PCLive：以 NDN 数据命名网络实现互联网直播的系统设计

骆家辉

摘要

NDN 数据命名网络自被提出后就成为国内外的研究热点，其将内容本身，而非地址，看作网络的主要内容，并以此目标颠覆了当前基于主机的网络架构。因而，NDN 成为下一代互联网中一个具有代表性的网络架构。然而，正是因其颠覆现有 TCP/IP 架构，缺乏应用的支持是当前 NDN 网络发展的最大障碍。本项目复现的论文主要内容是在现有网络架构下抽象出 NDN 逻辑网络，将应用程序需要传输的数据包 NDN 化，并通过该 NDN 逻辑网络进行内容分发，以最大兼容现有的网络体系结构并发挥 NDN 网络优势。基于该网络设计方案，开发出 PCLive 直播系统，将互联网直播通过 NDN 架构转发，通过在真实场景应用此系统，观察直播实时播放效果，验证该设计的有效性。PCLive 对互联网直播架构进行了最小的改动，实现了对现有视频直播软件的最大的兼容。PCLive 直播系统通过在用户端视频播放器嵌入 HLS-NDN 协议转换模块，达到用户透明性，在不需要用户配置的前提下利用 NDN 网络架构优势，有效提升用户观看直播的体验。同时，我观察到原文作者仅在已知数据存在的前提下实现了分发，并未关注数据发现环节。因此，我在原文已实现的数据分发功能基础上进行改进，使系统各节点具备网络资源发现能力，使其提出的数据分发架构对不同的应用场景有更高的兼容性。

关键词： NDN 数据命名网络；互联网直播；协议转换；实时数据分发；应用程序 NDN 化

1 引言

目前使用的互联网的最早前身是 ARPAnet，ARPAnet 是第一个可以实际运行数据包交换网络，同时也是第一个实现了以 TCP/IP 为通信协议的工作网络。TCP/IP 通过建立基于网络拓扑位置端到端的连接，这种以连接为基础的 TCP/IP 体系结构能够很好地满足网络设计初期的需求，由于这个时期的通信主要发生在具体的两台实体设备间，需要确定具体的设备位置，因此 IP 数据包的交换使用 IP 地址为导向。

随着互联网飞速发展，网络主要功能已经从简单的数据传输通道发展到满足高效率、大规模、安全的内容获取和分发。已经沿用了几十年的传统互联网架构已经越来越凸显出它固有的缺陷。首先是数据转发的效率低。建立的通信管道容易造成拥塞，假如同一份数据受到多个客户端的请求，就要从单个服务器发送无数次，这就造成了一个服务器带宽资源浪费的问题。目前有一些网络优化方案已经被提出而且被运用到了实际当中，例如 P2P 以及 CDN 技术，但是这些技术在提高数据共享和分发效率的同时，更加剧了大量重复传输这一问题。二是原有网络扩展性差的问题。在发现网络架构出现缺陷时，我们必须在原有的总体设计中打

补丁，长期如此就导致网络设计管理越来越复杂。另一方面，全球的 IP 地址在 2011 年已经分配完毕，为此我们设计出新的 IP 地址分配策略以应对 IP 地址枯竭问题，尽管新的 IP 地址格式宣称可以满足可预见的网络设备数量增长，但这依然只是一个治标不治本的方案。三是安全性弱。TCP/IP 架构重视对通信管道的维护，但不能保证数据本身的可靠性。例如在建立一个信任的连接后，发送方发送的恶意代码会被接收方不经检查地接收。第四个问题是移动性差。目前移动互联以及智能物联快速发展，网络实体的移动性越来越高，网络实体移动超出原有网络覆盖范围后，网络地址会发生变化，网络连接也需要被重新建立。

TCP/IP 提供的通信方案虽然是一个突破性的网络架构设计，但其仍然是类似电话线路通信方式的，需要在两个实体之间进行端到端的数字数据交换。TCP/IP 互联网只提供一个通讯管道，只负责维护管道，不关心传输的内容。不断在原有的体系结构中进行改进，无法从根本上解决问题，因为其依然以主机为中心，基于端点进行寻址和转发。

针对互联网体系结构的改进与创新，专家学者们的意见主要分为改良式、演进式和革命式，而革命式的发展思路自提出后已受到许多国家的关注和支持，目前已经形成了一系列实验平台和原型系统。因此，围绕革命式发展思路的下一代互联网体系结构成为了当今研究的热点方向。而当中的数据命名网络 NDN [16] 从 2010 年正式开始实施后已实现了飞速发展，不仅完善地解决了一系列包括路由策略在内的设计难题，开发出 NDN 数据转发进程 NFD 以及 NDN 网络模拟器 ndnSIM 等，更在全球范围内建立起建立起 NDN 测试平台。

