清 华 大 学

综 合 论 文 训 练

题目：基于命名机制的轻量级分布式计算服务网络

系　　别：自动化系

专　　业：自动化

姓　　名：黎江源

指导教师：曹军威　研究员

2014 年 6 月 4 日

中文摘要

近年来随着网络技术在各方面上的迅速发展，人们对着网络的需求也在逐渐的改变，用户的整体需求更多的转向了对网络中服务的需要，而当初以端对端传输数据为目的设计的 TCP/IP 体系可以说对于当前的需求来说已经有些过时而无法更好的适应。随着存储与计算的成本大幅度降低而传输的成本却没有太大的变化，整体网络上的数据与信息呈现爆炸性的增长，进入大数据时代后每十八个月整个网络上的数据就会翻一番，在这种信息溢出的时代已经没有太多人关注数据究竟是来自哪个位置了，相反的更关心的是数据与服务的具体内容。此外 IP 机制自身也出现了 IPV4 地址枯竭的重大问题，在 IPV6 目前仍未发展成熟的情况下，想要直接过渡到 IPV6 在成本与性能上都是无法接受的，于是目前就存在有提倡对 IPV4 进行改进完善使其平滑过渡到 IPV6 的渐进派与发展设计新型网络结构以适应当前与未来需求的革命派。

在此情况下，众多新兴的网络体系一一涌现，希望能够在未来取代或者辅助当前传统的 TCP/IP 体系，而 NSN (Named Service Networking)服务命名网络正是其中一员。NSN 网络是基于数据命名网络 NDN (Named Data Networking)发展而出的新型基于数据与服务内容而非关注地址的网络体系，目标是在网络层面上解决远程服务调用的问题，发挥自身独特安全机制带来的强健的安全性能与缓存内容再利用的能力，致力于解决当前传统网络面临的地址枯竭、断续连接、网络配置等方面的诸多难题。

本文首先介绍了 NSN 网络出现前的总体背景与前身 NDN 网络的具体理念、原理和结构。之后具体介绍了 NSN 网络的概念与系统架构，还举出了 NSN 用于具体系统实现的用例。全文剩余部分则主要介绍基于 NSN 网络完成的楼宇管理系统与分布式计算系统的结构框架与实现方案，还对系统性能完成了评估对比。

关键词：服务命名网络；分布式计算；数据命名

ABSTRACT

With the rapid development of network in recent years, people's needs of the network is aslo changing , the overall needs of users turned to the need for more network services , TCP / IP system which was designed for end to end data transfer can be said for the current demand is already somewhat outdated and can not be a better fit. With the reduction in the cost of storage and computing costs while transmission has not changed much , the data and information on the explosive growth of the network , the data on the network doubles every eighteen months ,in this age has not too many people concern about where the data come from , but more concerned about the content of the data and services. IP mechanism also appeared problems like IPV4 address depletion , while development of IPV6 is not yet mature, the direct transition to IPV6 in terms of cost and performance is unacceptable , there is currently advocating that improve IPV4 to smoothly develop to IPV6 while others want new network architecture to meet with future needs .

In this case , many emerging network system appear , hoping to replace or assist the current traditional TCP / IP system, while NSN (Named Service Networking) is one of them. NSN is based on the NDN (Named Data Networking) which is a new content -based network system rather than concerns address , the goal is to solve the problem of remote service calls at the network level , to play with unique security mechanisms to secure robust performance and cache content reuse , committed to solve traditional network facing problems,like address depletion, intermittent connectivity, network configuration and many other aspects .

This paper describes the general background before NSN network appears and the principles of the NDN network. After Intro NSN network's system architecture , also mention use cases of NSN system implementation . The remainder introduces the structural framework and implementation plan of building management systems and distributed computing systems and their performence .

Keywords： Named Service Networking; distributed computing;Named data

目录

第1章 序言

1.1 NDN 网络产生背景

近年来随着网络技术在各方面上的迅速发展，人们对着网络的需求也在逐渐的改变，最初网络是设计出来在半固定的节点之间传输数据，所以采用了基于地址的的连接方式。沿用至今的这套 Internet 协议与结构都是于 19 世纪 60 至 70 年代间制定的，当时的主要目的是解决多台计算机之间共享数据的高成本问题，采用的模型是使需要交换数据的两台计算机之间建立起直接的连接。之后便发展成今日最常用的基于 IP 地址建立 TCP 连接的连接方式，这种网络结构关注的是两台相关计算机在哪，并依靠 IP 进行连接。

然而经过之后 50 余年的发展与变迁，日渐降低的网络与储存成本使得网络上传递的数据量呈现爆炸式的增长，而进入大数据时代后数据总量更是每 18 个月全球数据就会翻一倍，仅 2012 年两天产生的数据就相当于 2003 年之前全球所有数据之和。在这种信息溢出的年代人们更关注数据内容本身而不是数据来自哪里，然而网络的连接方式却还在关注数据的位置，可以说这种连接方式已经落后于时代的需求[7]。

此外 IP 机制本身也出现了严重的问题，当初设定 IPV4 时没有预料到网络发展速度如此迅猛，在当时看来约 43 亿的地址接近无穷无尽，然而到仅仅几十年后的今天根本不够分配，世界各地都开始出现 IPV4 地址枯竭的问题。目前呼声最高的解决方案是采用 IPV6 来代替 IPV4，但其间的过渡却没有民众想象中那么简单顺利，首先 IPV6 本身尚未成熟存在许多问题与漏洞，然后现今的网络协议大多都是依托 IPV4 实现的而无法运用于 IPV6，要逐一进行更改将会是一项很大的工程，最后如果要全面启用 IPV6 则基础设施也得跟上，更换新型路由等设备的成本也是一笔难以忽视的支出。

面对这种局面目前主要有两种对未来网络走向的看法，一种是通过对 IPV4 协议本身进行补全与改善的方式使其平滑过渡到 IPV6 的渐进派，另一种则是希望彻底重新设计以信息为中心的新型网络以满足未来发展需求的革命派。而既然过渡到 IPV6 需要从协议到附属设备进行全面的更换，为何不趁此机会对网络的整体构造与连接方式进行改革使其更适应于当今时代的需求，现

在正处于一个新兴网络出现并替代已经有所落后的 TCP/IP 体系的最佳时机，于是基于内容的网络结构 NDN——数据命名网络也就应运而生。

1.2 NDN 网络的核心理念与基本结构

1.2.1 NDN 网络的核心理念

数据命名网络即 NDN(Named Data Networking)网络，前身是 2006 年由 Van Jacobson 等人提出的 CCN(Content Centric Networking)项目[1]，是新兴的用于替代当今 TCP/IP 体系网络的网络结构，与关注位置的传统网络最大区别便是其传输的关注方向，依靠面向传递的数据本身，以每个被命名的数据成为网络间的信息基础单元。与 TCP/IP 面向传输与会话不同，面向数据最大的好处便在于客户端不需要知道数据储存的具体位置，只需要根据数据的名字从 NDN 网络中直接获取即可，服务器也只用处理传输的数据本身，不需要处理相对复杂的网络关系，具体的寻址、转发等功能都是由 NDN 网络自身完成的，大量简化了相关的操作。

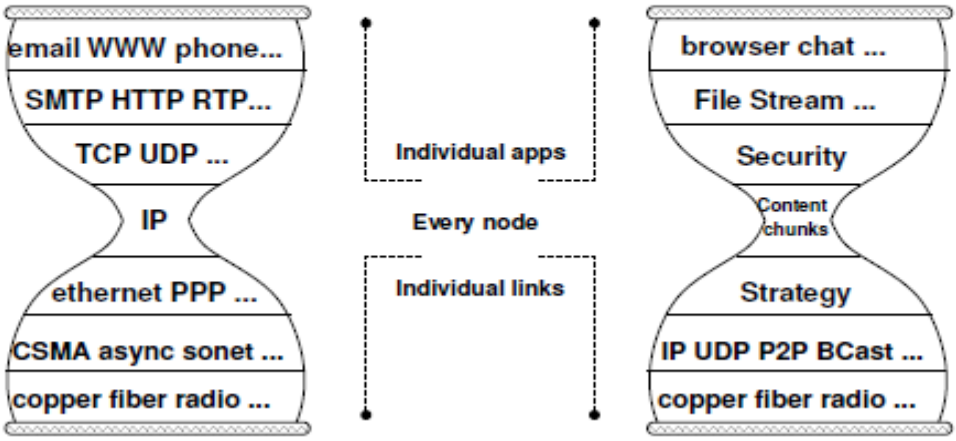

图 1. 1 TCP/IP 体系结构与 NDN 结构对比

目前传统 TCP/IP 体系网络是端对端通信模式，其中心是各台主机，所以其核心思想便集中于寻找端点地址，基于此实现其上的协议与应用，如图 1.1 中可以看出 TCP/IP 体系的沙漏模型中瓶颈处在于 IP 本身上面，而 IP 自己的缺陷导致了地址缺乏和传输拥堵等问题，虽然可以通过 P2P 和 CDN 等方式来解决，但这种结构带来的局限性导致无法进行进一步的创新与开发。

而 NDN 保留沙漏模型的同时将瓶颈转移到了数据块上来，避免了 IP 带来的种种问题与局限，可以基于 TCP、UDP、P2P 等多种方式进行传输并在其间运用多样的路由策略，对数据包本身进行加密而免去在各节点实施加密措施的麻烦并提高安全性能，实现端对端加密，对网络透明的安全系统。而传输与主机分离的机制也使得恶意数据包发送到指定目标变得十分困难，使得整体结构实现和管理更加容易，而且更加直接与灵活。

在网络拥堵方面由于不需要指定端对端的传输，从而可以将之前传递过的数据缓存在路由器中，之后接受到相同请求的时候直接从缓存中调出回应，既加快了响应速度又保证同一线路中不会被重复数据堵塞，由于每一个节点都可能获得需要的数据，可以实现每一跳的流量平衡。这种结构让 NDN 在内容分发、组播、移动等多种情况下都比 IP 单一最优路径的方法更加优秀，自然的实现了就近获取与负载均衡等功能，尤其在大规模的传输上有着很高的性能与可靠性。

1.2.2 NDN 网络架构与内容

NDN 网络包括了从接收到兴趣包起到返回相应的数据包为止其间的各部分内容，其结构可以主要划分为以下几部分模块：

1) 传输层：与传统网络的传输层很相似，用来于各个节点间传递数据，不过 NDN 可以与各种传输协议兼容，既可以依托于现有的 TCP/IP 体系来搭建，也可以使用其他协议实现。有区别的一点便是 NDN 作为中继节点的路由器带有一定的缓存，由于 NDN 结构的特点可以有效的利用这些缓存提升整体网络的性能 。
2) 策略层：这个是 NDN 与传统网络不同之处，由于不需要保证端到端的连接，NDN 的路由寻址方式可以十分灵活，可以实现逐跳寻找数据，而策略层正是决定在数据传输时使用何种寻址算法的部分。通过换用不同的策略，

可以有效的进行流量控制，针对性的对当前情况选取最适用的策略可以增强系统整体的稳定性，在大量连接或移动连接时实现性能优化。

3) 数据块：整个 NDN 体系的核心，每个数据块都事先被命名，其名字便代表与什么内容相关，主要分为兴趣包与数据包两种。兴趣包是客户端需要某种数据时发送进 NDN 网络的数据块，其名字代表它所寻找的数据名，当一路转发找到后数据包后数据包可以沿着其来路返回。数据包则是收到相应的请求后传入 NDN 网络由原路返回，突中经过节点时则在其缓存中留下相同的内容，以便之后相同内容调取。

4) 安全层：抛弃传统端对端的安全措施，对数据块本身进行加密，使用秘钥进行签发后接收方可以判断数据块内容是否与其签名一致，如果有不同之处就说明数据被改动或是假数据包而弃用。还可以通过签名还原发送方的证书，以此确认对方身份，可以依此实现 ACL(Access Control List)访问控制列表，来保证只有指定身份的命令才执行。

1.3 文章内容及目标

本文主要关注基于 NDN 网络实现的分布式计算等系统开发中的一系列问题以及解决这些问题的具体实现或想法，大致可以分为以下几点：

1) 系统搭建：本次主要实现了楼宇管理系统与分布式计算系统两种类型的系统，在实现时涉及到了安全问题，传输问题等多方面的因素，使用 NDN 自带的 keychain 系统实现了 ACL 保证了安全性能。

2) 命名服务：基于与 NDN 同样想法而实现的服务命名网络 NSN(Named Service Networking)，通过发送特定名字前缀的兴趣包配合名字中的参数而调用相应的服务，类似 Web Service 的运作模式。

3) 实验对比：对计算系统中 NDN 几项特点带来的性能上的改变进行实验对比，整理 NFD 平台丢包率与数据发送间隔并推测丢包原因，测试节点有无缓存在速度上的差距。

1.4 文章结构安排

全文第一章主要是对 NDN 网络产生的背景、其基本结构与思想，进行了概述，此外还说明了本文主要内容与结构，简略说明了分布计算系统在实现过程中碰到问题与解决手段。

第二章则是对本次系统所基于的 NDN 网络平台进行一个全面的讲解，介绍其具体实现与结构，在此网络中传输数据的原理，数据块的实际结构，确保安全性能的机制等方面都将一一分析。

第三章则是此次实现的服务命名网络 NSN(Named Service Networking)，将各项服务加以命名后可以如 NDN 中的数据一般寻找与调用，以及作为其应用的楼宇管理系统与分布式计算系统，解释 NSN 如何继承了 NDN 特性与优点，又有何不同之处，并以两个系统作为实际应用范例。

第四章是基于 NSN 的楼宇控制系统的概述，描述如何将 NSN 的特性融入到控制系统中，使用分层名字结构使得整体楼宇管理模块化，应用 NSN 的控制系统与传统的控制系统间的区别，以及实现时使用的策略与构筑途中遇到的问题和解决方法。

第五章是重点实现的分布式计算系统的设计与实现，在过程中碰到的安全性问题的解决方法，平台自带的各个寻路策略及自己编写的策略使用时的表现与差异之处。还对系统进行了一系列性能分析，例如使用分布式计算系统与参考过具有相似特性的 Web Service 的优劣之处对比，对平台丢包率与发送间隔关系的分析，测试 NDN 缓存为传输速度带来的优化程度。

