordinated en-route web caching. Computers, IEEE Transactions on, 2002, 51(6):595–607.
- [72] Li Z, Simon G. Time-shifted tv in content centric networks: The case for cooperative in-network caching. Communications (ICC), 2011 IEEE International Conference on. IEEE, 2011. 1–6.
- [73] 张国强, 李杨, 林涛, et al. 信息中心网络中的内置缓存技术研究. 信息中心网络中的内置缓存技术研究, 2014, 25(1):154–175.
- [74] Laoutaris N, Syntila S, Stavrakakis I. Meta algorithms for hierarchical web caches. Performance, Computing, and Communications, 2004 IEEE International Conference on. IEEE, 2004. 445–452.
- [75] Laoutaris N, Che H, Stavrakakis I. The lcd interconnection of lru caches and its analysis. Performance Evaluation, 2006, 63(7):609–634.
- [76] Eum S, Nakauchi K, Murata M, et al. Catt: potential based routing with content caching for icn. Proceedings of the second edition of the ICN workshop on Information-centric networking. ACM, 2012. 49–54.

- [77] Psaras I, Chai W K, Pavlou G. Probabilistic in-network caching for information-centric networks. Proceedings of the second edition of the ICN workshop on Information-centric networking. ACM, 2012. 55–60.
- [78] Zhang G, Li Y, Lin T. Caching in information centric networking: a survey. Computer Networks, 2013, 57(16):3128–3141.
- [79] Alimi, R R, A Y, et al. A survey of in-network storage systems. Technical report, RFC 6392, October, 2011.
- [80] Carofiglio G, Morabito G, Muscariello L, et al. From content delivery today to information centric networking. Computer Networks, 2013, 57(16):3116–3127.
- [81] Ghodsi A, Koponen T, Rajahalme J, et al. Naming in content-oriented architectures. Proceedings of the ACM SIGCOMM workshop on Information-centric networking. ACM, 2011. 1–6.
- [82] Zhang L, Afanasyev A, Burke J, et al. Named Data Networking. ACM SIGCOMM Computer Communication Review, 2014, 44(3):66–73.
- [83] Zhang L, Deering S, Estrin D, et al. Rsvp: A new resource reservation protocol. Network, IEEE, 1993, 7(5):8–18.
- [84] Floyd S, Jacobson V, McCanne S, et al. A reliable multicast framework for light-weight sessions and application level framing. ACM SIGCOMM Computer Communication Review, 1995, 25(4):342–356.
- [85] Deering S E. Multicast routing in internetworks and extended LANs, volume 18. ACM, 1988.
- [86] Clark D D, Lambert M L, Zhang L. Netblt: A high throughput transport protocol. ACM SIGCOMM Computer Communication Review, volume 17. ACM, 1987. 353–359.
- [87] Michel S, Nguyen K, Rosenstein A, et al. Adaptive web caching: towards a new global caching architecture. Computer Networks and ISDN systems, 1998, 30(22):2169–2177.
- [88] Tlv wiki. <https://en.wikipedia.org/wiki/Type-length-value>.
- [89] Yi C, Afanasyev A, Wang L, et al. Adaptive forwarding in named data networking. ACM SIGCOMM Computer Communication Review, 2012, 42(3):62–67.
- [90] NDNLPv2: NDN Link Protocol version 2. <http://redmine.named-data.net/issues/2763>.
- [91] Yu Y, Afanasyev A, Clark D, et al. Schematizing trust in named data networking. Proceedings of the 2nd International Conference on Information-Centric Networking. ACM, 2015. 177–186.
- [92] Venkataramani A, Kurose J F, Raychaudhuri D, et al. Mobilityfirst: A mobility-centric and trustworthy internet architecture. ACM SIGCOMM Computer Communication Review, 2014, 44(3):74–80.
- [93] Seskar I, Nagaraja K, Nelson S, et al. Mobilityfirst future internet architecture project. Proceedings of the 7th Asian Internet Engineering Conference. ACM, 2011. 1–3.
- [94] Afanasyev A. Addressing operational challenges in named data networking through ndns distributed database[D]. University of California Los Angeles, 2013.
- [95] Afanasyev A, Yu Y, Zhang L. Ndns: scalable and distributed name mapping service for ndn. Poster, UCLA Tech Forum 2013, May, 2013. <http://www.engineer.ucla.edu/techforum/index.html>.

