# 绪论

## 研究背景

当前的互联网架构是基于70年代提出的TCP/IP协议栈实现的，其中由IP地址标识的目的端点、源端点地址是上层通信实现的基础。然而随着硬件技术和网络应用的发展，网络变得更加面向内容，而不是面向端点。人们更加在意网络所提供的内容，而不是请求的内容在哪个IP地址。内容请求和内容发布成为了网络通信的主要组成部分。

命名数据网(NDN, Named data networking)利用这一点，摒弃TCP/IP协议栈对地址的需求，将所有上层通信归纳为由数据名称标识的数据请求和数据应答间的交互。这样的设计思路带来的优势有对网络多播和数据共享的天然支持，对多个连接的同时利用，和面向内容的信任模型等等。然而同样也带来了许多挑战，比如应用层的各个应用需要根据其需求，进行命名空间的重新设计。

网络游戏是网络的重要应用之一。基于IP的网络游戏经历了由点对点(P2P, peer-to-peer)架构，到客户端服务器(Client/Server)架构，再到P2P重新开始被研究的发展历程。C/S架构具有着控制方便的优势，但是有着流量汇聚在几个服务器，和服务器单点出现异常会直接导致服务不可使用的劣势。然而由于P2P架构带来的控制上的困难性，当前流行的商业化的网络游戏一致采用了C/S架构。

本文提供基于NDN的大型网络游戏(Massive Multiplayer Online Role Playing Game, MMORPG)应用的设计。其利用了NDN网络的优点，并且是自组织、基于完全P2P架构的。该应用不仅是网络游戏联机模型的新思路，而且对别的需要基于位置进行发现、同步的NDN应用有启发意义。同时，该应用是基于虚拟位置，进行发现、同步，利用NDN的第一款应用。

## 研究现状

### 新网络架构

新网络架构是十年来的研究的新兴领域。当前由美国自然科学基金会(NSF)资助的新网络架构研究项目主要有NDN项目(<http://www.nets-fia.net/>)，MobilityFirst项目，NEBULA项目，eXpressive Internet Architecture项目，ChoiceNet项目。这些项目的共同特点是认识到了TCP/IP架构不再适合当前的网络使用用例。例如，NEBULA项目利用大规模的云存储技术，使得服务提供、控制变得更加迅速和方便；MobilityFirst项目通过网络资源利用的优化，实现流动性和效率的平衡。

### 新网络架构的主要思路

新网络架构的研究思路主要有两个类别：基于当前TCP/IP协议栈的改良，或是完全摒弃TCP/IP协议栈的革命性思路。

ICN(Information-Centric Networking)：基于内容的网络架构。ICN是一种革命性思路，虽然其中的多种实现可以作为IP协议的负载，或者将IP作为其负载 。Named Data Networking属于此范畴，也满足可以作为IP负载，或是将IP作为其负载的特点。

SDN(Software Defined Networking)

### NDN项目

NDN项目的研究始于2006年Van Jacobson发表的文章Networking Named Content(ref)。同时PARC的研究小组提供了项目的初步实现：基于内容的网络架构实现项目(Content Centric Networking(CCNx))。在此之后，项目的理念仍然与当初的设计基本一致，但是实现上已有了许多完善。例如在2014年，NDN小组完成了路由节点的新实现，NRD/NFD(NDN Routing Daemon/NDN Forwarding Daemon)和NDN-tlv，基于类型-长度-值的NDN数据包结构；在2013年，完成了ChronosSync同步模型的设计、实现，完成了客户端函数库的面向对象实现等等。

目前作者的指导老师，美国加州大学洛杉矶分校的Prof. Lixia Zhang 和Prof, Jeff Burke在领导项目的研究，而作者的项目可以简单理解为将节点发现和虚拟位置临近性综合考虑后的同步模型的拓展实现。

### 游戏联机

随着网络愈发便捷，多人联机网络游戏成为网络娱乐的重要组成部分。最为流行的MMORPG，魔兽世界有着超过1400万的注册用户，而其开发公司，暴雪公司有着8000台服务器为该游戏提供联机支持。随着许多MMORPG的商业化，如何更高效的提供网络游戏联机服务也成为了网络研究的一个小方向。近来，多篇基于IP网络架构的P2P游戏联机方案(ref)被多个研究组织提出。但是这些方案都基于IP网络架构，其P2P的体现基本在DHT的应用。

## 研究目的和意义

本研究项目的主要目的有：开发基于NDN的MMORPG应用，为更高效的网络游戏通信提供实现；解决完全自组织带来的发现、同步及控制问题；衡量NDN网络架构在类似用例下的实用性；为同类型项目提供命名空间开发的启发和样例。