第2章 NDN 网络概述

2.1 NDN 节点模型

NDN 传递的数据块只有兴趣包(Interest)和数据包(data)两种，客户端将兴趣包向所有可行的方向广播出去以寻找需要的数据内容，任何接收到该兴趣包且拥有对应数据包的节点都可以由一个数据包进行回应，数据包当且仅当回应兴趣包时才会发送并将此兴趣包消去使其不再继续传递。

判断数据包与兴趣包是否相符的方法是对比兴趣包中的内容名字是否是数据包中内容名字的前缀，因为 NDN 中兴趣包与数据包的名字都是采取的典型分层结构，所以如果两者前缀相符则说明数据包处于兴趣包的姓名子树中。在目前的传统 IP 体系中也是使用这种方法来解决网络、子网、主机间的结构，多年来的经验表明这个结构能够支持有效的分布式分层路由寻址与转发方式，同时也能支持快速查询。

当发布者接收到一个新的符合前缀的兴趣包时，如果该内容事先不存在的话可以现场制作出对应内容的数据包并回应，这种对动态名字的支持使得 NDN 可以同时支持静态缓存与动态内容，能做到这一点在现代网络中是十分重要的，因为目前网络上对动态内容的需求正日益增长。

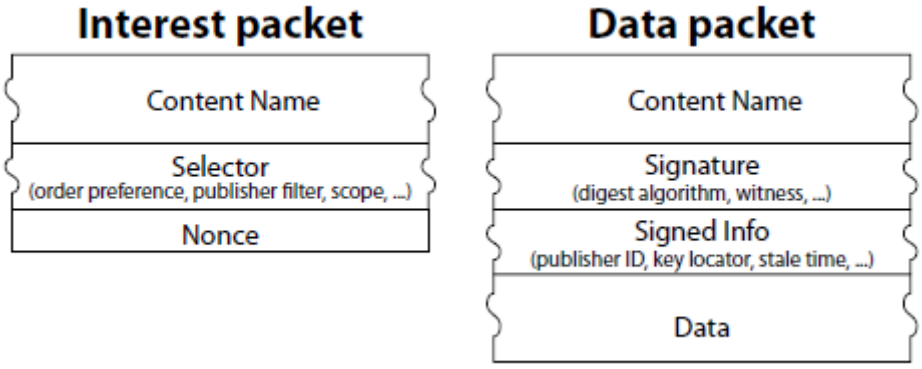

图 2. 1 兴趣包（Interest）与数据包（data）结构

从图 2.1 中可以看到两种信息包的结构，兴趣包由名字，选择器，随机数三部分构成，名字决定了要寻找的数据内容，选择器可以实现设置发送顺序、数据发布者过滤等功能，随机数则是防止不同客户端的相同内容的兴趣包产生重复问题。数据包则由名字，数字签名，签发内容，数据内容组成，名字用于与兴趣包匹配同时表明数据内容，数字签名用于验证数据包内容是否丢失或被更改，签发内容包含发布者 ID、发送时间等可以调用查看的信息，数据内容部分则是客户端所请求的信息。

NDN 网络节点的运作模式与传统网络 IP 节点十分相似，当节点监听接收到一个信息包后会对其名字进行最长匹配查询，根据查询到的结果决定下一步的操作。节点内部主要由 3 个模块所组成：

1) FIB(Forwarding Information Base):FIB 即转发表，与 IP 节点中的转发表基本上没有太大区别，主要用于将节点接收到的兴趣包转发向可能存在匹配数据的方向。在 NDN 中转发并不是依照传统网络中的生成树的方式，而是可以允许多数据源同时并行接受查找的模式。
2) Content Store:这个模块即 NDN 网络特点中的缓存部分，在传统 IP 节点中也有缓存区域，两者不同之处在于应用的替换策略。由于 IP 体系的端对端传输方式，处于缓存之中的数据是无法被二次利用的，所以当请求的数据已经经过此 IP 节点后缓存中的这部分数据已经没有任何存储价值，所以 IP 节点应用的是 MRU(Most Recently Used)替换，即当缓存区满后先替换最近使用次数最多的数据。而 NDN 的缓存内容是可以多次利用于回复对相同数据的请求的，所以与 IP 相反应用了 LRU(Least Recently Used)或 LFU(Least Frequently Used)的替换方式，尽量替换掉未使用时间最长或使用频率最低的数据，这样能够最大化再利用缓存数据的可能性同时降低对上流数据流量的压力，还能减少下流客户端接收的延迟。
3) PIT(Pending Interest Table):PIT 表记录了节点接收到的兴趣包的发送轨迹，包括兴趣包从何而来，节点又将其转发向了哪些数据源，根据 PIT 表中的记录之后发还的数据可以一步步沿原路发回目标请求者。NDN 中只有兴趣包在转发时需要进行路由寻址，每个节点中 PIT 表的作用就好比一条条“面包屑”的轨迹，让符合的数据包能够直接一路发送向目标。

当 NDN 节点接收到兴趣包时具体流程如下：

a) 对兴趣包名字作最长匹配查询，PIT 匹配处于 FIB 匹配之前
b) 如果缓存区域存在匹配数据，向兴趣包发送来的方向发送数据包并弃置该兴趣包(视为该数据包消费了兴趣包)
c) 否则对 PIT 进行匹配，如果之前 PIT 已经存在与此兴趣包相同请求的记录，将此兴趣包发送方向加入 PIT 中该条目的请求者名单中，并弃置该兴趣包(相同的兴趣包之前已经发送，当之前请求的数据返回时也会向这个请求方向发送一份数据)
d) 否则对 FIB 进行匹配，如果存在匹配的兴趣包过来方向以外的 FIB 入口则将此兴趣包向这些方向进行转发，在 PIT 中创建一条新条目记录该兴趣包过来与发送的方向。
e) 如果以上都没有与此兴趣包匹配的部分，则将兴趣包弃置(该节点没有任何与请求数据有关的内容也不知道该去什么地方寻找该请求数据)

当 NDN 节点接收到数据包时流程就简单很多，数据包只需要跟随 PIT 表记录连锁组成的路径一直前进即可：

a) 在数据包到达后对名字作最长匹配查询，PIT 匹配处于 FIB 匹配之前
b) 如果缓存区域存在匹配数据，说明之前已经有相同内容数据包经过产生重复，弃置该数据包。
c) 如 PIT 存在匹配说明这是之前请求的数据包，对数据包签名进行验证确定其内容未更改或丢失后将其加入缓存区域，并向 PIT 中所有请求方向发送该数据包。
d) 如果 FIB 出现匹配说明 PIT 表中不存在该数据包的匹配项，这表明该数据包是未经请求的，将其进行弃置。

之前提到 NDN 网络中最具特色的一点正是其每个节点都具有缓存部分，使得节点中存储内容可以同时被多次复用，由于其天然具有的多点数据检索特质带来的独立资源可用性与策略，整个 NDN 网络都具有在高度动态环境中通讯的稳定性。利用这种特质能实现许多传统网络不能或成本过高的功能，例如使用移动端节点作为网络中介串联起相互不连通的区域，每个节点都可以视作一个带缓存的路由器，以及在断续的连接中保证延时连通性等。

2.2 传输与策略

网络传输需要考虑的一大问题就是当出现信息包丢失、损坏或目标不可用等问题时的处理方案，尤其 NDN 正如前面章节所说，设计时是以能在高度动态的移动端连接等不可靠的信息传递环境下运作的，更需考虑如何实现重传等方面的问题。

当 NDN 网络中兴趣包发送出去后如果一定时间内仍未被满足就需要重传，但 NDN 的发送方是无状态的，最初的请求者如果超时后仍然需要这份数据就得负起重发兴趣包的责任。这项功能一般由网络中的策略层来完成，策略层会决定在那些接口重新发送、最多允许多少未满足兴趣包、不同兴趣包之间补发的优先度等等。

另外一大问题就是如何避免出现网络传输中的重复问题，NDN 的多点分布式查询更是容易导致同一节点接收到重复信息。当节点接收到重复的数据包时直接弃置掉，因为之前接收到时已经储存于缓存之中，而数据包发送时由于跟随 PIT 表路径是不会产生循环问题的，兴趣包有可能产生循环，使节点误认为存在一个虚拟的请求者，为了避免这一点兴趣包在末尾加上了一个随机数，这样便可以判断是否同一个请求者发出从而弃置重复信息。

2.2.1 流量控制

TCP 固定的端对端通讯导致当网络规模较大时在发送方与接收方两点外很容易产生拥塞，即使每个对话都是稳定的数据流也难以避免，这就带来了延迟与丢包等各种负面影响，TCP 中解决的方法是在端点处动态调整接收窗口的大小，以此保证线路中的交通总流量低于产生拥塞的等级。

TCP 中需要进行拥塞控制是因为他是以端对端的形式而无法直接控制中间的节点流量，而 NDN 中所有通讯都是相邻节点之间传输，所有完全不用对他们的通讯平衡进行管理，线路间的流量平衡是用各条线路两端的节点自行维护的，不需要进行额外的操作来进行通讯间的拥塞控制。

在端对端的流量控制中使用相邻节点间的反馈压力来调节持续数据流的传输，但 NDN 节点中缓存并没有先进先出的规则，相反的通过 LRU 或 LFU 的替换方式与弃置重复数据包的方法成功的将数据传输从连续数据流分割开来，实现了端对端反馈控制的解耦化并抑制了震荡的产生。

2.2.2 序列编号

TCP 传输数据时因为是使用的连续数据流，所以可以通过简单的将每一个分割后的数据块加上序号以用于排序的功能，而 NDN 不能通过序号来解决这一切，因为同一个数据包可能需要同时发送到多个请求者处，在不同的请求者看来这个数据包应有的序号都是不一样的。

所以 NDN 需要一种更高级的方式来展现传输数据的结构本身而不是仅仅只是在某次对话中进行排序，NDN 的解决方法是在数据的名字上加入深入的分层结构，使数据名字在对人有一定可读性的同时还能反映出其部分原本的结构。名字被分层意味着它是由一系列部分组成的，而每个部分都是任意数量的八位字节，这些部分组合而成的名字对上面更高的层能起到辨识等作用，但是对于传输层来说并不关心名字的内容是什么而是名字各部分的结构如何。所以数字或其他复杂内容可以直接通过二进制编码的形式来传输而不必转化为文字的形式，而如果有需要数据名还可以加密以保证别人无法通过数据包名称获取其中内容详情。

在平时为了记叙方便一般使用“/”来作为数据名字各部分间的分割符，例如/ndn/music/a.mp3/_v<timestamp>/_s1，但在真正的数据包中是没有这些分隔符的。上面举例的名字里说明了当前数据的版本号(_v 部分)与该数据块于整个数据中的分段(_s1 部分)，之后发送出去时还会与每个数据包一样在最后加上这个数据包的 SHA256 数字签名。

但知道数据完整的名字后兴趣包能很精确的指出需要哪一部分数据，但在大多数情况下往往是不知道数据全称的，这时候因为 NDN 姓名树是完全有序的所以可以通过单纯的 next 与 previous 关系指示 NDN 网络来获取数据的其他相应部分。

例如在之前的例子中，客户端可以发送‘/ndn/music/a.mp3 RightmostChild’获得这份数据的第一部分，接受第一部分数据后可以通过 LeftmostRightSibling 来获取之后的部分，只要应用能知道该数据分割的规则就能很轻易的获取数据的每一个部分[3]。

这种排序方法的好处是遵循这类命名规则的数据块组合可以通过应用上的设计充分的利用这种相关性取回的兴趣包特点，也可以通过树的部分遍历寻找出可用的资源。

2.2.3 连通性、移动性与策略

随着智能手机，掌上电脑等设备的兴起，今日的机器越来越多的具有多重网络接口并变得更加移动化，这对于只能使用生成树转发的 IP 体系来说无论是充分利用多重接口还是适应高速的设备移动化都不尽如人意。因为 NDN 信息包不会循环，所以 NDN 能够完整的利用设备上所有的网络接口，而由于关注的是数据而不是节点，所以 NDN 不需要将设备的 IP 地址这种第三层身份与MAC地址等第二层身份进行绑定，所以即使移动化设备导致连接快速变更，只要设备仍在 NDN 网络上就能保证相关的数据包与兴趣包进行交互。

前文中提到 CCN 通过 FIB 每个接口可以实现明确的多重连接，而可以满足所有接口各种情况的策略是不存在的，所以设计思路是每个 FIB 接口都包含一个计划，里面抽象出多种转发方式的机制进而决定如何发送兴趣包。机制中的动作可能包括发送给所有、发送给最好的、标记为最好的等，触发器可以定为兴趣包满足、兴趣包超时、接口关闭等而当事件发生时调用动作列表中的操作。

这些触发，动作与倾向等整合起来之后就组成了 NDN 中的策略层，每个 FIB 接口中的计划就是与这个接口前缀有关系的数据包的策略。目前的默认策略是将兴趣包发送想所有有广播能力的接口，如果没有回应再依次尝试剩下的接口。

2.3 路由寻址

发展到今日路由寻址算法早已出现了许多种类，其中既包括想法新颖的也有效率较高的，任何在 IP 机制下能应用良好的路由寻址机制在 NDN 下都应该能发挥出相同的水平，这是因为 IP 的转发机制实际上可以看做 NDN 转发机制加上多数据源、多目的地、避免循环等多方面上的限制后的一个子集，还运用着相同的最长匹配查询的名字机制。

NDN 可以说是实施路由寻址协议的绝佳媒介，大多数路由寻址协议的核心都与 NDN 以信息为中心的洪水式扩散机制十分相似，因为路由寻址的时候往往不得不在对象身份与位置都未知的时候在整个网络的拓扑结构中运作，而

NDN 能够提供一种强健的信息安全机制，所以在 NDN 上进行路由寻址传播的时候几乎可以实现全自动的路由设施安全保护[12]。

2.3.1 域内路由寻址

域内路由寻址协议是一种用于探索与描述同一个域中节点间的本地连通性的手段，也可以用于描述直接相连的数据源。这两个功能一个描述了图中的各个连接，另一个描述了图中每个节点能够提供什么，正好互补的组合成整个域内图的信息。