- [96] IANA. Domain name system (DNS) parameters. <http://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml>.
- [97] Mockapetris P, Dunlap K J. Development of the domain name system. Symposium Proceedings on Communications Architectures and Protocols, New York, NY, USA: ACM, 1988. 123–133.
- [98] Yu Y, Wessels D, Larson M, et al. Authority server selection in dns caching resolvers. ACM SIGCOMM Computer Communication Review, 2012, 42(2):80–86.
- [99] Eastlake D, et al. RFC 2136: Secure domain name system dynamic update. 1997..
- [100] Lewis E, Hoenes A. RFC 5936: DNS Zone Transfer Protocol (AXFR). Technical report, RFC 5936, June, 2010.
- [101] Ohta M. RFC 1995: Incremental Zone Transfer in DNS (IXFR). 1996..
- [102] Ndn security library. <http://named-data.net/doc/ndn-cxx/current/tutorials/security-library.html>.
- [103] Weiler S, Ihren J. Minimally covering nsec records and dnssec on-line signing. Technical report, RFC 4470, April, 2006.
- [104] Laurie B, Sisson G, Arends R, et al. Rfc 5155: Dns security hashed authenticated denial of existencedns security (dnssec) hashed authenticated denial of existence. Request for Comments, 2008..
- [105] Fu W, Ben Abraham H, Crowley P. isync: a high performance and scalable data synchronization protocol for Named-Data Networking. Proceedings of the 1st international conference on Information-centric networking. ACM, 2014. 181–182.
- [106] Zone synchronization in ndns. <https://github.com/shockjiang/ndns-tr/tree/master/ns-sync>.
- [107] NDNS: DNS for NDN. <https://github.com/named-data/ndns>.
- [108] ChronoChat. <https://github.com/named-data/ChronoChat.git>.
- [109] CPU Sensor. <https://github.com/shockjiang/cpu-sensor.git>.
- [110] Mastorakis S, Afanasyev A, Moiseenko I, et al. ndnSIM 2.0: A new version of the NDN simulator for NS-3. Technical Report NDN-0028, NDN, January, 2015.
- [111] Spring N, Mahajan R, Wetherall D. Measuring isp topologies with rocketfuel. ACM SIGCOMM Computer Communication Review, 2002, 32(4):133–145.
- [112] Shi J, Zhang B. NDNLP: A Link Protocol for NDN. NDN Technical Report NDN-0006, 2012..
- [113] Dhamdhere A, Jiang H, Dovrolis C. Buffer sizing for congested internet links. INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, volume 2. IEEE, 2005. 1072–1083.
- [114] Wang Y A, Huang C, Li J, et al. Queen: Estimating packet loss rate between arbitrary internet hosts. Passive and Active Network Measurement. Springer, 2009: 57–66.
- [115] Afanasyev A, Shi J, Wang L, et al. Packet Fragmentation in NDN: Why NDN Uses Hop-By-Hop Fragmentation. Technical Report NDN-0032, 2015..
- [116] Gilbert E N. Capacity of a burst-noise channel. Bell system technical journal, 1960, 39(5):1253–1265.
- [117] Hoel P G, Port S C, Stone C J. Introduction to stochastic processes. Waveland Press, 1986.