项目的意义主要有：通过基于P2P架构的自组织设计，解决当前大型网络联机游戏面临的几个问题，即服务器负载过重，维护成本高昂，依然无法满足超大量用户同时存在于一个虚拟空间的需求。衡量NDN的设计思路是否可以为此类应用带来效率上的提升。为类似应用，例如ICN-CarSpeak(ref)的开发提供启发和帮助。

## 研究内容

本文研究的内容涵盖多个层面的设计和实现。设计角度而言， 本文分析了NDN的已有同步模型，并根据MMORPG的用例需求进行了新的同步方案的设计和改良。同时，本文分析了发现过程面临的抽象问题kNN问题的主要设计思路及实现。实现角度而言，为了对基于Unity游戏引擎的游戏实现提供支持，作者开发了基于C#语言的NDN客户端函数库；基于自主开发的函数库，提供了发现模块和更新模块的实现；并在具体游戏引擎中进行了应用和测试。

## 论文结构

本文的组织结构如下：

第1章：绪论。本章介绍论文研究背景、研究现状、研究的目的和意义，研究内容以及总体的组织结构。

第2章：相关技术概述。

第3章：系统分析。本章对系统的需求、系统框架、系统功能以及数据流进行了详细的描述和分析。

第4章：系统设计。本章主要对系统使用到的类进行了详尽的分析和描述。

第5章：系统实现。本章详细讲述了系统实现的过程、界面设计以及系统测试，运行结果及功能展示。

第6章：总结与展望。对系统进行全面的归纳和总结，对尚存在的问题进行分析，并对进一步工作进行展望和规划。

# 相关技术概述

## 命名数据网

NDN是本文的设计所针对的基础架构。其主要特点是将数据，而不是地址，作为网络架构的基本单元，并把数据和地址、安全及访问区分开来，同时以数据的名称来请求对应的数据。以下将对该网络架构进行概述。

## 综述

当今网络的架构和设计原则继承着60年代、70年代的思路，当时所希望解决的主要问题是资源共享，即由多个终端远程访问共享的昂贵资源，比如读卡器或是高速磁带硬盘。由此开发出的通信模型可以抽象为两点之间的信息交互：服务请求方希望使用服务提供方的特定服务。因而IP报文中包含两个标示符，目的地址和源地址。而网络中的几乎所有流量(大多是基于TCP协议的会话)，都是数对请求方、提供方之间的通信。

基于数据报交互的网络架构形成的50年来，计算机及其附件变得便宜和大众化。网络所提供的方便的互联性和低存储成本使得大众对网络价值的衡量变得倾向于其所提供的数据本身，而不是数据具体存储在哪里。

直接、统一的解决这一供求不对等问题的方案是用“什么”取代“哪里“，即基于数据名称而不是位置的命名数据网将是能够更好描述当今网络用户需求的网络架构模型。CCN底层并没有“端”或地址的概念，取而代之的数据的名称。然而，其设计仍然保留TCP/IP协议栈简单、鲁棒和可扩展的设计原则。

图1对比CCN和IP协议栈。协议栈中大多数层体现了两者之间的相似性，比如第二层的成帧协议体现了物理链路中发送方、接收方之间的协商，第四层传输层协议体现了数据或服务的请求者、提供者之间的协商。承上启下，为双方提供统一接口的第3层，即网络层，体现了IP协议成功的许多因素。首先，网络层对链路层提供的服务的需求很少，无状态、不可靠、不保序、尽力而为即可。CCN的网络层(即第三层)，在这一点上和IP是吻合的，即不对链路层提供的服务提出过高要求。这一点为其保留了IP较多优点。同时值得一提的是，CCN也可以作为多层的负载，包括作为IP的负载。

CCN在许多关键点上不同于IP。其中两者是策略和安全。两者作为其协议栈中新出现的层。CCN可以最大化利用同时存在的多个连接，比如以太网、3G、蓝牙和802.11，因为CCN和第二层的关系更加简单。策略层在变化的外界条件下，根据多个同时存在的连接，做出详细的、动态的优化，从而最大化的利用多个连接。安全层保障数据本身是受信任的，而不是数据所经过的通路是受信任的，由此避免了安全策略上IP网络的许多问题。

### 设计理念

CCN网络通信是由数据请求方驱使的。网络中有两种包，数据请求(Interest)和数据应答(Data)。数据请求方向本地的所有连接广播数据请求，任何收到数据请求，并且有名称满足要求的数据的节点可以应答。数据应答包的发出都是由收到特定的请求所触发，换言之，数据应答包都是为了满足特定的请求。

当且仅当数据请求包的名称是数据应答包的名称的前缀子串时，数据应答可以满足数据请求。CCN 是层次结构的，所以前缀的吻合可以描述为数据应答包的名称是数据请求包的名称的子树。IP用同样的方式来解析树状的IP地址，即<网络地址，子网地址，主机地址>。IP的经验表明，这样的方式可以实现路由表树型的高效压缩和快速查找。名称前缀也可能是上下文相关的，比如”/ThisRoom/Projector”可以用来和局部环境中存在的投影仪进行交互。而”/Local/Friends”可以和局部环境中的朋友进行交互。