前面提到过 IP 与 NDN 的转发方式实际上几乎是相同的，都使用了基于前缀的最长匹配查询来寻找本地更靠近身份符合者的邻居，基于两个体系 FIB 间的相似性，可以较为方便将创建 IP 的 FIB 时使用的分布式路由寻址机制应用于创建 NDN 的 FIB。

但是因为 NDN 虽然与 IP 一样采用分层的名字形式，但 NDN 的前缀与 IP 相比有很大的差异，所以要套用 IP 的机制还得判断这种特别的前缀能否应用于一些特殊的路由寻址协议。幸运的是通过一种叫 TLV(Type Label Value)的通用格式可以直接在 IS-IS 与 OSPF 中描述相连的资源，这是因为 TLV 这种格式很适合于分布式的 NDN 内容前缀的描述。

这意味着部署了完整 NDN 转发机制的路由器可以不经修改的直接依附于现存的 IS-IS 或 OSPF 网络与路由器，NDN 路由器获知网络的物理拓扑结构并向拓扑结构中相邻点宣称自己的位置，使用 TLV 的格式依靠泛洪法散发出它的前缀，这显示出了 NDN 可以基于现存设施逐步扩展的性质。

当网络内出现多个对同一前缀的声明时 IP 与 NDN 的处理方式是完全不同的，IP 中特定的节点会将所有匹配的通信发送向其中一致的一个声明者，而 NDN 中所有节点会把所有匹配的兴趣包发送向所有声明者。这是由于两者前缀声明语意上的差距导致的，IP 中某个路由器声明一个前缀表明通过此节点可以到达所有拥有此前缀的主机，而 NDN 中路由器对前缀的声明表示通过此节点可以获得某些匹配这个前缀的数据。因为 IP 没有在内容层面上避免循环的方法，所以 IP 只能将其转发拓扑结构设计为无循环的(如各种树状结构)，所以遇到重复声明时只向其中一个声明者发送信息，而 NDN 的信息包天然的具有避免循环的性质，加上 NDN 前缀声明不代表这个节点具有所有与此前缀

匹配的内容，所以要向所有声明者发送兴趣包以确保能接触到该前缀的全部匹配数据。

上述 IP 与 NDN 间语意差异带来的区别却并不需要更改内部网关协议来使得两者相互适应，因为这是一种实现上的差异而不是协议上的差异，IP 由于其结构不得不根据前缀声明计算生成树，而 NDN 不需要考虑这一点，但是所有的计算都是在信息被接收使用的地方完成的，产生信息的地点不需要做任何计算，所以 IP 与 NDN 两者通过各自的协议都能获取到完整的信息

2.3.2 域间路由寻址

逐渐有一些 ISP(网络服务提供商)开始尝试使用 NDN，这些 ISP 开始在自己的域内布置带缓存的路由器以减少互传成本与降低用户平均延迟，因为用户们都是直接与 ISP 相连接，所以对他们来说不需要知道这些带缓存的路由器是如何通过在用户与 ISP 间连接上运行的服务发现协议来运作的，这些协议可以是基于 NDN 的注册、公告任播、DNS 约定、DNS SRV、SLP 等任意一种，这些方法的共同特点都是不需要任何域间的内容前缀分布。

而这种结构的关键在于如何沟通多个布置了带缓存的路由器但却被传统 ISP 分隔开来的域。如果一个布置了 NDN 的域中的节点希望获得另一个布置 NDN 域中的内容但是两个域之间没有带缓存的路由器进行连接，可以利用 DNS 查询前缀来定位目标带缓存路由器的 IP 地址，这样可以很成功的在两个节点间自动建立起 UDP 隧道的虚拟接口。

但是这个解决方法的问题在于它只当两个域间隔位于边缘的时候有效，如果两个域是被其他不支持 NDN 的 ISP 所连接，这时请求者就没有方法准确的获知数据源所在域的确切节点，这时请求者就只能无任何机制修饰的直接向目标域发送兴趣包，请求者域中的路由器就只对请求到的数据有益而对发送出域的内容没有任何帮助。这从某种程度上削弱了 NDN 一个主要的长期优势之处：保证接近内容分布树根部的通信与该内容被请求的频率独立开来，而这种通信正于今日变得越发普及。

目前修复这个问题的方式是将域间内容的前缀整合进边界网关协议中，这样边界网关协议在域间的路由寻址就会拥有类似内部网关协议 TLV 的机制，可以允许各个域如同之前的节点一样发布自己的内容前缀，这样每个域就可以建立一个类似域内内部网关协议里拓扑结构的地图，区别只在使用了

AS(Autonomous System)而不是网络前缀的级别。这样一张地图与内部网关协议中那张一样能起到了解哪些域提供哪些前缀的内容已经去到这些域路径上最近的支持 NDN 的域，也就能应用域内寻址使用的所有算法。

2.4 基于内容的安全性能

NDN 与传统网络关注节点与连接的安全机制完全不同，这是建立在基于内容的安全机制，保护并形容数据块自身让其自由传播，不用提供其传播过程中连接的安全性保证。

NDN 中所有数据包都是依靠其数字签名进行验证，而其中内容为了隐私进行加密保护，正是有这种安全机制才促成了 NDN 的动态数据缓存能力，如果想要能直接利用最近的缓存中数据，你得具有能验证这份数据究竟是否可靠的能力。

传统网络要验证一份数据是否可靠是取决于取得这份数据的位置与方式，即从什么主机获得与通过哪种管道获得的，所以只有直接对数据最原始的资源主机进行连接才值得信任。NDN 将安全机制嵌入数据本身内部而不是主机内，能够减少我们需要布置在网络中介上的安全措施，进而扩宽网络各节点间的分享渠道，之后部分便详细对 NDN 各部分安全机制进行解释。

2.4.1 内容验证

验证一个数据包时主要是鉴定其名字与内容间的绑定关系是否完整，当数据包签发时名字后端都会加上由数据包内容抽样得到的数字签名，这样便可以使发布者将任意的名字与内容进行绑定。许多之前的方法还需要名字进行自我认证以保证名字本身没有被更改，例如通过对名字的内容进行加密摘要等方法，而使用对用户或应用有意义的名字能直接并安全的加强数据包的可用性也更易于传输。如果没有这套机制的系统就只能通过一些间接基础设施来将对人有意义的命名转发为安全、不透明的自我认证名字，这种系统的安全性能就取决于这些间接基础设备的安全程度。

每个 NDN 数据包都是可公开验证的，数据包上的签名都是标准的公钥签名，包括数据传输端点在内的所有人都可以通过特定的密钥来验证数据包名字与内容之间的绑定关系是否完好。数据发布者可以从一系列固定设定中选择采

用何种签名算法，不同算法可用于满足不同的性能需求，例如某些算法使得签名长度最小，某些是产生签名时的计算延迟或计算成本最低。

每个签发出去的数据包中都包含有只有通过公钥才能找回的信息来验证自身，包括该公钥的加密摘要或指纹，可以将此看做发布者身份的一个标记，也就能由此快速的从本地缓存中检索这份密钥。信息中还包括密钥定位器，可以指出这份密钥可从何处获得，既可能是密钥本身也可能是一个能取得该密钥的 NDN 名字。

在最底层 NDN 内容的检验完全是语法上的检验，只是检验内容是否由它所支持的密钥所签发的，它属于的这份密钥没有任何现实意义，仅仅配合签名指出用户是否应该信任这一份数据的内容。

在对抗许多类型的的网络攻击时即使是最低程度的安全验证也能带来极大的好处，例如 NDN 用户可能通过名字或发布者来寻找获取数据，这时候很有可能连接到假的或恶意的数据的接口上去。这时如果路由器选择对所有数据包进行验证，而只转发允许的数据源的数据包，这样就可以避免不能信任源的数据进行传播，此外也可以随时动态的修改而检验更多的回应数据包以便检测出网络上的攻击。

2.4.2 信任管理

NDN 网络平时都是以对等网络的形式来传输数据，但它同时也能保证发布者与接受者端到端的安全性，NDN 的接受者必须测明接收到的数据是否可信赖或可接收，NDN 中采取的是基于前后关系的信赖模式。

例如有人可能需要得到某个法院认证的人签发的一份法律文件，而有一个网站只提供这个人签发的各种文档，这样这个网站作为发布者发布的数据就一定是这个人所签发的。这种方式比起试图制定一个能适应所有情况的方法来信任更加的简单与弹性化，可以将各个数据发布者标记为可信或者不可信以判断哪些回应的数据值得信任。

基于内容的安全性是在能够验证数据包名字与内容是否绑定的基础上建立的，而这也可以用于构造验证更高层可信性的机制。CCN 签发产生名字与内容间绑定的行为实际上是对该内容赋予证书，如果数据姓名与某些个人或组织有关联而内容是一份公钥，这样的结果实际上就是一份数字证书，这使得 NDN 能够很轻松的支持建立传统网络中对密钥信任的机制。如果认证后的内

容能够通过一个安全的连接关联到其他内容便能对其他内容进行认证，这使得可以通过较少数目的密钥来认证大量的内部连接的数据内容。

在应用级别的 NDN 客户端中必须完善传统密钥管理的机制，将所有公钥与个人或组织连接起来，这样就可以判断目标的现实身份，可以有助于判断哪些目标来源发送的数据内容是可以接受的。

NDN 中步骤如下：

1) 直接寻址某个只寻找验证一份特定数据必须的密钥的问题，解决这个问题得到的密钥也是 NDN 数据的另一种类型，其简单的命名规范使得其更容易被寻址找到。
2) 仅仅只把公钥作为 NDN 的内容可以迅速的为其生成一份证书，具体流程就是将一个由签发者认证的名字与这份公钥进行绑定，这证书之后就可以直接用来表示公钥间任意的信任关系。
3) NDN 没有提供一种能适应全部情况的信任机制，其中的信任是基于发布者与内容接受者的，适用于一个应用的信任机制不一定能适用于另外一个，用户既可以选择使用现有的机制如 PKI(Public Key Infrastructure)等，又可以选择自定义新的的更适合 NDN 网络的机制来应用。

在此之外还可以通过向 NDN 结构化的名字中添加一部分安全引用的概率，可以比作受信任的超链接或书签的形式，使得一份 NDN 数据的内容可以不光通过目标的名字关联上目标，还可以通过目标内容的加密签名或目标发布者的身份(密钥)来关联。

这种关联形式起到的最大的作用就是展现授权，也就是说如果连接 B1 上的发布者 A1 与连接 B2 的发布者 A2 关联，则 B1 会与所有通过 B2 与 A2 进行关联的内容产生关联。

这些关联在能表现传统授权形式的同时还能用于基于内容的信任网络，经个人签发后的一份数据可以有效的认证大量其他与此数据有关联的内容。例如用户如果决定信任一个网页，那么之后用户就会自动信任这个网页安全连接向的所有内容(图形，声音，广告等)，用户方不需要再支付任何额外的管理与配置开销，而这些关联上的内容只有在该网站的范围内才会被认为是有效值得信任的。

这样的机制下每一份用户收到的数据都可能成为一份潜在的证据用了判断与其相关联的内容的可信度，在许多可信的发布者都声明某份数据在某种语

境中值得信任时，用户也会更可能选择信任她。这样当攻击者入侵了一个发布者改变其声明试图伪造一份恶意数据时，由于更多的证明会指向正确的数据，这份恶意数据只会被标记为不可信的，随着这份数据被更多的发布者所支持为可信任的，攻击者就越来越难于成功伪造——除非他能够同时颠覆大多数可用的发布者提供的证据。

2.4.3 网络安全

NDN 的设计特性使其能够避免许多类别的网络攻击，对包括路由寻址与政策在内的所有信息进行验证能够阻止数据被更改或仿冒，而 NDN 只能与数据内容进行沟通却不能与主机沟通的模型使得很难发送恶意数据到指定的目标。攻击想要真正生效就得想办法阻断 NDN 服务，例如隐藏起合法安全的内容或使用大量伪造信息包淹没目标使其无法对请求作出回应。

为了确保客户端在数据有被仿冒威胁时能够真正得到需要的内容，客户端发送兴趣包时可以在其中添加约束，使得只有指定的发布者的内容才能与兴趣包进行匹配，适当的约束可以在网络效率与客户端得到内容的易用性上取得平衡。

客户端最低程度上得指出他想得到哪个发布者所签发的相应内容，或者指出签发了这个发布者使用密钥的密钥，这种方法可以间接的避免系统变得过于失去弹性，这就要求客户端必须先验的知道目标内容是由哪个特定的密钥所签发的。

为了阻止前面提到的大量不需要的恶意转发通信导致的堵塞，NDN 创建了多个相关机制。兴趣包与数据包之间的流量平衡能够阻止对本地连接进行数据泛洪攻击以试图强行阻断 NDN 服务的行为，这种流量平衡是以跳跃到跳跃间的形式实现的，只有与之前发向下流的兴趣包数量相同的请求的数据包可以发回上流去，无论这个节点收到了多少个数据包都不会改变向上转发的数量。到了聚合节点更是只有一个请求的数据包会发向请求的客户端，这样使得基于数据包的 DDoS(Distributed Denial of Service)攻击很显然的变得不可能。

与数据包不同的是任何一个客户端都可以由任意的速度制造兴趣包，所以理论上是可以发动基于兴趣包的泛洪攻击的，由分布式客户端产生大量的兴趣包以期望能够超过目标 NDN 网络认为匹配这些兴趣包的节点的带宽使其堵塞瘫痪。然而请求相同目标的兴趣包会在 NDN 路由器节点处被整合，只有一份兴趣包会发送向上游数据发布者，所有如果试图发动兴趣包泛洪攻击，攻击者

就得发送名字大前缀相同能指向目标发布者但名字后面部分完全不同的兴趣包以避免被整合。

NDN 路由寻址的两大特征能够较为简单的缓和这种攻击。

首先，如同数据包能够追寻其匹配的兴趣包的路径返回客户端一样，每个 NDN 中介节点也能够观察到每个兴趣包转发的方向，无论这个兴趣包有没有取回数据包。兴趣包泛洪攻击明显是不会有数据包回应的，可以设计一个简单的适应算法限制同时间内只有一定数量的兴趣包能够发送向同一前缀，如果兴趣包成功返回数据包则不计入其中。

其次，被攻击的域可以请求下流的路由器对目标前缀发送的兴趣包数量进行压制，如同 IP 体系中要求下流 IP 压制或封锁访问无用的地址一样，但是区别在于 NDN 能在内容命名空间中添加政策的能力使其支持语意上的选择控制。