- [118] Team N P. NDN Technical Memo: Naming Conventions. Technical report, NDN Technical Report NDN-0022, 2014.
- [119] NDN Flow: Adaptive Chunk Mode Prototype. <https://github.com/shockjiang/ndnflow.git>.
- [120] Afanasyev A, Moiseenko I, Zhang L. ndnsim: Ndn simulator for ns-3. <http://irl.cs.ucla.edu/ndnSIM.html>.
- [121] Nygren E, Sitaraman R K, Sun J. The akamai network: a platform for high-performance internet applications. *ACM SIGOPS Operating Systems Review*, 2010, 44(3):2–19.
- [122] Krishnamurthy B, Wills C, Zhang Y. On the use and performance of content distribution networks. *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*. ACM, 2001. 169–182.
- [123] Hofmann M, Beaumont L R. Content networking: architecture, protocols, and practice. Morgan Kaufmann, 2005.
- [124] Freedman M J, Freudenthal E, Mazieres D. Democratizing content publication with coral. *NSDI*, volume 4, 2004. 18–18.
- [125] Pathan A M K, Buyya R. A taxonomy and survey of content delivery networks. *Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report*, 2007..
- [126] Hoffman P, Schlyter J. The dns-based authentication of named entities (dane) transport layer security (tls) protocol: Tlsa. Technical report, RFC 6698, August, 2012.
- [127] Li Z, Bi J, Wang S. Http-ccn gateway: Adapting http protocol to content centric network. *Network Protocols (ICNP)*, 2013 21st IEEE International Conference on. IEEE, 2013. 1–2.
- [128] Geodns bind patch. <http://www.caraytech.com/geodns/>.
- [129] Maxmind. <https://www.maxmind.com>.
- [130] RMS: Resource Management System. <https://github.com/shockjiang/rms>.
- [131] RCRWireless News. Cisco demo on mwc2016: 5g network innovation. http://www.cisco.com/c/m/en_us/training-events/events-webinars/mobile-world-congress.html#demonstrations.
- [132] Ndn tools. <https://github.com/named-data/ndn-tools.git>.

致 谢

衷心感谢我的导师，清华大学毕军教授对我的精心指导，我于 2008 年加入毕军教授团队，多年来他的言传身教使我受益终身；同时感谢网络体系结构与 IPv6 实验室全体老师同学的帮助，尤其是刘冰洋老师、王旸旸老师、博士生付永红同学和我的室友王优同学。

我在美国加州大学洛杉矶分校 (UCLA) 进行 12 个月的学术交流期间，承蒙 Lixia Zhang 教授的指导；同时感谢 UCLA Internet Research Lab 成员 Alexander Afanasyev、余颖迪、谭杰文、商文涛、Michael Sweatt、Spyridon Mastorakis 等的帮助，这段学术交流让我受益良多，非常感谢。

感谢东南大学胡晓艳老师、北京邮电大学博士生南国顺，与两位的多次深入讨论开拓了思路，完善了设计，解决了很多难题。

感谢清华大学王继龙老师、中科院计算所谢高岗老师和其他三位匿名评审的老师帮助评审论文，并给出了细致宝贵的意见和建议。

还要感谢我的父母、兄长和女友，在我多年的求学生涯中予我支持；亲人健康是最大的福气，亲人支持是最大的力量。

六年的博士生活，收获的绝不止是知识；三十年的生命，经历的绝不单是颓唐。虽然今天的答卷并不尽如人意，我依然庆幸在当年那个时刻，选择了读博，选择了这样的生活。谨以此文答复我三十岁前的青春、答谢青春中有幸有缘相识的人。

生活是一场旅行，生命是一次修行。

“自强不息，厚德载物”。

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： _____

个人简历、在学期间发表的学术论文与研究成果

个人简历

- 1987 年 2 月 20 日出生于湖北省黄冈市黄梅县
- 2006 年 9 月考入清华大学软件学院，2010 年 7 月本科毕业并获得学士学位，成绩排名 12/65
- 2010 年 9 月免试进入清华大学计算机科学与技术系攻读博士学位至今，导师毕军教授
- 2011 年 9 月参加清华-伯克利全球技术创业一年期项目，所在团队被评为优秀团队
- 2014 年 7 月到加州大学洛杉矶分校 (UCLA) 进行为期 12 个月的学术交流，接收方导师 Lixia Zhang 教授