CCN节点的工作方式和IP节点是类似的：节点收到数据包，进行最长前缀匹配，并由匹配结果决定下一步行动。图3是CCN前递逻辑的结构图。其中主要有三个数据结构：前递表(Forwarding Information Base)，数据缓存(Content Store)和待应答表(Pending Interest Table)。

前递表的作用是将数据请求传递向可能有对应的数据应答的节点。这与IP的路由表是相近的，但是它允许一个表项和多个接口进行匹配，而不是对于每一个接口，存在一张路由表。这一点体现了CCN不局限于基于一颗生成树的前递，其允许一个节点同时对多个连接的数据源进行请求，多个数据源也可以同时处理收到的请求。

数据缓存和IP路由的缓存作用近似，但是替换策略不同。由于每个IP包包含了源和目的地址，其对于别的点之间的交互是不起到作用的。因此，IP路由器在将收到的包写入缓存，然后转发出去之后，就可以将该包擦除(利用最近使用(MRU)替换)。CCN包是幂等，自识别和自授权的，因而每一个包有可能可以满足多个用户的数据请求(比如多个看同样的报纸和Youtube视频的用户)，只要数据请求的名称匹配。为了最大化数据共享的可能性，从而最小化上行带宽的占用和下行的延迟，CCN尽可能长时间的缓存数据包(利用最近/最少未使用替换)。

待应答表保存了向数据提供方前递的数据请求(上行请求)，因此当收到数据提供方的应答时，可以根据待应答表中的记录将数据应答传送给数据请求方。因此，在CCN中，只有数据请求包在上行时经过路由，并在经过路由节点的同时留下一系列的记录，下行的数据应答包可以根据路由节点的记录找到数据的请求者。在路由节点收到下行数据应答后，其将所收数据应答对待应答表中的对应表项中的所有节点进行多播，并且擦除该表项。长时间没有收到数据应答的数据请求会超时，而如果数据请求方仍然希望请求该数据，请求方应当负责重新发送数据请求。

当数据请求包到达某节点的某接口后，节点首先进行名称的最长前缀匹配。上述数据结构的查找是有序的，所以数据缓存的匹配优先级高于待应答表，高于前递表。

因此，在一个节点收到请求后，如果数据缓存找到了满足请求的数据，会直接返回该数据，同时收到的请求由于已经被满足而会被丢弃。

如果数据缓存没有前缀匹配项，而待应答表中有名称完全匹配项，收到请求的接口会被加入到路由节点待应答表中该名称的下行接口表上，同时收到的请求会由于别人已经在请求同名数据，且别人的请求已经向数据源上行而被丢弃。这时路由节点所要做的只是当数据应答到来时，将数据应答也向这个数据请求的接口发送一份。

如果二者都没有匹配项，而前递表有匹配项，则数据请求根据前递表的匹配项上行。收到请求的接口将会被从前递表匹配项上移除，如果前递表匹配项此时仍不为空，则对于其所记录的每一个接口发送该数据请求。并以收到请求的接口，创建一个新的待应答表项。

如果三者都没有匹配的表项，则丢弃该数据请求，因为收到请求的路由节点既没有满足要求的数据，也不知道该向哪里前递已获取满足要求的数据。

数据应答包的处理要相对简单。由于数据应答包不经过路由，而是直接根据待应答表进行下行，在收到数据应答时，首先进行名称的最长前缀匹配。如果数据缓存有匹配项，则收到了重复的数据，予以丢弃。如果前递表发现了满足项，则说明待应答表中没有满足项，说明数据没有请求者，是不需要的，予以丢弃。如果带应答表中发现满足项，则(可选的)进行数据核实和写入数据缓存，之后根据待应答表的满足项移除收到数据应答的接口后，对满足项中的其它接口进行下行。

不同于IP先入先出的缓存模型，CCN缓存模型允许整个网络的节点缓存实现透明缓存。所有节点可以根据自己的能力和策略进行缓存。

通过数据请求指定的多点数据回取的特点使得CCN在变化快速的环境下依然可以灵活应用。任何处在多个网络中的节点可以作为其所处的多个网络之间的缓存和路由。利用其缓存，一个移动的节点可以作为多个彼此不相连区域进行连接的媒介，或者为不连贯的链路提供延迟的连接。因此，CCN的传输是允许干扰的(Disruption Tolerant Networking)。数据请求/应答的交互在仅有局域网连接时也可以正常工作。

### 同步模型

同步问题应用在文件共享服务，实时聊天应用等等多个用例。多人联机游戏也是重要用例之一。自组织的多人联机游戏的每一个节点都在运行游戏的一个独立实体，每一个节点都对自己的实体负责的游戏对象有决定权，并能发布关于它们的数据更新。对于自己没有决定权的对象，游戏实体应该通过网络接收数据更新，从而保持信息的一致性。