第3章 Named Service Networking(NSN)概念与系统架构

3.1 NSN 概述

NSN (Named Service Networking)即服务命名网络，是于内容命名网络 NDN 的基础上发展而来的一种新型网络结构，继承了 NDN 放弃端对端通信转而关注数据内容的网路传输模式，以及其分布式数据源的机制。同时还参照 Web Service 的运作方式加入了将服务内容命名化的创新，使得各项服务可以如 NDN 中数据一般进行命名，客户端可以通过兴趣包进行调用与操作，既可以当做单纯的操作命令也可以是要求返回某些计算后的结果数据。

提出 NSN 的想法是由于目前互联网的用户更加关注的是服务本身而不仅仅是数据，无论是信息检索、云平台还是网盘存储等服务都随时有着大量的人有所需求，这样看来网络应该是一个服务池结构而不仅仅是单纯的数据传输。再加上计算与存储的价格日益下降而传输的成本却没有太大的波动，所以便考虑到可以以增加计算与存储内容的方式来减少传输的压力，同时进一步减少用户调用服务时的步骤，提升网络的服务质量。

面向服务的未来互联网体系结构与机制研究计划是一项与 NSN 理念有一定相似之处的项目，项目中提出了基于当前 TCP/IP 体系以服务标识作为沙漏模型的细腰部分，再依托标示出的服务来带动路由寻址与数据的传输。这种名为 SOPIA 网络的体系可以将网络转化为包含传输、储存与计算等各方面内容于一体的服务聚合体，还可以提供网络虚拟化功能与内在的安全系统，对 NDN 网络的相关开发很有启发性[10]。

此外面向服务的网络功能还有一个很好的例子——Web Service[11]，每个 Web Service 向外界只提供一系列服务接口，当有人调用这些服务并向接口中发送参数之后内部进行相应的运算与操作，最后得到结果后由调用的接口返还给请求的客户端。其优点在于目标不需要了解 Web Service 内部的实现机制，甚至可以当做一个需要联网的函数来使用，而且以 XML 语言为传递媒介可以实现跨平台、跨编程语言的调用。

而之后发现 NDN 中虽然可以直接访问各种数据的内容，但是没有一种类似的提供远程服务的方式，而目前的环境需要一种能在网络层解决服务问题的

方法，于是便想到如果借鉴 NDN 命名数据的方式，将其中数据的部分改为服务的名字，这样就可以在 NDN 中如同访问数据一般调用服务，所以将这种网络称作 NSN 服务命名网络，可以看做是 NDN 网络的一种新的可能的发展方向。

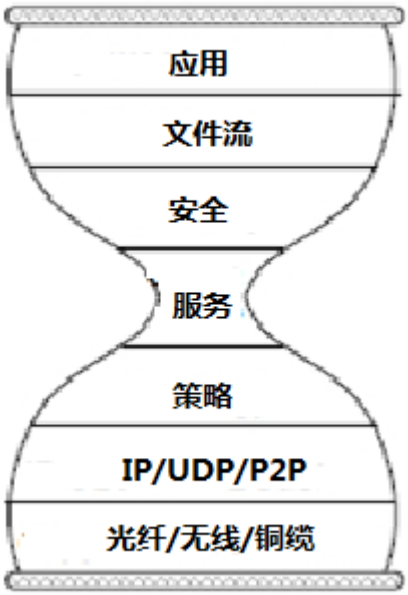

图 3. 1 NSN 网络沙漏结构示意图

图 3.1 展示的是 NSN 网络的沙漏结构，其中与 NDN 结构区别沙漏的细腰部分由数据层换做了服务层，服务层部分是将数据包中名字与提供的服务联系起来并提供参数等信息的描述，这部分即是对人可读的，同时也是一种对机器可读的服务描述，让系统能了解调用的服务名称、参数等等。服务的具体描述方法应该是一种服务提供方与请求方都了解的统一接口，其中有着具体的对兴趣包前缀每个部分对应的含义与参数格式等信息的规范，此描述方式需要可以既保证兴趣包前缀与现实意义的关联，又使其仍然是节点可以直接识别的而不需要经过间接转换的工序。

沙漏其余部分大致相同，服务层之上就是安全层，负责对数据本身进行加密以及对信息包安全性与权限的审查，无论是密钥间的相互签发与检验，还是信息包的加密解密与签发都是在这个层面内所完成的。

再上这是文件流层，负责将通过检验的信息包中的数据转化为文件流发送到真正需要这些数据的应用层，而应用层则是真正实现一个系统的层面，通过

调用其下各层的各类服务来实现种类繁多的功能，再将这些功能组合成真正具有实际使用意义的应用。

服务层往下是策略层，这个层次决定的是数据包在节点间具体的发送与接收，还有数据在传输途中的路由寻址方案，管理着接口间的关系以及节点内部 PIT 表、FIB 表的管理与匹配等环节。

再往下就是网络的连接方式了，可以分为物理连接方式与基于其上的各种传输协议。网络的物理连接方法可以依靠光纤、无线信号或铜电缆来实现物理上的相连，而传输协议则包括 TCP、UDP、P2P 等多种协议，NSN 网络的兼容性很强所以可以通过其中任意一种协议来完成。这些各个层面组合起来就构成了 NSN 网络的完整结构。

服务层的实现可以通过定义一种服务的发布方与需求方都了解的语言模板，例如 /repo/<commandverb>/<RepoCommandParameter>/<timestamp>/<random-value>/<SignatureInfo>/<SignatureValue>，其中每层可以相互嵌套，如 <RepoCommandParameter>中就包含有 Name、Selector 等子元素，这样就能通过兴趣包的名字来清楚的对调用的服务及其细节进行描述。而这套模板可以通过某些中介网站等方式公布出去，有需要的人就可以依照说明使用模板内容，从而使双方拥有统一的接口。

NSN 与 NDN 最大的区别就在于兴趣包的使用方式，NDN 中兴趣包名字部分仅仅表示了寻找的数据包的内容，而 NSN 中的兴趣包不光能实现 NDN 中访问数据的功能，还可以用于调用服务时传递参数以及作为不需要回应的命令。

NSN 中调用服务时兴趣包名字可以前半段是与该服务匹配的前缀，之后再额外加上几个名字部分作为参数，与 URL 中传递参数的方式比较相似，之后服务方接收到兴趣包后根据事先的约定从名字中指定的部分获得需要的参数之值，依此进行计算或操作再返回结果。

例如兴趣包名字可以取作/proc.com/ndn-caculator/plus/1/2，这里面 /proc.com/ndn-caculator 是服务的整体前缀，/plus 是具体调用的服务的名字，/1/2 则是之前调用的相加服务的两个参数。服务方接收到这个兴趣包之后通过整体前缀判断与自己匹配，从约定好的第三个部分获取服务名——这里便是相加服务，同理从约定的四、五部分获取加数与被加数两个参数，进行计算 1+2 后得到结果 3 并作为数据包内容返回。

3.2 系统节点构造与实现

NSN 系统节点结构大多数与 NDN 中节点的结构相同，路由器等中介节点上没有任何区别，区别只存在于发布者类节点上。

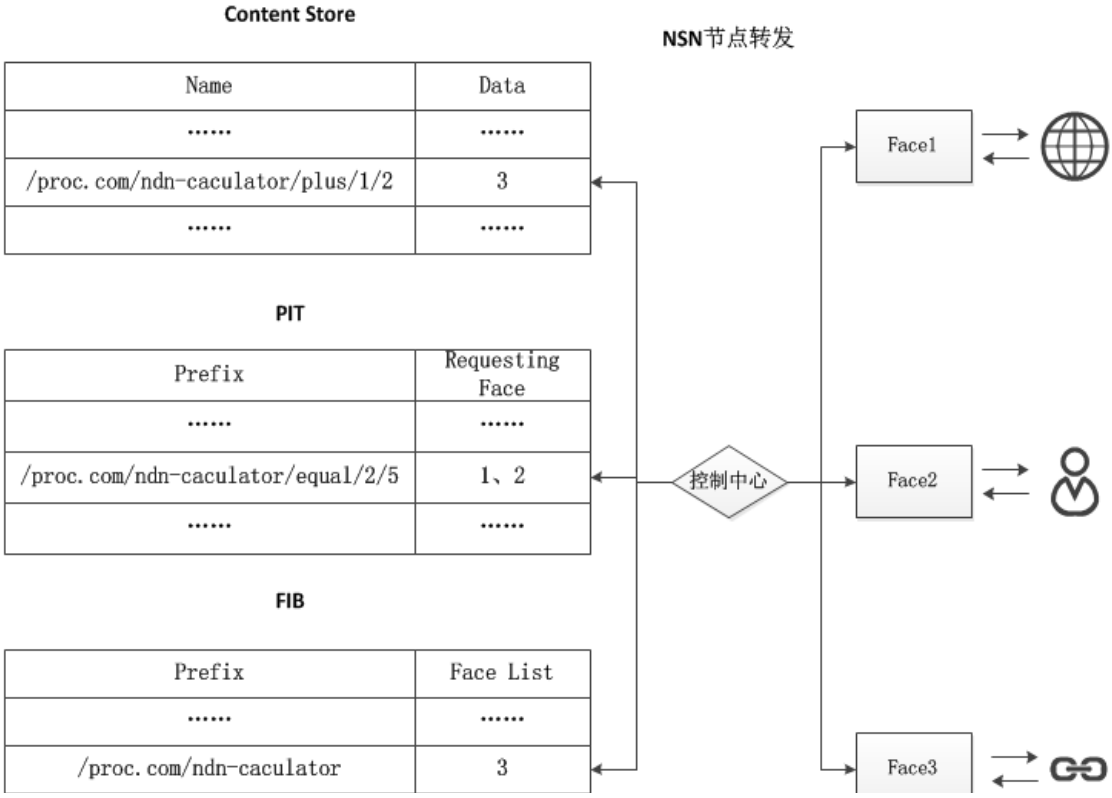

图 3.2 NSN 节点转发机制

图 3.2 中左边部分就是 NSN 转发节点的三大主要部分——Content store，PIT(Pending Interest Table)表与 FIB(Forwarding Information Base)转发表，右边则是节点向外提供的多个接口，这些接口与因特网、以太网等多个网络连接，有可能接到来自各种网络的请求。

从接口接收到信息包后首先是发送到节点的控制中心，这部分主要负责的就是对信息包进行各类匹配再决定对其进行的操作。

Content store 缓存部分为信息包最先进行匹配的环节，如果兴趣包存在此处的匹配，说明之前有过相同的请求并返回了相应的数据包，可以直接调用这

部分缓存内容作为数据包发回，这里需要注意的是有一些服务与具体时间相关，所以匹配前缀中会存在 TimeStamp 的部分以作为区别，这种情况下虽然缓存中可能存在相同服务的数据但也不能直接使用。而如果是数据包产生的匹配，则说明这个数据包是重复数据，缓存中已经具有相应备份所以可以直接弃置掉。

PIT 表匹配其次，兴趣包出现 PIT 表中的匹配说明有人申请了相同的服务且兴趣包之前通过了这个节点，目前正在等待数据源的回复，这时候就将接收的相同兴趣包弃置，在 PIT 表该条目的请求方中加入这个兴趣包来自的接口方向，这个操作就相当于将多个同样目标的兴趣包整合为一个，使得操作统一的同时又减少了对下游网络流量的压力。这时候调用服务应该注意的是兴趣包内的参数问题，比如两者虽然都调用了/ndn-caculator 的 equal 服务，但有一方的参数是 2 与 5，另一方是 3 与 3，这时候双方的结果显然是相反的，所以不能轻易的整合，需要扩大前缀匹配的范围将参数包括进去才行。数据包的 PIT 表匹配说明这个数据包正是之前通过此节点的兴趣包所请求的，应该加入缓存之中并向每一个请求接口发送一份数据包。

FIB 转发表是最后匹配的，兴趣包的 FIB 匹配意味着之前没有任何相同目标的兴趣包通过，且这个节点有路径能通向目标数据源，之后就应该在 PIT 表中新建条目记录兴趣包目标名字与来自的接口，再将兴趣包向除了前来的方向外所有可能存在目标数据的方向转发。如果数据包存在 FIB 匹配则说明 PIT 中没有相关的记录，这是未经请求的数据包从而被弃置。

NSN 中发布者节点存在两种，一种一般是最底层节点提供各种命名的数据，这类节点与 NDN 中没有太大差异；另一种则是提供被命名的服务的中间节点，虽然客户端可能是直接向这类节点请求服务，但这类节点还得向前一种底层节点获取数据，进行计算加工后才有结果返回，当然相应的也存在第一种与第二种节点的混合节点。