发表的学术论文

- [1] Xiaoke Jiang, Jun Bi, IS: Interest Set to Enhance Flow Transmission in Named-Data Networking, China Communications (IEEE), Vol.13, No.1z, pp65-71, 2016 (SCI Impact Factor 0.420/0.313)
- [2] Xiaoke Jiang, Jun Bi, Guoshun Nan, Zhaogeng Li, A Survey on Information-Centric Networking: Rationales, Designs and Debates, China Communications (IEEE), Vol.12, No.7, pp14-25, 2015 (SCI 检索号: 000358527700001, SCI Impact Factor 0.420/0.313)
- [3] Xiaoke Jiang, Jun Bi, Interest Set Mechanism to Improve the Transport of Named Data Networking, ACM Computer Communication Review, Vol. 43, No.4, pp515-516, also in proceedings of ACM SIGCOMM13 (poster, SCI 检索号: 000327465900074, SCI Impact Factor 1.102/2.695, EI 检索号: 20133616707798 & 20140217182476)
- [4] Xiaoke Jiang, Jun Bi, nCDN: CDN Enhanced with NDN, the 33rd IEEE International Conference on Computer Communications (INFOCOM14), Name-Oriented Mobility (NOM) Workshop, pp446-351, Toronto, Canada, 2014 (EI 检索号: 20143017982352)
- [5] Xiaoke Jiang, Jun Bi, You Wang, What Benefits Does NDN Have in Supporting Mobility, the 19th IEEE Symposium on Computers and Communications (ISCC14), Madeira, Portugal, 2014 (EI 检索号: 20144400139320)

- [6] Xiaoke Jiang, Jun Bi, You Wang, Pingping Lin, and Zhaogeng Li, A Content Provider Mobility Solution of Named Data Networking, in proceedings of the 20th IEEE International Conference on Network Protocols (ICNP12), pp1-2 Austin, USA, 2012 (poster, EI 检索号: 20131016092876)
- [7] Xiaoke Jiang, Jun Bi, Yangyang Wang, Zhijie He, Wei Zhang, Hongchen Tian, IPv6 Evolution, Stability and Deployment, in proceedings of the 19th IEEE International Conference on Network Protocols (ICNP11), pp123-124, Vancouver, Canada, 2011 (poster, EI 检索号: 20115214633144)
- [8] Xiaoke Jiang, Jun Bi, You Wang, Pingping Lin, Zhaogeng Li, An Easy Matrix Computation based Simulator of NDN, the 3rd IEEE International Conference on Networking and Distributed Computing (ICNDC12), pp36-39, Hangzhou, China, 2012 (EI 检索号: 20130716024854)
- [9] Xiaoke Jiang, Jun Bi, You Wang, MCBS: Matrix Computation Based Simulator of NDN, Journal of Computers (Academy Publisher), Vol. 9, Num. 9, pp2007-2012, 2014
- [10] 蒋小可, 毕军, 王旸旸, IPv6 网络部署监测及演化分析, 第四届中国互联网年会, 2015
- [11] You Wang, Jun Bi, Xiaoke Jiang, Mobility Support in the Internet Using Identifiers, the 7th ACM International Conference on Future Internet Technologies (CFI12), pp37- 42, Seoul, Korea, 2012 (EI 检索号: 20124315607780)
- [12] Zhaogeng Li, Jun Bi, Sen Wang, Xiaoke Jiang, The Compression of Pending Interest Table with Bloom Filter in Content Centric Network, the 7th ACM International Conference on Future Internet Technologies (CFI12), pp47, Seoul, Korea, 2012 (EI 检索号: 20124315607783)
- [13] 田洪成, 毕军, 蒋小可, 王德凯, 张威, 一种快速且安全的概率标记追溯技术, 《清华大学学报(自然科学版)》, Vol. 50, No.4, pp542-547, 2011 (EI 检索号: 20113214224186)
- [14] Hongcheng Tian, Jun Bi, Wei Zhang, Xiaoke Jiang, EasyTrace: Easily-Deployable Light-Weight IP Traceback on an AS-Level Overlay Network, in proceedings of the 19th IEEE International Conference on Network Protocols (ICNP11), pp129-130, Vancouver, Canada, 2011 (poster, EI 检索号: 20115214633147)
- [15] Pingping Lin, Jun Bi, Hongyu Hu, Tao Feng, Xiaoke Jiang, A Quick Survey on Selected Approaches for Preparing Programmable Networks, in proceedings of ACM AINTEC2011, pp160-163, Bangkok, Thailand, 2011 (EI 检索号: 20120414716596)
- [16] Hongcheng Tian, Jun Bi, Xiaoke Jiang, An Adaptive Probabilistic Marking Scheme for Fast and Secure Traceback, Networking Science (Springer), Vol. 2, No.1-2, pp 42-51