同步问题同是也是NDN网络研究的重要问题。虽然为了提供更广的支持，NDN仍然拥有“只从数据发布源获取数据“的功能，但是由于弱化了数据存储位置的概念，能体现NDN思想的设计思路并不利用该功能。由此带来的问题是，多个节点的数据希望保持一致时，一般不存在某一特定中心节点可以为别的节点数据负责，换言之，不存在服务器可以为多个客户端提供一致的数据。因此，发布数据的端点如果希望别的端点可以获得更新的数据，应该通过命名空间的设计，无二意的对其发布的数据进行命名，并且以双方提前协商好的方式进行通信。

CCNx在提出时包含了最基础的同步设计方案，本文将在下面进行概述。然而该基础思路存在许多问题，为了更高效的实现同步，NDN小组提出了ChronosSync(ref)，其特点也会在下面进行概述。

#### CCNx Sync

#### Chronos Sync

### 客户端函数库

虽然NDN对于底层(例如链路层)协议的实现已经有了规划，但是为了和当前的硬件环境保持兼容，NDN通信的实现仍然是作为传输层协议的负载的。因此，客户端函数库由Java、Python、C++等高层语言实现，就传输方式而言，NDN客户端函数库直接利用这些高层语言所包含的传输层接口的实现，NDN流量将以NDN协议的格式进行解析，但以TCP或UDP协议为负载进行通信。

开发过程使用的游戏引擎Unity3D为了兼顾不同的平台，采用了DotNet架构的开源实现Mono Framework，并且提供了从C#，语法上类似Javascript的UnityScript，以及自定义语言Boo到IL的编译工具。当前的NDN开发者函数库并没有针对DotNet的实现，因此本文作者的工作也包含了与ndnd-tlv兼容的NDN开发者函数库的开发。在此简单介绍NDN开发者函数库的结构。

由于设计思路较为明确，且语法及设计思路上C#与Java是类似的，大多数的函数开发工作并不困难。需要特别注意的重点是二者不同的异常处理、传输层接口、安全/加密实现和JavaNIO提供的高效字节缓冲区。

## 游戏联机技术

多人游戏联机主要采用两种模型。早期的互联网游戏的联机是点对点的，每个节点所运行的是一个独立的游戏实体，通过同步使得多个独立游戏实体在每一个时间点保持相同。这样的游戏有帝国时代1、2以及星际争霸1。早期点对点技术带来了许多问题，例如实际传输流量更大，联机速度由连接速度最差的端点决定。

游戏Quake引入了客户端/服务器的联机架构。每个节点所运行的应用只起到用户接口的作用，即通知服务器用户的输入，并对服务器返回的数据进行渲染。所有的决定权在服务器端， 信息的处理也在服务器端完成。如此架构遇到的主要问题是延迟。用户在赋予自己的角色前进指令后，需要等到一个往返时间之后服务器的确认才可以在本地体现角色的前进。为了解决该问题，Quake 2游戏的引入了本地渲染器的预测技术。对于上述的用例，本地会预测服务器端的确认，并作出渲染。该技术同时引入了对错误预测的处理。

近10年来游戏联机的C/S模型逐步成熟，并应用在主流商业化的大型MMORPG游戏中。然而其服务器端的流量汇聚，和服务器单点出现问题会导致服务不可用的问题依然没有得到解决。因而近年来仍有许多研究机构提出了基于DHT等技术的IP网络下点对点自组织的游戏实现。(ref)

## k最近邻居相关的数据结构

kNN问题是游戏发现、同步问题中虚拟环境局部性要求的抽象化描述。

由于整个游戏的虚拟空间可能过大，其中包含的节点过多，希望每一个节点获取整个虚拟空间的全部状态信息是不现实的。同时由于游戏本身的要求，每个节点只需要获取在它的感知范围内的其它对象，即为游戏虚拟环境的局部性要求。该要求可以理解为，每个节点在任一时刻只希望了解自己的k-近似最近邻居。此要求在系统需求分析中也有详细的叙述 (ref)。

### 静态树

### 动态树

### 位置敏感哈希

## 数据表示相关的数据结构

为了更高效的实现同步，在不需要数据更新时不进行更新，在后面的设计方案中每一个数据请求包含了本地实体的状态的体现。该状态有多种方式体现。以下是两种实际方案，其优劣在理论分析部分(ref)也有所记述。