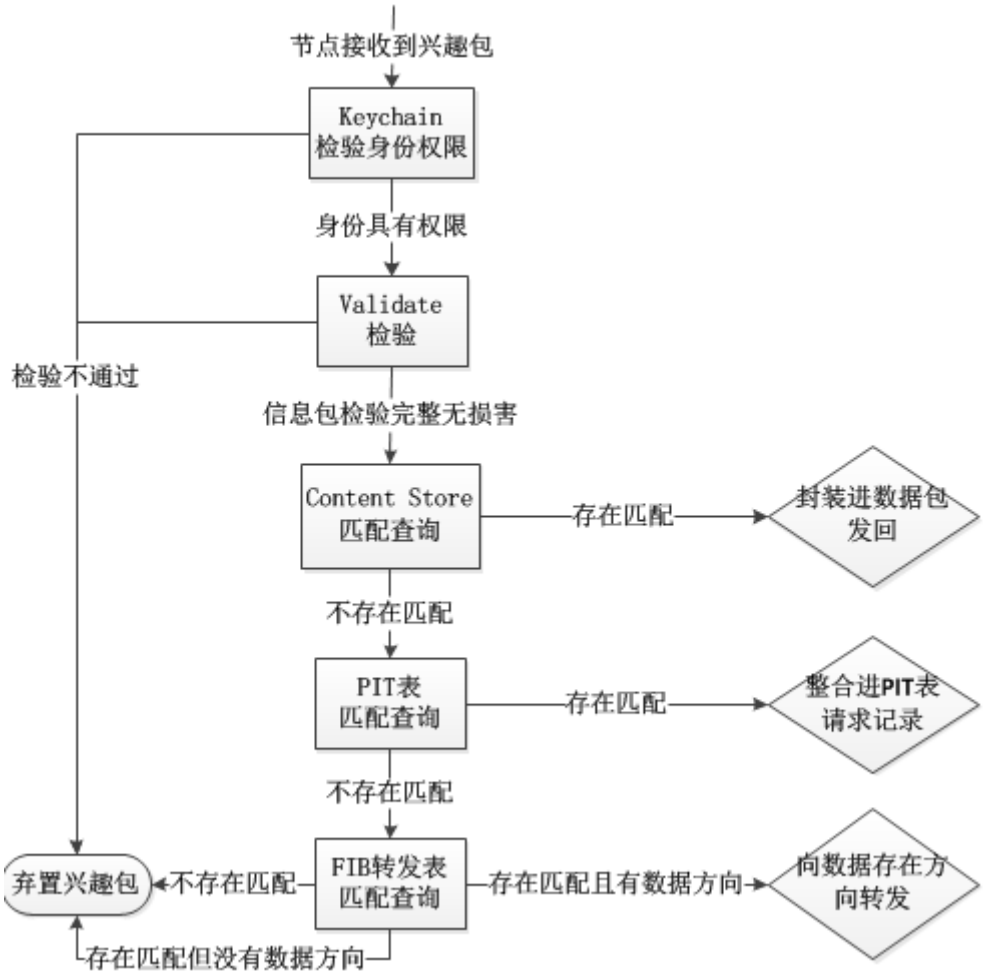

图 3.3　NSN 节点转发流程

当 NSN 发布者节点接收到兴趣包后，流程基本如下：

- 进行最长匹配查询，判断前缀是否存在匹配。
- 如前缀存在匹配，使用 keychain 函数进行检验判断请求者是否为存在权限操作者；

  前缀无匹配则弃置或转发。
- 如 keychain 判断请求者身份有权限操作，使用 validate 函数对信息包进行检测，通过之前 keychain 函数确定的身份获取相应的公钥与证书，对信息包名字后的数字签名进行验证以确定信息包的名字与内容未更改或丢失；

Keychain 判断无权限则弃置。

- Validate 函数判断信息包无更改与破损之后，直接根据前缀读取信息包名字的相应部位，这些部位包含的内容对应的意义都是事先约定好的，获取操作命令及其参数。
- 根据操作命令名执行相应的函数进行计算，代入参数后得到结果，其过程中可能需要执行向其他底层节点获取数据、或是对底层节点进行修改等操作。
- 如果接收到的是调用计算等服务的兴趣包，将结果打包进数据包中并原路发回；

  如果接收到的是修改数据等服务的兴趣包，实现其中的操作后便不用回复或者回复内容表示该命令操作已完成。

3.3 构造实现举例

3.3.1 楼宇管理系统

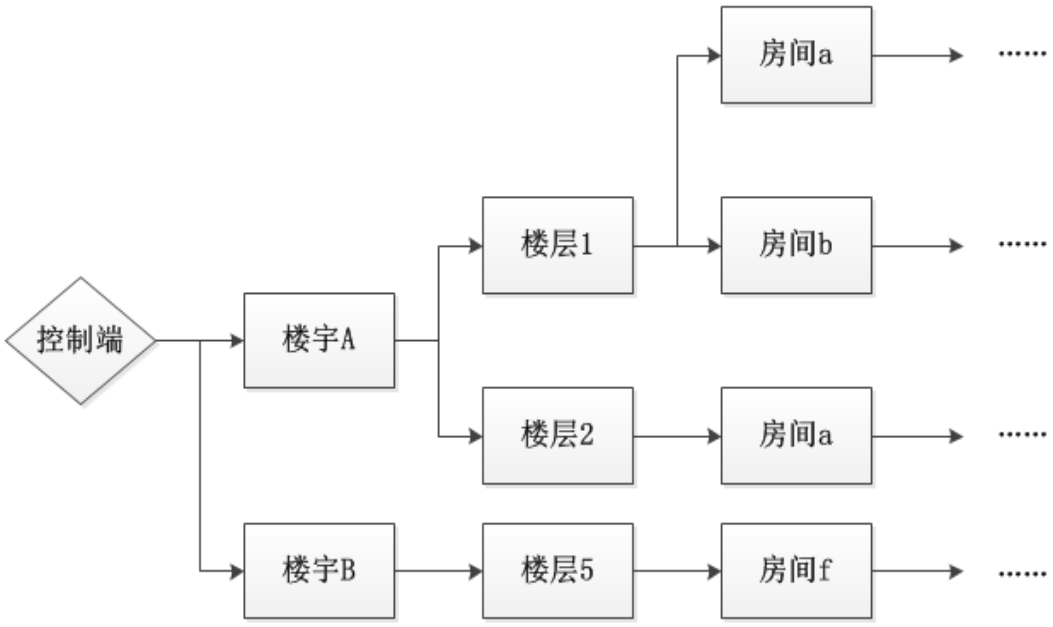

图 3. 4　楼宇管理系统结构示意图

图 3.4 中展现的是楼宇管理系统中节点组合间的结构，可以看到最顶端有个控制端起到监控与发布指令的作用，而下端的节点则是以层次化的结构依次连接，以第二层节点对应楼宇，第三层对应楼层，第四层对应房间等依次对照下去。

如此分层的好处就是给每一个节点赋予了现实中的意义，每个节点与现实部位一一对应也就意味着将这些现实部位都进行了 NSN 网络上的命名，可以很方便对需要的部分进行数值读取与操控。

实际采集数据的只有最底端的采集器 repo，例如每个房间内可能有多个 repo 分别采集水、电、温度等数据。当控制端需要知道某个房间数据时不需要对这些 repo 一个个进行访问，只需要向该房间对应的节点发出命令，房间节点如果没有数据就会自动向属于该房间的 repo 请求数据，得到后再打包发送给控制端。

同样如果控制端想知道某层楼、某栋楼的数据都只需直接向目标节点发送请求即可，这些节点都会自动向下一层节点请求数据并汇总发回，使得对楼宇各部分的数值监控变得相当直观与简单。通过发回的这些数值，控制端可以得出各项数值的实时监控并甚至实现反馈控制等功能。

3.3.2 分布式计算系统

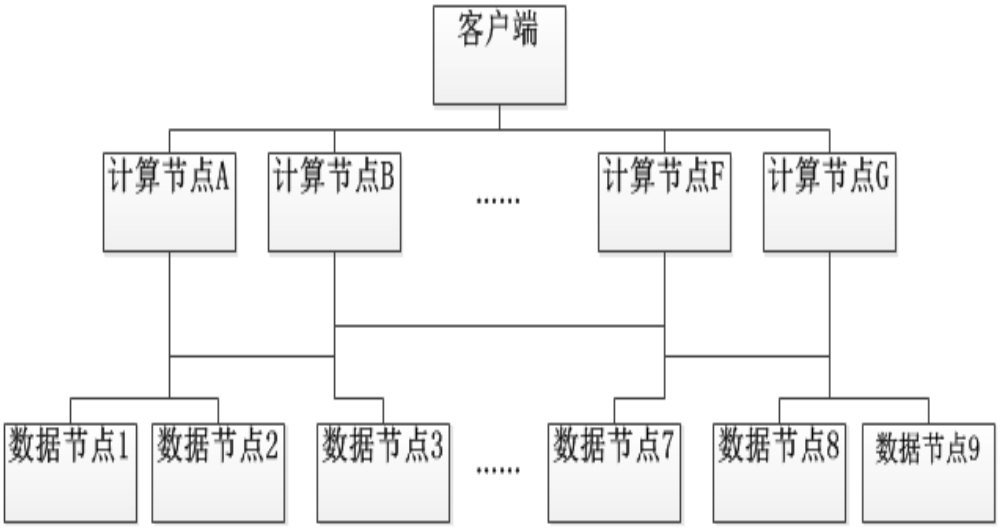

图 3. 5　分布式计算系统结构示意图

图 3.5 展示的是分布式计算系统的总体结构，可以看到最顶端是客户端，客户端负责将需要计算的问题拆分为一系列细微的问题，并依次发送到 NSN 网络中。第二层是多个计算节点，所声明的内容前缀完全相同，提供的服务也完全一致，设置多个计算节点的目的就是为了能将整体较为沉重的计算负载较为均匀的分摊到每个计算节点上去，充分利用所有计算节点的计算能力，这也正是分布式计算的精髓所在。

在客户端发送兴趣包的时候会对转发策略进行修改，使得发送的目标会在这些节点中按照一定规律分别发送，不采用 NDN 中将兴趣包广播到所有声明相同前缀的数据源处的原因是这样无法起到负载分摊的效果，如果广播的形式可能客户端最终只接收采用了 k 个数据包，但 n 个计算节点一共会返回 n*k 个数据包，每个节点都进行了 k 次运算，和使用一个计算节点没有任何区别之处。

第二层的计算节点和第三层的数据节点间是完全相互连通的，每个计算节点都能直接访问所有数据节点向其请求数据，这样所有计算节点所处的环境与

自身的条件都是完全一致的，保证每一个节点在不超过负载的情况下都能独立完成相关的任何计算任务。

使用 NSN 来实现分布式计算系统能提供很多相对与传统网络实现方式的优势。传统的 TCP/IP 体系来搭建即使是结构很简单的分布式计算系统也不是一件容易的工作，而 NSN 实现最大的好处之一就是实现相对方便简单，节点间不需要关注其他节点所处的地址，使得配置的难度骤降，同时这种非端对端直接连接方式让节点的增减都只需要直接连接或离开 NSN 网络即可，其他节点无需对此作出任何改动。

第4章 基于 NSN 的楼宇控制系统系统设计

4.1 楼宇控制系统概述

楼宇控制系统，即 BAS(Building Automation System)，其作用是对整栋楼宇内部各种设施设备的进行监管控制，例如实现控制照明、用电、供水、调温等功能。现代化楼宇控制系统可以通过计算机程序来实现整体的监控与反馈控制，以及对全局进行优化或设定时间规划，以集中管理的方式既节约了控制所需的人手，又可以依靠计算机的高效计算能力实时调整使得能够实现全面优化节约能量开销。

传统楼宇控制系统可以说主要由三大部分组成[2]：ICS(industrial control systems)提供了对设备数据的监控采集方面的功能，被称作智能电网的企业网络，还有就是 IOT(Internet of Things)物联网的运作，由物联网的广阔视野配合 ICS 的监控保证对整体各方面的了解与控制手段，而智能电网使得内部设备的功率与消耗达到优化的均衡状态。

控制系统里采集到的数据信息可以用于性能评估、成本计算、记录报告与错误检测等多个方面，而在本地系统的范围内也可以提供对设备的闭环反馈控制，将物理地址上分散的设备与区域在网络上整合控制，对其实现虚拟地址上的集中化。

实现信息化控制的好处首先在于快捷性，信息网络的传递速度使得监控人员足不出户即可掌控全局信息，大大减少了工作量与用时；其次在于很大一部分控制内容由软件接管，员工可以较为简单的进行操控从而降低培训成本与时间，还能减少维护员工可能出现失误导致整体设备产生问题的可能性；最后就是节能，快速计算与快速控制双管齐下，使得整体控制方案与具体实施都更接近最优化，减少各方面不需要的损耗，节能最多可接近一半。

由于一栋楼宇中的主要功能设备如电梯、空调等都是靠对设备的供电进行控制而当其运行时所处的状态与功率等都难以及时获取，处于一种开环的工作状态而无法对其进行有效的控制，既无法实现实时监控也不能有效调配资源。现在通过信息网络的实时传输使得可以将整体设备与监控中心间实现闭环回路的结构，各处底层监控设备获取设备工作状态数据后发送回控制中心，在中

心计算机上进行分析判断，得到的结构再返回各处设备进行反馈操作，无论这些设备数量再多分布再广也可以合理的在最小的成本下保证其运行的合理性与安全性。

4.2 基于 NSN 楼宇控制系统优势

之前的楼宇控制系统多基于 TCP/IP 的体系实现，这样的好处是体系结构熟悉、存在完善的规范与标准、有足够的经验保障，但相应的随着楼宇规模变大导致需要连接的节点数量增加，旧体系的缺点也就一一浮现：

- 内部寻址与网络配置：

  控制中心就算只是简单的想要获取某个最底层传感器节点的数据也需要知道对方的地址信息才能在多层结构中定位对方，这些信息包括且不限于虚拟局域网 ID、IP 地址、端口号码、子网地址等等。所有的地址信息都易于变化且与目标数据在系统中的作用毫无联系，对这么多设备与应用的各种信息进行管理变成一项十分繁杂且浪费时间的工作，而且系统规模越大出错的可能性也成倍增长，难以计数的节点与同样为数不少的应用间的交互使得配置难度过高。

- 网络中介设备：

  系统内部中介设备的作用就是缓和节点间寻址的压力，同时提供一个描述上的映射使底层网络地址能与应用间产生具体的联系。但这同时也添加了配置的压力，降低了整体网络的性能，所以中介设备并不适用于底层的 M2M 通讯，不能够完全解决网络寻址与应用逻辑的错误匹配问题。一些企业自定义的协议中会要求提供人类可读的名字与无意义的设备编号以通过这种间接的方式来适应协议的消息格式，配置时需要编制一个很长的每个设备人类可读的名字与其 IP 地址间的对应表单。

- 安全性能

  由于大多数楼宇控制系统中使用的自创协议都是在这种相对较为孤立的环境中产生的，所以对于外部攻击的防御性能是很弱的，对外来攻击者来说这种防御就和不存在没有太大区别。但楼宇控制系统又往往是关键基础设施中的一部分，意味这系统需要与外界网络乃至因特网进行连接，这就需要系统具有强健稳定的安全性能。即使在各个节点间的通道上都实现

安全措施可以办的，这也是一项成本很高的工程，小型设备由于节能控制等方面的原因很可能经常关闭与重启，每次重启过后就需要重新建立起与其相连节点间的安全连接，这既降低整体效率又提高了成本。