- [17] 林萍萍, 毕军, 胡虹雨, 蒋小可, 一种面向 SDN 域内控制平面可扩展性的机制, 《小型微型计算机系统》, Vol.34, No.9, pp1980-1983, 2013
- [18] Hongcheng Tian, Jun Bi, Xiaoke Jiang, Wei Zhang, A Probabilistic Marking Scheme for Fast Traceback, International Conference on Evolving Internet (INTERNET 2010), pp.137-141, Valencia, Spain, 2010

软件著作权

- [1] 毕军, 蒋小可, 刘冰洋, 张丽杰, 竺昱. 域间源地址验证系统模拟测试软件 v1.0: 中国, 已授权, 登记号 2011SR056506, 首次发表日期 2011 年 3 月 31 日
- [2] 毕军, 孙雅媛, 刘冰洋, 竺昱, 蒋小可, 张宝宝. 域间真实地址验证管理系统 v1.0: 中国, 已授权, 登记号 2011SR057363, 首次发表日期 2011 年 3 月 31 日

科研项目

- [1] 国家高技术研究发展计划 (863 计划) 项目, “未来网络体系结构和创新环境”(2013AA013500), 2013-2015
- [2] 国家十二五科技支撑计划课题, “IPv6 网络管控技术及其应用示范”, (2012BAH01B01), 2012-2014
- [3] 华为合作项目研究项目, “基于 ICN 的数据中心网络关键技术”(YB2013090053), 2012-2013
- [4] 华为技术合作研究项目, “HTTP-NDN 网关关键技术”(YBCB2011053), 2011-2012
- [5] 中兴合作项目, “开放标准化网络架构及关键技术”(20112001436), 2011-2012
- [6] Silicon Valley Community Foundation 合作项目, “ILNP 研究”(20113000446), 2012

开源项目 (正式列为合作者/贡献者)

- [1] NDNS^[107], NDN 网络中名字解析服务, 已部署于 NDN Testbed
- [2] ndnSIM^[120], NDN 研究中广泛使用的网络模拟器
- [3] ndn-cxx^[15], NDN 程序开发 C++ 语言库, NDN 研究社区中广泛使用
- [4] NDN Tools^[132], NDN 网络工具库, 如 ndn-ping 等

参加的学术活动

- [1] 2016 China Communications 审稿人
- [2] 2015 中国计算机协会 (CCF) 第四届互联网年会博士生论坛学生主席
- [3] 2013 SIGCOMM'13 会议抄写员
- [4] 2009 - 2015 亚洲未来互联网论坛 (AsiaFI) 青年研究组委员会成员
- [5] 2009 - 2013 亚洲未来互联网论坛网络管理员

攻读博士学位期间所获个人荣誉

- [1] 2014 清华大学综合一等奖学金, 清华大学 375 届博士生论文二等奖
- [2] 2013 清华大学综合二等奖学金, ACM SIGCOMM Travel 资助奖
- [3] 2012 清华大学光华二等奖学金, 清华大学社会实践三等奖
- [4] 2011 清华大学综合一等奖学金