### 哈希

### 可逆转的Bloom Filter

## 开发引擎Unity3D

## 本章小结

# 系统需求分析

## 系统需求

局部性

根据游戏设计，每个节点应当具有一个球形的感知范围。节点应当只收到落在自己的感知范围内的对象的位置和行动的更行。不仅如此，如前kNN模块所述(ref)，由于整个游戏的虚拟空间过大，其中包含的节点过多，希望每一个节点获取整个虚拟空间的全部状态信息是不现实的。此要求即为游戏虚拟环境的局部性要求。该要求可以理解为， 网络中每个节点请求、收到的内容应当尽可能的对自己有用，且尽可能的少。局部性是避免广播的内容过多，网络流量过大，从而减少延迟的重要手段。

实时性

当对象移动或是做出一定动作时， 每个对其感兴趣的节点，即包含该对象在感知范围的节点，应当尽快的获得关于移动或是动作的更新。特别是当一个对象从一个节点的感知范围外移动到感知范围内，或是从感知范围内移动到感知范围外，即加入和离开事件发生时，节点应该尽可能早的获悉事件的发生。

大规模可拓展性

由于本项目旨在解决大规模网络游戏的联机问题，只要不是用户主观需求，所有节点应当尽可能的处于同一个虚拟世界中，因此大规模可拓展性是设计方案中需要考虑的。这一点体现在NDN网络中即为尽可能多的利用节点的数据缓存中的信息，通过全局共享的命名空间，实现在满足实时性的同时存在最大规模的数据共享。

异常处理

在实际网络环境中，丢包、乱序、掉线时有发生。由于当前的客户端函数库是基于TCP协议的，因此丢包和乱序会被操作系统和底层捕捉到，并有对应的重传策略进行保障。但是，考虑到TCP协议的头部较大，影响传输效率，实际实现会采用UDP协议。同时，理论NDN架构并不基于TCP，理论NDN网络层也是无状态的，不提供针对丢包、乱序的处理。综上，更高层的协议，在本例中，即为游戏应用通讯的应用层协议，应当为这些异常情况负责。游戏应用应当在出现丢包、乱序时尽可能的自行做出处理，并在侦测到对象掉线后予以删除。

## 系统框架

## 本章小结

# 系统设计

## 客户端函数库

## 设计详述

本部分详细叙述应用设计和工作流程。

## 模块设计

本项目划分为以下模块，下面对各个模块的设计分别进行叙述。

### 发现模块

该要求可以诠释为，每个节点在任一时刻只希望了解自己的k-R最近邻居。相对于用固定的半径限制每个节点的感知范围，用kNN的优势在于其可以更好的利用网络资源。在节点所处的环境对象密集时，节点可以只获取最近的k个对象的位置，避免占用过大带宽。反之，在环境对象稀疏时，节点可以发现更远的对象。从游戏设计的角度来理解，可以视为是动态的调整感知范围的大小。

### 更新模块

## 本章小结

# 系统实现及测试

## 客户端函数库实现及测试

## 同步模块实现及测试

## 本章小结

# 系统理论分析、评估及优化

## 系统分析

## 其它设计思路

### 基于可逆转Bloom Filter的发现

### 基于地点敏感哈希的实现

## 理论评估

### 与IP下实现的对比

### 与其它设计思路的对比

## 系统优化

# 结论与展望

# 参考文献

# 致谢

# 翻译

命名数据网络

摘要

现今，网络使用已由数据发布与请求所主宰，而网络通信仍是基于多个端点的地址。用户的数据及服务请求需要经过所请求数据到数据所在地址的映射。本文提出以内容为中心的网络架构(Content Centric Networking, CCN)，将数据，而不是地址，作为网络架构的基本单元，并把数据和地址、安全及访问区分开来，同时以数据的名称来请求对应的数据。

简介

当今网络的架构和设计原则继承着60年代、70年代的思路，当时所希望解决的主要问题是资源共享，即由多个终端远程访问共享的昂贵资源，比如读卡器或是高速磁带硬盘。由此开发出的通信模型可以抽象为两点之间的信息交互：服务请求方希望使用服务提供方的特定服务。因而IP报文中包含两个标示符，目的地址和源地址。而网络中的几乎所有流量(大多是基于TCP协议的会话)，都是数对请求方、提供方之间的通信。

基于数据报交互的网络架构形成的50年来，计算机及其附件变得便宜和大众化。网络所提供的方便的互联性和低存储成本使得2008年一年就发布了500EB的数据。大众对网络价值的衡量变得倾向于其所提供的数据本身，而不是数据具体存储在哪里。

这两种模型之间的差异在以下几个方面影响到了用户。

可用性：高速、可靠的数据访问需要利用不自然的、预先计划好的，且不同应用之间难以共享的解决方案，例如CDN和点对点网络。这些解决方案同时消耗了多余的带宽资源。

安全：基于来源位置和连接信息的安全模型容易混淆对数据本身的信任。

地点有关性：将数据映射到所在位置使得配置和实现网络服务更加复杂。

直接、统一的解决这些问题的方案是用“什么”取代“哪里“。端对端通信模型仅适用于60年代的用例。本文认为，基于数据名称而不是位置的命名数据网将是能够更好描述当今网络用户需求的网络架构模型。CCN底层并没有“端”或地址的概念，取而代之的数据的名称。然而，其设计仍然保留TCP/IP协议栈简单、鲁棒和可扩展的设计原则。