而楼宇内部设备地址的分布性正适合 NSN 网络发挥其特点，在应用层面与网络层面应用的相同的数据命名，寻址问题在整体结构中一个位置就可以完成，不需要各层分别进行也就实现了简化寻址过程的效果，使得相比起需要多个层面交互操作的 IP 体系来说网络配置工作得到了极大的简化。此外 NSN 提供对数据块内容自身的加密方式，这种不需要基于连接通道适合楼宇控制系统中部分节点会频繁周期性上下线的安全传输，系统可以设计基于加密的身份控制机制，能够增强系统整体的扩展性并避免 DDOS 等网络攻击方式，这些简单而有些的安全策略目前在 IP 中并不存在。

NSN 实现楼宇控制系统还有以下优势：

- 在遍布楼宇各处的传感器、设备与用户间建立起分层结构的对应名字结构，保证这些名字既能体现各自节点的性质，又能展现出各个节点间的关系的同时能不经转换的直接应用于数据传输。
- 与 IP 体系相比更简单，更强健，更具扩展性的对设备配置不间断的管理能力。
- 可以扩展的用户群体与相应的特权管理，以支持企业级别的整体部署与之后可能的修改。
- 不经过通信通道对数据包直接进行数字签名验证，可以确定此数据包的来源身份。
- 不用物理或逻辑上的隔离也可以实现的基于加密技术的私密数据传输控制。

从这些优势中可以看出，基于 NSN 来构造的楼宇控制系统无论在命名与寻址策略，还是更高层次的系统配置例如寻址与中介设备，以及适合于该控制系统特点的安全机制上都与传统 IP 系统相比有了长足的提高，这从一定程度上显示出了 NSN 网络的特性在实际应用时带来的优势。

4.3 楼宇控制系统实现

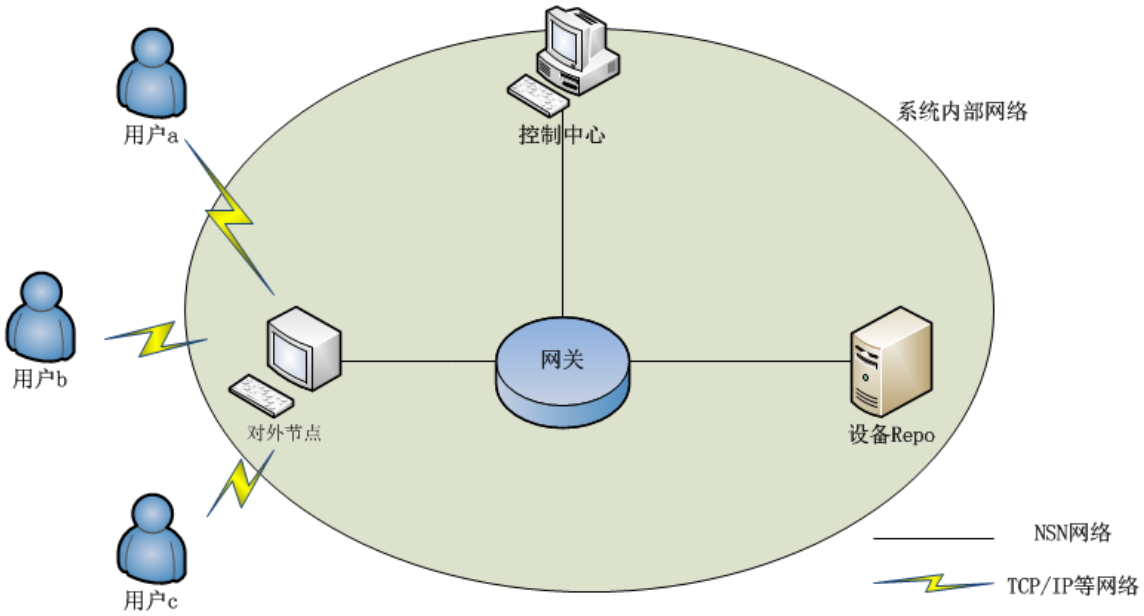

图 4. 1 基于 NSN 的楼宇控制系统实现图

图 4.1 是整体系统实现的一个大体的构造，具体形式先从系统内部网络部分说起，其中控制中心与对外节点都与网关部分相连，这里网关的主要作用就是起到 ACL 用户控制的作用，只允许指定拥有权限的用户对之后的设备 repo 节点进行操作。

设备 repo 节点是按照之前第三章图 3.2 的结构相互连接的，这里的连接不是如 TCP/IP 中通过地址信息建立通道的方式，而是上层的节点上线时或定时会发出一条包含自己部分名字的特殊前缀的兴趣包，下一层节点在除了监听自己名字的前缀外还额外对上一层发送的这种特殊前缀进行监听，当收到特殊前缀的兴趣包后下层节点就会将自己的前缀发送至到那个上层节点。这样上层节点就能定时更新与自己相关的下层节点的数量与各自的名字，这种连接是不会跨多层节点的，每个节点只会与自己上下一层的节点产生联系，例如某栋楼宇第三层向内部网络广播出名字为/NSN-BAS/build1/floor3/find 的兴趣包后，第三层楼的每个房间都会向 floor3 的节点返回类似名字为

/NSN-BAS/build1/floor3/room2/found 的信息包，floor3 接收到之后将这些前缀都存入节点中一个数值中以作记录。

控制中心是管理系统内部监控设备状况，实现反馈控制的地方，而外部节点是接收外部网络用户的命令并将其转化为 NSN 网络兼容的信息格式的中转点，控制中心与外部节点发送的兴趣包是十分相似的，区别在于代表的请求来源一个是内部网络，一个是外部客户。

当控制中心或外部客户需要设备节点中储存的数据时，发送的兴趣包在网关通过 keychain 函数与 validate 函数的检验以确定内容的完整性和身份权限，再转发到设备节点部分，当指定的节点接收到该数据包后对自身节点内数据进行检查，如果请求的数据存在且未过期则直接返回，如果没有或数据过期则根据自己下层节点记录表中的前缀依次发送数据请求，获得数据计算整合后再返回并更新自己节点数据。

外部节点只有读取数据的权限，而如果是控制节点还可以对设备进行改变数据或状态等命令，这些命令一般是直接转发到最底层单个设备的节点上直接进行操作，不过事先在某些较高层节点写好算法也是可以实现例如整层楼耗电平均化等复杂的操作。

第5章 NSN 系统实现架构与性能分析

5.1 计算系统的实现

5.1.1 计算系统结构

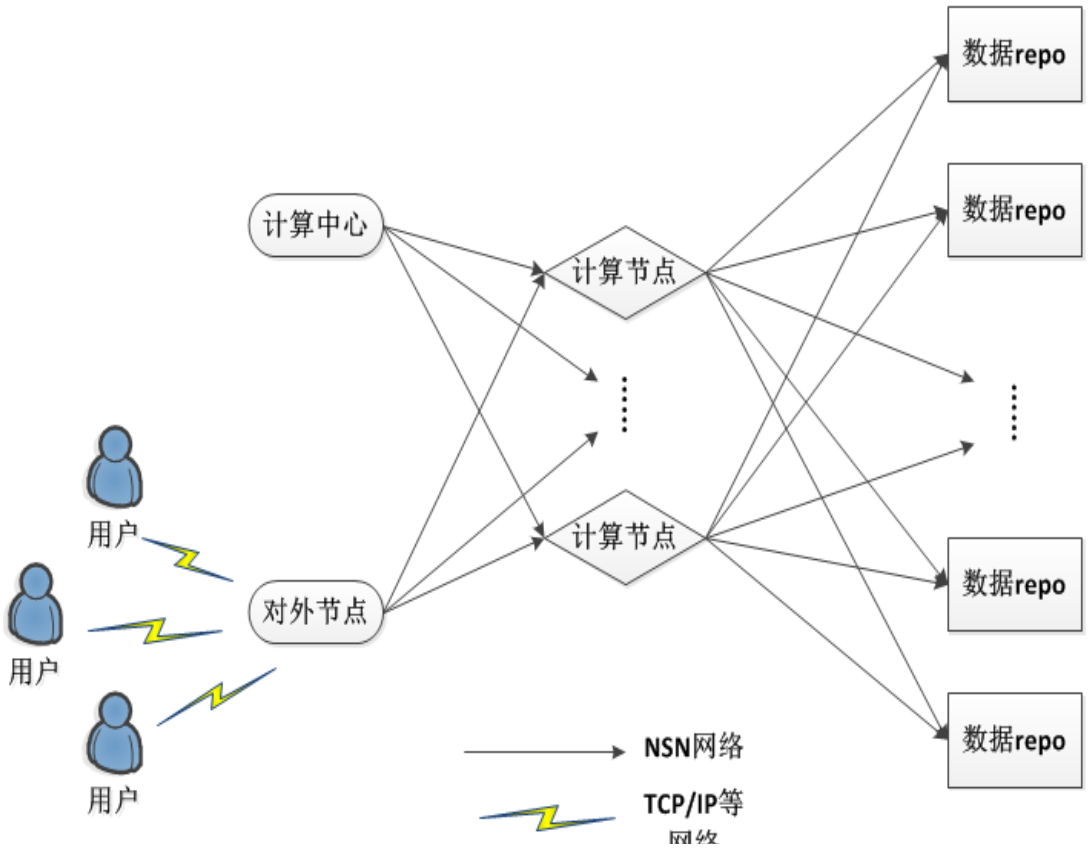

图 5.1 NSN 分布计算系统整体结构

整个计算系统的结构如图 5.1 大致分为三层：发布命令的计算中心，实行分布计算的计算节点，存储数据的数据 repo。

第一层分为内部 NSN 网络直连的计算中心与充当外部网络转换器的对外节点，这一层主要负责的是将完整的整体计算任务拆分开来，化作一定数量的小型计算问题在冠以相同的前缀作为兴趣包，按照一定的间隔与顺序发送出

去，再回收到第二层计算节点发送回的结果后进行整合运算以获取最终需要的计算结果。

第二层是可以任意扩展数量的多个计算节点，这些结算节点除了名字之外结构与内部提供的服务、算法都完全一致，平时除了自己的名字外还对某个固定的前缀一并监听。计算中心对单个节点的操控时兴趣包匹配的前缀就可以具体到每个节点名字，而多个节点一起实现分布式计算时就使用计算节点通用的前缀，当接收到的兴趣包通过安全验证后，计算节点会根据兴趣包名字除去固定前缀后剩余的部分确定调用的计算服务名字与参数，在向第三层获取需要的数据或根据之前缓存中的数据进行计算得出结果封装入数据包中发回。

第三层是数据 repo，主要是存储数据以供上层节点调用，可以接受上层节点的命令对数据进行改动或复制等操作，必要时也可以进行一些简单的计算并返回结果。第二层的每个计算节点都与所有的数据 repo 相“连接”，也就是说计算节点间能获取的数据范围是完全一致的，这是因为计算节点发送的兴趣包前缀部分是完全一致的，所以在数据 repo 看来就相当于只有一个计算节点，这样保证分布式计算的每一个节点都是一致的，才能保证每个子任务无论分配到哪个计算节点都能完成。

可以从图中看出计算系统的结构并不是采取的树状分布模式，而是一种类似层次化网状结构的模式，区别就在于每一层节点之间的连接方式，层次化网状的每个节点可以说都是与上下层所有节点有着联系的，这种结构带来的好处就是使整体系统更具弹性与扩展性，可以很简单快捷的增减计算节点的数目且不需要对其他节点作出任何改动。

这种分布式不基于通道的通讯方式使得计算中心不需要关心下一层计算节点的数量与他们的地址信息，真正做到将其聚拢起来看做一个巨大的计算节点，实际的分配工作是由 NSN 网络平台的策略层来实现的不需要计算中心进行管理。

正是无需节点自身进行其他节点地址等信息的配置，整体系统的配置难度可以说比起传统基于 TCP/IP 体系的分布式计算系统简化了太多，这也是本计算系统相较以往计算系统的一大优势，节点编写时节省了地址配置等方面的麻烦，整体配置是更是既节约了时间又节约了人力，还进一步避免了配置错误带来的各种错误。

5.1.2 计算系统的安全性

NSN 计算系统的安全性主要由信息包检验和 ACL 身份控制列表两个步骤来实现：

1) 信息包检验的过程是使用 validate 函数来完成的，当发送信息包时，发送节点会先对名字与内容等部分进行摘要，摘要得到的是通过 hash 函数将任意长的被摘要部分转化成的固定长度信息。通用的方法是使用 SHA 编码加密得到 128bit 长度的摘要，之后再使用发送身份的私钥对摘要再次加密后得到的信息即为数字签名，将其添加在信息包的末尾一并发送出去。

   当节点接受到匹配的信息包之后，使用 validate 函数对信息包是否完好进行检验，其内部检验的方法是通过 keychain 函数由信息包获得发送者的身份，从而可以获得对方身份对应的公钥，使用公钥来解密数字签名得到摘要信息，同时对信息包再次进行 SHA 编码加密得到新的摘要，通过比对新旧摘要是否相同便可以确定此信息包是否伪造或者中途产生破损。

   Validate 检验的配置文件可以主要分为“规则”与“信任集合”两个部分，规则是用于告诉验证器应该如何检验一个到来的信息包，而信任集合告诉验证器哪些证书是可以直接信任的，其中而规则又可以拆分为验证信息包是否符合资格的“过滤器”与检验是否需要进一步验证过程的“检验器”[9]。

   当节点接收到信息包后过滤器会使用每一条规则与数据包进行比对，直到其中出现一条匹配的规则为止，之后该匹配的规则所属的检验器就会用于检验此数据包的有效性；如果过滤器没有找到任何一条匹配的规则，这个信息包就会被视作是无效的。由于当数据包被某条规则匹配之后其后的规则都无法再对其进行匹配，所以应该把最特殊的规则放在前方，否则该规则就会变得毫无意义。

2) ACL 身份控制的实现是事先在节点配置文件内加入允许操作的身份以及各个身份对应的权限，节点运行后便读入配置文件并按其中内容向 keychain 函数添加过滤器，使接收到的信息包在 validate 时只有添加进的有权限的身份 keychain 函数才会继续进行下一步，没有权限的身份发送的信息包在这一步就被直接弃置。

举例来说 ACL 列表中可以含有以下内容：Cert、hostPrefix、sevicePrefix、Auth 等，Cert 包含的为发布命令的证书，hostPrefix 为该节点对外的前缀，起到的是类似地址的作用，sevicePrefix 则是对调用服务的前缀用于指定服务与参数，Auth 代表的是该身份的具体权限。接收到的兴趣包参考这几项便可以确定其身份与权限究竟如何，是否该执行其操作[6]。