图1对比CCN和IP协议栈。协议栈中大多数层体现了两者之间的相似性，比如第二层的成帧协议体现了物理链路中发送方、接收方之间的协商，第四层传输层协议体现了数据或服务的请求者、提供者之间的协商。承上启下，为双方提供统一接口的第3层，即网络层，体现了IP协议成功的许多因素。首先，网络层对链路层提供的服务的需求很少，无状态、不可靠、不保序、尽力而为即可。CCN的网络层(即第三层)，在这一点上和IP是吻合的，即不对链路层提供的服务提出过高要求。这一点为其保留了IP较多优点。同时值得一提的是，CCN也可以作为多层的负载，包括作为IP的负载。

CCN在许多关键点上不同于IP。其中两者是策略和安全。两者作为其协议栈中新出现的层。CCN可以最大化利用同时存在的多个连接，比如以太网、3G、蓝牙和802.11，因为CCN和第二层的关系更加简单。策略层在变化的外界条件下，根据多个同时存在的连接，做出详细的、动态的优化，从而最大化的利用多个连接。安全层保障数据本身是受信任的，而不是数据所经过的通路是受信任的，由此避免了安全策略上IP网络的许多问题。

本文的2~5章将描述CCN的架构和流程；第6章基于我们的初步实现，对其进行评估；第7、8章讨论未来的工作方向并总结。

CCN节点模型

CCN网络通信是由数据请求方驱使的。网络中有两种包，数据请求(Interest)和数据应答(Data)。数据请求方向本地的所有连接广播数据请求，任何收到数据请求，并且有名称满足要求的数据的节点可以应答。数据应答包的发出都是由收到特定的请求所触发，换言之，数据应答包都是为了满足特定的请求。由于请求包、应答包都是通过名称索引的，多个请求同样名称的节点收到的应答包可以在广播媒介上用传统的多播压缩手段共享[3]。

当且仅当数据请求包的名称是数据应答包的名称的前缀子串时，数据应答可以满足数据请求。CCN名称是由多个部分组成的透明的二进制对象(见图4)。传统意义上的名称都是层次结构的，所以前缀的吻合可以描述为数据应答包的名称是数据请求包的名称的子树(见3.2)。IP用同样的方式来解析树状的IP地址，即<网络地址，子网地址，主机地址>。IP的经验表明，这样的方式可以实现路由表树型的高效压缩和快速查找。这种匹配方式中其中值得一提的一点是节点可能收到对于尚不存在于网络中的数据的请求，这使得数据的提供方可以利用数据请求方指定的名称产生并发布数据。这样的“动态名称”使得CCN透明的支持动态生成的数据和静态缓存的数据，这两者在当今网络中扮演了重要角色。名称前缀也可能是上下文相关的，比如”/ThisRoom/Projector”可以用来和局部环境中存在的投影仪进行交互。而”/Local/Friends”可以和局部环境中的朋友进行交互。

CCN节点的工作方式和IP节点是类似的：节点收到数据包，进行最长前缀匹配，并由匹配结果决定下一步行动。图3是CCN前递逻辑的结构图。其中主要有三个数据结构：前递表(Forwarding Information Base)，数据缓存(Content Store)和待应答表(Pending Interest Table)。

前递表的作用是将数据请求传递向可能有对应的数据应答的节点。这与IP的路由表是相近的，但是它允许一个表项和多个接口进行匹配，而不是对于每一个接口，存在一张路由表。这一点体现了CCN不局限于基于一颗生成树的前递，其允许一个节点同时对多个连接的数据源进行请求，多个数据源也可以同时处理收到的请求。

数据缓存和IP路由的缓存作用近似，但是替换策略不同。由于每个IP包包含了源和目的地址，其对于别的点之间的交互是不起到作用的。因此，IP路由器在将收到的包写入缓存，然后转发出去之后，就可以将该包擦除(利用最近使用(MRU)替换)。CCN包是幂等，自识别和自授权的，因而每一个包有可能可以满足多个用户的数据请求(比如多个看同样的报纸和Youtube视频的用户)，只要数据请求的名称匹配。为了最大化数据共享的可能性，从而最小化上行带宽的占用和下行的延迟，CCN尽可能长时间的缓存数据包(利用最近/最少未使用替换)。

待应答表保存了向数据提供方前递的数据请求(上行请求)，因此当收到数据提供方的应答时，可以根据待应答表中的记录将数据应答传送给数据请求方。因此，在CCN中，只有数据请求包在上行时经过路由，并在经过路由节点的同时留下一系列的记录，下行的数据应答包可以根据路由节点的记录找到数据的请求者。在路由节点收到下行数据应答后，其将所收数据应答对待应答表中的对应表项中的所有节点进行多播，并且擦除该表项。长时间没有收到数据应答的数据请求会超时，而如果数据请求方仍然希望请求该数据，请求方应当负责重新发送数据请求。

当数据请求包到达某节点的某接口后，节点首先进行名称的最长前缀匹配。上述数据结构的查找是有序的，所以数据缓存的匹配优先级高于待应答表，高于前递表。

因此，在一个节点收到请求后，如果数据缓存找到了满足请求的数据，会直接返回该数据，同时收到的请求由于已经被满足而会被丢弃。

如果数据缓存没有前缀匹配项，而待应答表中有名称完全匹配项，收到请求的接口会被加入到路由节点待应答表中该名称的下行接口表上，同时收到的请求会由于别人已经在请求同名数据，且别人的请求已经向数据源上行而被丢弃。这时路由节点所要做的只是当数据应答到来时，将数据应答也向这个数据请求的接口发送一份。

如果二者都没有匹配项，而前递表有匹配项，则数据请求根据前递表的匹配项上行。收到请求的接口将会被从前递表匹配项上移除，如果前递表匹配项此时仍不为空，则对于其所记录的每一个接口发送该数据请求。并以收到请求的接口，创建一个新的待应答表项。

如果三者都没有匹配的表项，则丢弃该数据请求，因为收到请求的路由节点既没有满足要求的数据，也不知道该向哪里前递已获取满足要求的数据。

数据应答包的处理要相对简单。由于数据应答包不经过路由，而是直接根据待应答表进行下行，在收到数据应答时，首先进行名称的最长前缀匹配。如果数据缓存有匹配项，则收到了重复的数据，予以丢弃。如果前递表发现了满足项，则说明待应答表中没有满足项，说明数据没有请求者，是不需要的，予以丢弃。如果带应答表中发现满足项，则(可选的)进行数据核实和写入数据缓存，之后根据待应答表的满足项移除收到数据应答的接口后，对满足项中的其它接口进行下行。

不同于IP先入先出的缓存模型，CCN缓存模型允许整个网络的节点缓存实现透明缓存。所有节点可以根据自己的能力和策略进行缓存。

通过数据请求指定的多点数据回取的特点使得CCN在变化快速的环境下依然可以灵活应用。任何处在多个网络中的节点可以作为其所处的多个网络之间的缓存和路由。利用其缓存，一个移动的节点可以作为多个彼此不相连区域进行连接的媒介，或者为不连贯的链路提供延迟的连接。因此，CCN的传输是允许干扰的(Disruption Tolerant Networking[11])。数据请求/应答的交互在仅有局域网连接时也可以正常工作，比如，两个处于同一自组织无线网的同事可以在二人没有公司局域网或广域网连接的情况下共享文件。

传输

CCN传输是基于不可靠的包交互设计的，其包括对高度动态的连接性的移动和普及计算。因此，数据请求、应答都可能在传输中丢包或是损坏，或者请求的数据会短时不可用。为了提供可靠的传输，一段时间内没有得到满足的CCN数据请求必须进行重传。不同于TCP，CCN发送方是无状态的。因而如果请求者依然希望得到数据，数据请求者需为重传请求负责。接收方策略层将为某一特定接口的数据请求重传进行负责(因为它会得到该接口上行数据请求超时的通知)，同时，接收方也应决定向多少、哪些可用的连接发送请求，允许多少未收到应答的请求存在，和不同数据请求之间的优先级等等。

下层的数据包交互网络和CCN的多点交互可能会产生多个同样的数据包。根据上述逻辑，同样的数据应答包会被节点丢弃。虽然数据应答不会在CCN的结构中形成环路，数据请求是可能形成环路的，并且使其看起来像是有数据请求存在，实际没有数据请求存在。为了检测和避免数据请求的环路，数据请求应包含一个随机生成的随机数字段(nonce)，多个接口在收到相同随机数的数据请求时也可以丢弃对应的数据请求。

CCN请求的流量控制和序号和TCPACK包是类似的，二者将在后面继续讨论。另外，一个节点必然可以看到其请求带回的应答，且反应时间和速度可以直接测量并用于决定对于某些前缀满足请求的最好方式的特性将在第3部分讨论。

可靠性和流量控制

一个数据请求最多可以带回一个数据应答。这项原则为网络维持请求、应答的平衡。从而使得连接速度不同的机器更有效的交互。这与TCP的设计是类似的，只是可以重叠数据应答和请求。多个数据请求可以同时发出，取回的数据应答将满足第一个请求。在此用例中请求起到的作用类似TCP的滑动窗口。收方可以通过调整其发出兴趣的多少，来实时的控制窗口的大小。如此的流水线式的设计造成的影响在6.2中讨论。由于每个CCN包是独立命名的，在出现丢包时流水线不会停顿，这一点与TCP SACK的固有性是一致的。