一般来说内部计算中心节点具有大多数的权限包括计算任务的分配与执行、数据的读取与更改等，而如果有多个计算中心应该有一个节点具有最高的控制权限，控制权限可以控制节点的上下线以及前缀的更改等配置方面的功能。而外部节点代表的来着外网的用户权限就有高有低了，外部节点在转换外网用户的信息时也会为之申请一个 NSN 网络中的身份，并使用这个身份来签发相应的兴趣包，这个节点完全是为了有些特殊情况需要从外网接入操作时留下的，这时候还需要保证与外网节点间的 TCP/IP 通道的安全性，如没有必要可以取消掉对外节点并不影响整体计算系统功能。

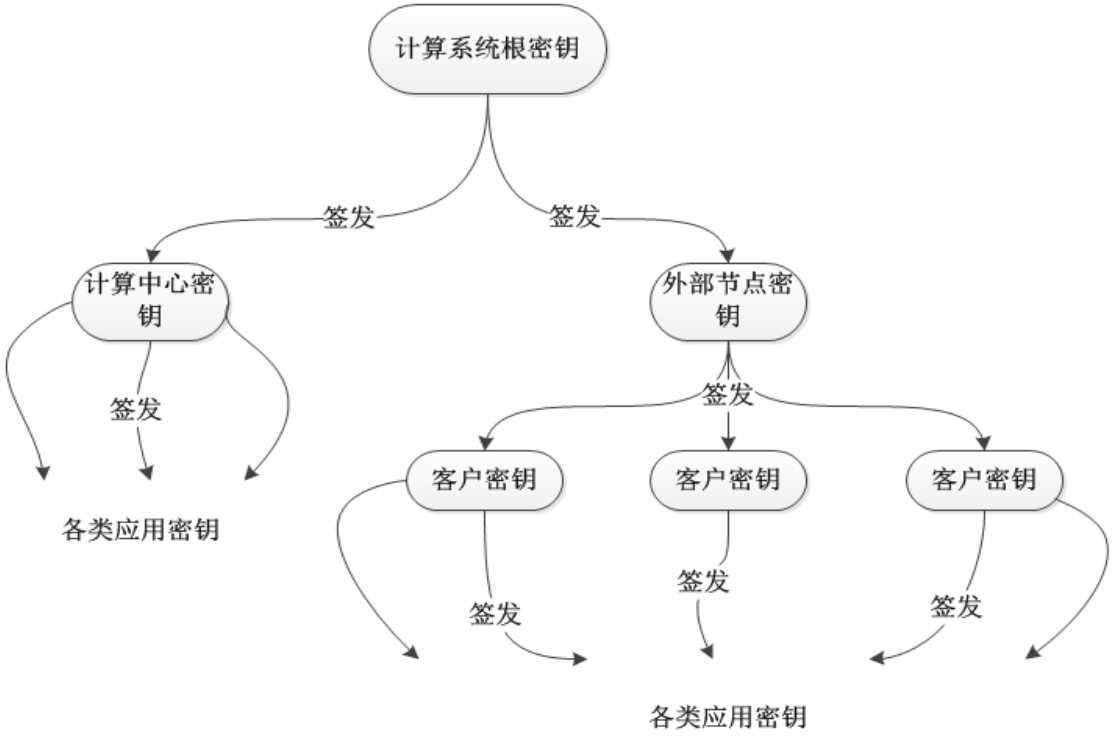

图 5. 2　计算系统密钥信任机制

图 5.2 是计算系统中密钥是如何相互签发的过程，先是由根密钥签发计算中心和外部节点等重要节点的密钥，之后这些节点密钥又可再次签发其中各个应用的密钥或先关联的用户的密钥，用户密钥也能签发相应的应用密钥。以这种层次签发密钥的方式能保证各层密钥足够的安全性与易用性。[5]

通过这两大部分的验证的信息包完全可以确定其安全性及对应的权限，同时验证时得到的身份如有需要也可以记录进文件中，这份记录不但可以用于安全监控还可以统计数据来源流向等内容。

5.1.3 计算系统的寻址策略

目前 NSN 网络最新的平台是 NFD(NDN Forwading Daemon)，可以对命名空间中每个前缀分别设定路由寻址策略，如未设定策略平台会自动向该前缀最靠前的邻居节点转发。

平台自带了四种寻址算法：best-route、broadcast、client-control 与 ncc，在计算系统中试验使用的效果分别如下：

- best-route：
  与未设定策略时类似，只向当前节点最靠前的邻居节点进行转发，只有那个邻居节点接收到兴趣包。
- broadcast：
  把兴趣包向所有邻居节点进行广播，邻居节点都同时会接收到所有的信息包且回复，但是每个发送出去的兴趣包只接收其中一个邻居节点的回应数据包，最后到的接收到的数据包来源比例并不均衡且不固定。
- client-control：
  与 best-route 的情况比较相似，但是不一定是最靠前的邻居节点接收到兴趣包，不过同样只会全部发送至一个邻居节点。
- ncc：
  效果和 broadcast 的类似，不过每个邻居节点可能并不能接收到所有的发出的信息包，而且接收到的回应数据包来源比例更加失衡，会出现某个邻居节点占了绝大多数的情况。

然而这些算法都不能满足分布计算系统的需求，首先分布式系统最关键的一点就在于将整体计算任务分割后由多个计算节点分担，所以 best-route 与

client-control 这种只发送到一个计算节点的方式明显是不可取的；其次发到多个节点的情况下希望能够尽量平均发送到所有可用计算节点，所以 broadcast 与 ncc 接收回应数据包来源不平均这也是一个问题所在；最后一点是如果每个节点都收到了广播出去的所有兴趣包，那么意味着每一个节点都会对这个子任务进行计算，虽然返回的数据包中只有一个被接收其他都被弃置，但是每个节点都付出了时间与资源来进行计算，这样完全无法体现分布式计算的优势所在，所以需要自行开发一种新的算法。

最后经过试验选定了一种自己编写的 random 随机算法，具体过程是通过随机数生成函数对该前缀接口数取余，假设有 n 个接口就会获得一个 0~n 之间的随机数，根据此随机数选定转发的接口。由于随机数生成函数是采取的精确到微秒的系统时间函数作为种子，所以可以保证低重复性，使得分发的接口尽量接近平均，同时一次只转发向一个端口保证只有一个节点会进行计算而避免重复计算带来的时间与资源上的浪费。

5.2 NSN 与 Web Service 对比

5.2.1 系统对 Web Service 的参考

NSN 的理念由于同样是面向网络层面的服务方面，所以在很多问题上都参考了 Web Service 的做法并加以改进使之适应于 NSN 网络的环境，可以在以下方面进行对比：

- 服务调用：Web Service 中通过服务的 URL 可以直接进行调用，参数就能包含于 URL 之中，NSN 将服务进行分层命名，使用前缀作为服务的标识而参数也同样位于信息包名字之中。
- 服务描述：Web Service 拥有 WSDL，这是一门基于 XML 的接口定义语言，用于描述 Web Service 服务的各项接口信息，向用户说明如何与此 Web Service 进行通信，同时可被人与机器所辨识。NSN 网络则也能提供一种基于分层结构的独特命名语言模板，其中服务名字的每个部分都可以进一步嵌套且具有具体点含义，用户依照说明书编写兴趣包名字即可成功按自己的需要调用相应服务，同样提供了一份人机可读的统一接口。

- 服务目录：Web Service 依靠 UDDI 来发布服务，UDDI 可以看作是一个 Web Service 的信息平台，可以在上面对服务进行注册与检索，同时这也是一种规范，使 Web Service 对服务的描述规范、透明化，提供有效的对包含所需功能的 Web Service 的检索并提供其具体调用方法与说明。而 NSN 由于还处于发展期目前还没有如此官方、大型的平台，只能通过个人网站发布或直接发送给有需要的用户说明书等形式来展示自己服务的功能与通信方法，今后如果能发展到一定规模就可以参考 UDDI 的方式建立属于 NSN 的发布检索平台。

5.2.2 NSN 相对 Web Service 优缺点

NSN 作为由 NDN 网络发展而来的新兴网络模式与基于 TCP/IP 模式同样面向服务的 Web Service 相比有这各方面的优缺点，认清 NSN 在各方面的表现优劣才能更好的对其进一步改进。

实现方面：

NSN 优点：

1) 不需要关注节点间的地址，转发由 NSN 网络平台负责服务接口内部不用编写地址环节，所以实现难度较低且省时省力，配置网络时也能大大减少工作量与可能的错误。
2) 节点间不通过地址信息的方式建立通信通道，不关注位置的理念使节点的增减变得十分轻松，只需要对该节点进行处理而不用对其他节点做任何改动。
3) 相比 XML 格式来说多层命名方式的传输开销要小上不少，能节约传输所消耗的成本并减少传输用时，提高系统整体性能。

NSN 缺点：

1) 没有 XML,HTTP 等成熟的业界标准支持，NSN 由于体系创新，所以使用的的许多协议标准等都是自己制定尚未成熟，而且新用户需要花费时间学习
2) XML 为 Web Service 带来了跨编程语言交流的能力，这点 NSN 无法办到，目前 NSN 主要是基于 C++、python、javascript 进行开发，暂时无法兼容更多语言，同时支持的几个语言间也无法直接通信。

安全方面：

NSN 优点：

1) 不用保证成本与难度都更高的连接通道方面安全性，转而保证传输的信息包的安全，能够更好的适应于断续连接与移动连接等模式。
2) 相比与连接安全性这类信任度永远处于一个暂时值的模式，数据块本身一旦能确保安全性能所带来的可信度是长久性的，整体网络安全性能更加强健。
3) 不需要端对端的安全性保障之后使得各地中继节点缓存内部的数据都可以得到充分利用，从而减缓网络流量的压力与用户延迟，提升网络性能与服务品质。
4) 其结构体系的天然性质使得泛洪攻击等网络恶意攻击方式难以生效，而数据间相互引用使得假冒数据需要同时篡改网络上大多数数据的内容，这往往是难于办到的。

5.3 计算系统丢包率与发送间隔关系

5.3.1 NFD 平台概述

NDN Forwarding Daemon (NFD)是最新的依靠 NDN 协议实施与发展的网络转发平台，因为 NSN 继承于 NDN 的大部分结构，所以 NFD 同样也适用于 NSN 网络下的转发。目前 NFD 已经成为组成 NDN 网络的一个核心部分，今后也会随 NDN 的发展同步更新[8]。

NFD 设计时的主要目标是能够支持各式各样的 NDN 网络技术的实验，这个设计方案强调的是程序的模块性与可拓展性，以便可以基于平台简易的对新的协议、算法或应用进行测试。目前的 NFD 还处于未完成的状态，并没有达到最理想的性能，现在的想法是通过尝试使用不同的数据结构与算法来优化性能表现，经过长时间的测试后即使在相同的设计框架下也会出现更优化的布置与实施方案。

NFD 会持续在以下三个方面进行发展更新：

1) 改善整体框架的模块性
2) 及时更新使进度追赶上 NDN 协议的变更
3) 添加更多其他新出现的特性

整个平台的发展倾向是保证模块化的结构更加稳定，允许研究者在其上面部署与实验更多不同的新特性，而这些特性中突出者有可能被应用于下一次更新的 NDN 协议之中。

NFD 的主要功能模块如下：

- 核心部分提供一系列可以在不同 NFD 模块间通用的服务，包括而不限于 hash 计算、DNS 解析、文件配置、接口监控等等。
- 接口模块负责将 NDN 接口的抽象说明布置到下方各层的运输机制上
- 表格模块负责布置 Content Store 缓存部分，PIT 表与 FIB 转发表与其他数据机构，以用于支持 NDN 数据包与兴趣包的转发。
- 转发模块用于实现基础的信息包传输路径，与接口模块、表格模块、策略模块间都有着交互关系。
- 策略模块可以看做转发模块的一个核心组分，可以支持不同的转发策略的实施，包括策略选择、管理并与转发管道进行挂钩。
- 管理模块部署 NFD 管理协议，协议中允许应用配置 NFD 与设定/查询 NFD 内部状态，协议间的交互是通过 NDN 中应用与 NFD 间的兴趣包与数据包交换而完成的。

5.3.2 产生丢包原因

在此前一系列的实验中发现如果计算中心节点连续高频率向 NSN 网络中发送包含分割后计算任务的兴趣包时，会随着发送间隔的缩短而产生一部分兴趣包丢包的现象，这些兴趣包丢失的原因依照发送流程有下列可能性：

1) 程序没有发出兴趣包
2) 发出方 NFD 未转发兴趣包
3) 兴趣包于传输途中丢失
4) 接收方 NFD 未接受兴趣包
5) 接收方未回复所有的兴趣包

通过对双方 NFD 平台的转发接受记录进行调查后首先可以判断发送方程序已经发出正确数量的兴趣包且其 NFD 全部进行转发，而如果发生丢包则接收方 NFD 只接收到一部分兴趣包，但回复数据包数量与接收兴趣包数量相等。

由此经过筛选可以判断丢包的原因是 3)与 4)中一者，而由于当发送频率较低时从未发生过丢包现象可以判断节点间连接情况良好，而带宽也十分宽裕不会是由于传输中的堵塞，所以可以认为是接收方 NFD 短时间内来不及处理

完前面的兴趣包而新的兴趣包已经接收到，这时后面的兴趣包就很有可能被忽略丢弃掉，从而导致接收方节点只回复 NFD 处理完的兴趣包数量的数据，也就随之产生了丢包的现象。

由目前分析来看这属于 NFD 尚未完善的连续接受部分问题，暂时还无法直接解决，希望在之后的更新中能改造这方面的性能，下面章节将会对一次性发送的数据包数量或发送间隔与丢包率之间的关系进行分析。

5.3.3 丢包率关于发送间隔的分析

数据包数量 / 丢包率 / 发送间隔	500	1500	5000
25ms	40.6%	38.4%	37.8%
50 ms	2.8%	13.6%	11.4%
100 ms	0%	3.7%	5.3%
200 ms	0%	0%	1.4%