在大型网络中，TCP会话端对端的性质意味着即使每个会话满足流量出入平衡，端对端之间的节点仍然会出现拥塞，由于多组会话可能都通过同一个中间节点。这样的拥塞导致的结果是延迟和丢包。对于此问题，TCP的解决方案是由断点实时设置其窗口大小，从而使得汇聚后的流量仍然低于拥塞出现的流量。这样的拥塞控制本质上是由TCP会话的端对端特性导致的。而在CCN中，不同的是每个点都满足数据请求、数据应答的出入平衡。因为CCN流量控制是每一跳都满足的出入平衡，所以不需要为了防止某一中间节点出现拥塞而另外设计方案。这一点和利用相邻节点的反馈来维持每一跳的流量平衡的方案是不同的。CCN中取代链路间的先入先出队列缓存的是最近最少未使用缓存，其可以起到解开每两跳间反馈控制的反馈环和欠阻尼震荡的作用。(这一点将在未来的论文中讨论)

排序

在端对端的TCP会话中，数据包是通过序号唯一识别的。CCN需要实现更复杂的策略，因为数据请求方可能会请求大块数据的某一特定部分，并且该部分将可能与多个别的请求者共享。数据的定位和共享数据通过部分上有实际意义，人可阅读的，树状名称实现，而不是通过某一当前会话中的序列号。尽管CCN名称显得丰富很多，它们对传输过程起到的作用和TCP ACK的序列号是相同的，即指定请求方需要请求的下一块数据。

在阐述如何描述下一块数据前，我们先更详细的解释名称。如前面所提到的，名称是树状结构的，树中的每一个节点是名称中的一块。每一块由事先规定好的字节开始，名称本身对CCN的传输不起影响。名称需要体现上层协议的设计思路，从而对上层应用有意义。名称的传输除了要保留块结构外，并没有其它的要求。整型或是其它更复杂的数据类型可以直接使用其二进制编码进行传输，而不必转化为文本。出于隐私性的考虑，可以加密特定的名称块。为了表达方便，本文中名称的表示形式类似URI，如图4所示，用’/’来划分多个名称块。需要注意的是’/’本身并不是名称中的一部分，也不应出现在名称的二进制编码中。下面的例子介绍了应用层当前采用的命名大块数据的方式(利用一个版本标示符，\_v，编码为FD，加上一个整型的版本号；和一个分块标示符，\_a，编码为00，加上一个整型的块编号。这样的块编号的含义可以是大块数据中某一部分的帧编号；某一特定的字节编号；或是视频传输中的某一帧的序号)。每个数据应答包名称的最后一个部分隐式包含了该包的SHA256哈希值。数据请求包可以准确指定需求的哪一块数据，但是在很多用例中下一块数据的全名是不清楚的。请求者可以在名称中通过指定下一块数据和上一块在名称上的关系，来请求下一块数据。这一点的原因是CCN名称树是完全有序的，同一父节点的子节点按照字典序排列，前一或者后一这样的关系可以被完全不了解名称语义的中间节点无二意的理解。

例如，图5是图4种名称树的一部分。一个需要显示最新一帧画面的应用可以通过发出/parc.com/videos/WidgetA.mpg *RightMostChild*的数据请求来完成图中高亮的遍历方式，从而取回该视频中第一块的第二版。在取回这一块后，下一块的取回可以通过“本块最左的右兄”名称标示符，或是根据应用设计的名称空间，计算出名称\_s1。

如此例中所示，数据命名的分块是为了利用名称树的依据相对关系遍历的特点。尽管命名规则本身不是CCN传输的一部分，它们却是应用开发中最为重要的。我们希望开发出一系列的可重用的命名空间开发规范，并在共享的函数库中进行实现和标准化，从而为应用开发者抽象出更高层的库支持，例如基于CCN的文件传输，视频流传输等。

综上，数据请求包提供了整个CCN网络由标示符限制的查询功能，该功能使得请求者可以方便的制定下一块请求的数据。在此受到篇幅限制，我们对当前设计的标示符不予讨论。在未来的实现中，不仅可以对应答方的资源做相对关系的限制，还将可以除去特定名称对应的名称子块。同时我们还在开发高层的名称发现策略，从而使得用户可以在大的名称树中不请求数据本身的情况下浏览名称树。

丰富的连接性，流动性和策略

当今的计算机往往有多个同时连接的网络接口，并且有更强的流动性。由于IP的前递是限制在每个接口的生成树上的，使得IP较难同时利用多于一个的网络接口，或是适应流动性带来的快速的网络接口变化。CCN包不会出现循环，所以CCN可以充分利用同时存在的多个网络连接。CCN针对数据，而不是端点