表 5.1　计算系统数据包发送数量/发送间隔与丢包率关系表

上表中展示了一系列实验中一次性发送数据包数量与数据包发送间隔各种组合对应的丢包率。

可以看到当发送间隔很短只有 25ms 时，发送数据包无论多少丢包率都很接近，这是由于这已经远超过了 NFD 最高处理速度，所以数据包数量无法产生太多影响，这时的丢包率平均大约有 40%左右。

当发送间隔提高至 50ms 时，整体丢包情况有较大的改善，尤其是发送数量最少的 500 个的丢包率已经很低接近没有，其他两个的丢包率仍然超过 10%，这可以说明相同发送频率下最初部分的数据包由于对方 NFD 最初处于空闲状态所以能处理更多的兴趣包，后面部分的丢包率就会明显上升一截。

当发送间隔设为 100ms 时一次发送 500 这种较低数量的兴趣包已经不会发生丢包的状况，而 1500 与 5000 则还有少量 5%以下的丢包比例；发送间隔进一步提高到 200ms 时 1500 个也没有丢包了，只剩下 5000 个还有微量丢包发生。

从上面的数据中我们可以总结出丢包率与发送间隔成反比例关系，尤其发送间隔短过某个阈值后丢包率会陡增，而一次性发送的数据包数量越多也会导致相同发送间隔下丢包率有所上升，当数据包多到一定数量后区别就微乎其微了。所以可以一次性发送很大量的数据包，依次对各个发送间隔进行测试，得到丢包率合适的间隔后可以直接套用在连续的数据流情况上，因为此时一次性发送极大量数据包与连续不间断发送数据流的丢包情况应该极度相似。

5.4 NSN 缓存机制与计算系统性能

5.4.1 缓存机制实验环境与流程

- 进行缓存机制的实验环境：

  运行环境：虚拟机集群

  CPU： Intel(R) Xeon(R) CPU E31225@3.10GHz

  内存： 2G

  硬盘： 20G

  操作系统： Linux Ubuntu 13.04

  依赖软件： crypto++

  Sqlite3

  Boost 库

  Libpcap

- 缓存机制的实验流程：

在末端数据 repo 节点设置一份数据，请求节点与数据 repo 间有着一到多个计算节点。

请求节点先向最近的计算节点请求该数据，所有计算节点最初缓存中都没有这份数据，这个计算节点又会依次向下一个节点请求，直到倒数第二个节点向数据 repo 请求道数据后再依次返回，当请求节点获得数据后停止计时得到花费的时间。

之所以没有最初直接将目标取为数据 repo 让其他节点直接转发兴趣包，就是为了加入每个节点内部处理的因素作为参考，观察多层次节点相互调用数据时缓存起到的作用大小程度。

如此反复多次之后再以相同的模式但每个节点取消缓存数据的方法来进行，得到所花费时间数据后再与有缓存的模式进行对比。

5.4.2 有无缓存的性能差距

实验中先以一个三层的结构来对比，即请求节点与数据 repo 间只有一个中继计算节点。

最后得到的数据是无缓存时每次请求用时与有缓存第一次请求用时基本相等，请求一次数据从发出到接收用时约为 49~62ms，平均 56ms；而有缓存时第一次后的每一次用时约 14~30ms，平均 22ms。

可以看到有了缓存之后即使中间只有一层节点，也能节约一半以上的时间，这节省的不光是数据传输的时间，还包括中介节点编写数据包决定发送目标等一系列操作用时。

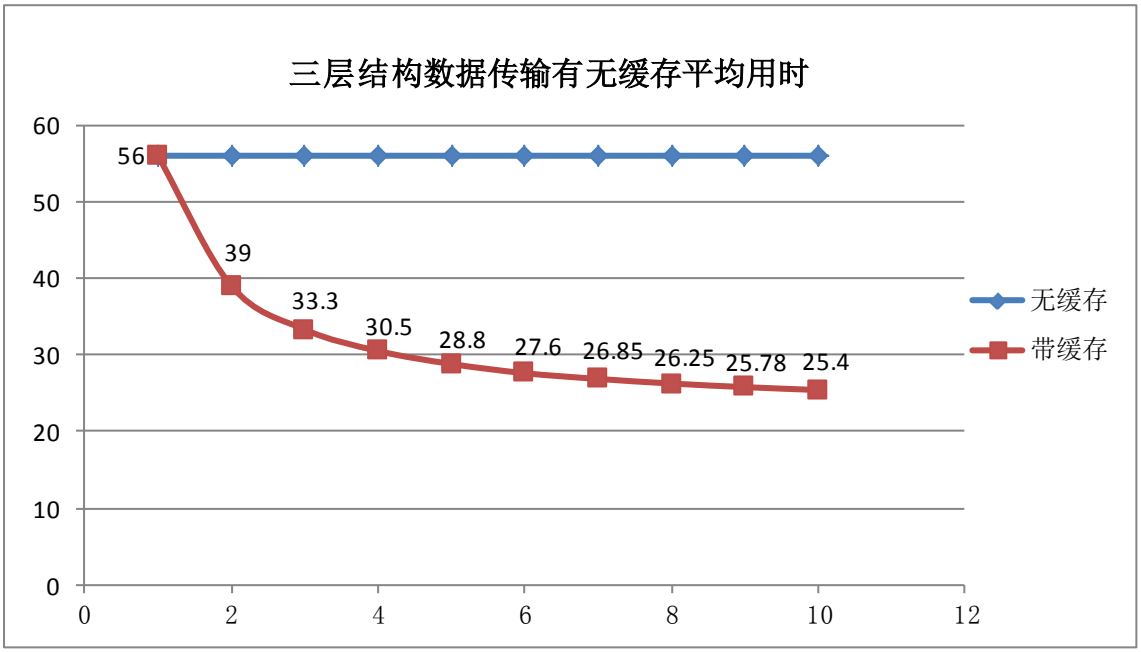

图 5. 3　有无缓存的三层结构数据传输平均用时

图中显示的是进行不同次取回数据运算平均每一层花费的时间，没有缓存的情况下无论进行多少次平均时间都是与第一次用时一致的 56ms，而有缓存时每额外进行一次操作平均用时就会有相应的减少，尤其是前面几次降低幅度较大，最终平缓趋近于一个略高于有缓存节点第二次操作用时的值。

当进行多层数据的时候缓存带来的降低平均用时会更明显，因为有缓存时实际上除了第一次之外的操作都只用经过一个节点的路程，而无缓存每一此操作都要走完全程，中间节省的时间差会变得更大。

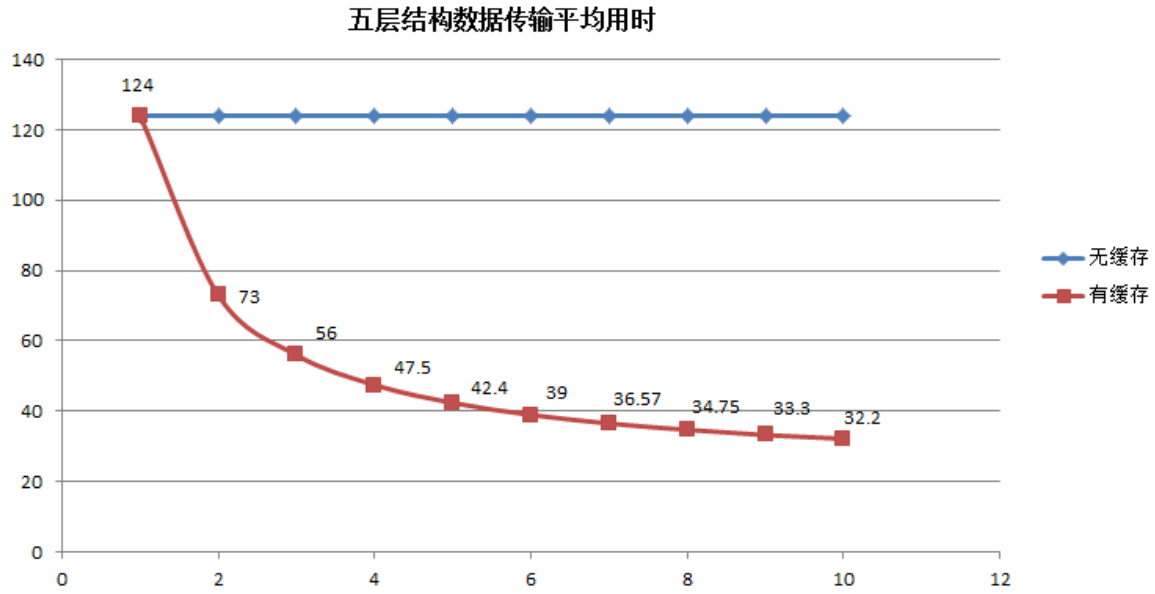

图 5. 4 有无缓存的五层结构数据传输平均用时

可以看出当节点层次提升到五层后，有无缓存的平均用时比由三层时的 50%左右直接降低到 25%左右，同时前几跳的坡度也比三层时相比显得更陡，说明降低平均用时的效果生效也更快。

所以从上面的结构我们可以得出当同一个或多个计算节点需要多次调用同一个数据的时候缓存能起到巨大的降低延迟与减缓带宽压力的作用，其中路程经过的节点越多缓存的优势就越明显。

这也就能从另一个角度说明 NSN 网络关注与数据本身的安全机制不光在安全性能上有重大作用，同时这种安全机制能使用缓存数据的天然性质使其还能不可忽视的提升整体系统的性能与负载能力。

第6章 基于 NDN 的分布式计算系统整体评估及未来工作展望

6.1 分布式计算系统整体评估

本计算系统通过对 NDN 网络的研究与了解，提出并实现了新型的 NSN 网络并以此为基础来构建整体系统，依靠 NSN 网络特有的基于命名服务的特效来实现分布式计算，与传统网络相比无论是在构造难度、配置难度还是系统性能上都有所优化。

通过自行编写的寻址策略，计算系统中分割出的子任务在各个计算节点间的分发状况良好，既让每个节点接收到的任务数量相近保证了负载均衡，又确保了同一个任务不会重复分发到多个计算节点上造成重复计算而浪费时间与资源。最终系统经实验可以支持按照一定间隔发送的连续大量子计算任务，基本满足了分布式计算系统的初步需求，但之后的程序改进过程中还应该进一步优化路由寻址算法使其能根据各个节点的状况来优化计算任务的发放而不是目前的随机分配方式。

系统整体基于 linux 下的 C++环境来实现，目前 NDN 网络暂时还不支持 windows 下的运作，而 NDN 除了 C++还对 python 与 javascript 有着支持，系统如有需要可以考虑编写其他语言的版本。

NSN 网络是一种新兴的试图替代 TCP/IP 体系的网络体系，国内目前还没有对这类基于数据的网络研究，国际上这方面也仍处于研究的萌芽阶段，所以 NSN 网络作为 NDN 网络在网络层面向服务方面的一种拓展具有相当的研究意义。

而考虑现实意义上 NSN 网络也有着其独有的优势，无论是在能够防止多种现存网络攻击方式的独特安全机制上，还是在支持缓存的高效数据与带宽利用上，还是在完美契合断续连接与移动连接等模式的分布式结构等方面都具有自身巨大的应用价值。其简单的结构配置与节约带宽等资源的能力，强健的安全性能与提供网络层服务的特性必定能支撑 NSN 网络在未来的科学与商业领域得到长足的发展。

6.2 未来工作展望

现在基于 NSN 的分布式计算系统仍处于研发改进的阶段，目前的成果更多是作为初步特性的示例程序而实现。程序本身在接口、策略、乃至整体结构上都还有着许多可以改进之处，在系统能承受的计算规模以及具体的用户身份权限管理部分也需要进一步的测试与实验后才能优化整体系统，目前的整体程序还是处于离成品有很长距离的示例程序阶段。

今后的开发工作中希望能在以下方面进行改善加强：

1) 改善 NFD 等转发平台的路由寻址策略，做到针对计算节点的负载、是否存在错误、连接质量等方面来优化任务分配
2) 进行更大规模数据的实验与改进，使得系统能真正负载起正式计算系统级别的流量
3) 尽量消除或减少网络中传输时的丢包现象
4) 建立完整的网络层面对服务进行描述的结构作为统一接口

插图索引

参考文献

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in Proceedings of the 5th international conference on Emerging networking experiments and technologies, 2009, pp. 1-12.

[2] W. Shang, Q. Ding, A. Marianantoni, J. Burke, and L. Zhang, "Securing Building Management Systems Using Named Data Networking," National Science Foundation,2013

[3] J. Finis, R. Brunel, A. Kemper, T. Neumann, F. Farber, and N. May, "DeltaNI: An efficient labeling scheme for versioned hierarchical data.", Proceedings of the 2013 international conference on Management of data. ACM, 2013.

[4] B. F. Cooper, N. Sample, M. J. Franklin, G. R. Hjaltason, and M. Shadmon, "A fast index for semistructured data," VLDB. Vol. 1. 2001.

[5] D. K. Smetters, and V. Jacobson, "Securing network content," Technical report,PARC,2009.

[6] E. Osterweil, D. Massey, B. Tsendjav, B. Zhang, and L. Zhang, "Security Through Publicity," UCLA.

[7] 雷凯. 命名数据网络与互联网内容分发，中国计算机学会通信，第八期, 2012

[8] NDN 官方网站：http:// http://named-data.net/doc/NFD/current/

[9] http://redmine.named-data.net/projects/ndn-cxx/wiki/CommandValidatorConf

[10]刘韵洁. 面向服务的未来互联网体系结构与机制研究标书, 2012

[11]http://en.wikipedia.org/wiki/Web_service

[12]W.So, A. Narayanan, and D. Oran, "Named data networking on a router," ACM SIGCOMM Computer Communication Review, 2013, Vol.43 (4), pp.495-496.

致　　谢

在本次毕设项目的进行过程中，我的指导教师曹军威研究员帮助我制定了项目研究的方向，在研究时使用的方法上给予指导，对项目实现的软件程序进行审查并指出应该更改的缺陷，对项目整体进程予以督促，正是有了曹老师的悉心教导本次毕设项目才得以顺利的完成预期目标，在这里得先感谢曹老师毕设其间给予的帮助。

然后实验室的陈硕师兄也对我的研究工作给予了很多帮助，能够成功的在我不熟悉的环境下编程开发都是靠陈硕师兄的耐心解答，同时还教给了我科研工作其他方面的各种方法与窍门，实在是受益匪浅。