NDN 的核心关键在于完全舍弃 IP 地址，采用基于内容本身的通信方式，根据内容本身对网络中的所有数据进行命名，在数据转发过程中通过对数据名字的匹配来提供。这样的通信方式不再关心内容数据的存储位置，而直接提供面向内容的服务。NDN 中有两类角色，分别为消费者和生产者；有两类包，分别为兴趣包和数据包。消费者若要请求某内容，则产生带有相应名字的兴趣包，通过网络转发到达存有此内容的结点，此结点收到兴趣包后沿着兴趣包的反向路径返回携带相应名字的数据包，最后消费者成功获取请求的数据。因此，NDN 采用接收者驱动的数据分发机制达到网络数据共享的效果。

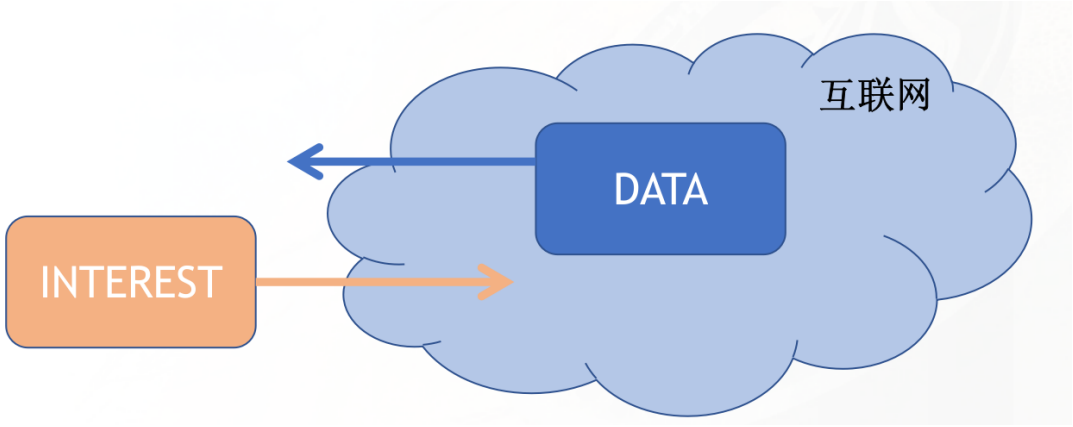

图 1. NDN 网络通信原理

NDN 网络架构具有天然的优势。一是可以做到多源多路径的数据分发，并且使用数据缓存让数据分发更为高效。二是其不需建立管道连接，更适合现代场景，可运用于移动互联网和自然灾害救援等网络环境变化性大的场景。三是通过 NDN 网络转发的所有数据必须具有签名，需要时可以进行数据加密，保证了数据的可靠性以及不可抵赖性。

然而，目前 TCP/IP 体系已经沿用多年，NDN 对传统 IP 网络结构进行改革，因其与现有网络之间的不兼容性，直接导致其受阻于广泛的应用支持，大部分情况下只能在专用的网络内部使用。目前解决这种矛盾的方式是将现有的应用进行 NDN 化，通过在应用层和 NDN 之间进行转换来实现 NDN 化现有应用，这样不仅可以享受到 NDN 架构带来的好处，还可以缩小开发的工作量。同时获取真实流量数据用于 NDN 网络进一步的研究。

拟复现论文原文通过将该方法应用于互联网直播，开发出使用 NDN 分发机制的网络直播系统 PCLive，只对互联网直播的数据分发部分进行 NDN 化的处理，这样可以在复用现有直播系统数据发布部分的前提下，通过实现 NDN 数据分发部分，兼容现有互联网直播应用，同时充分利用 NDN 的优势，验证应用程序 NDN 化的有效性。在 PCLive 系统实现过程中，我与原文选择同样的 HTTP Live Streaming(HLS) 作为进行 NDN 转化的直播视频格式。PCLive 针对性地解决了互联网直播系统的协议转换、数据命名、数据版本更新等常见的 NDN 系统架构问题。

具体来说，PCLive 在视频播放器中插入一个数据转换模块，将对 HLS 直播的请求转换为 NDN 数据请求，并向 NDN 网络发送。当网络中某个具有该数据的 NDN 节点收到该请求后，将直播数据返回给请求发送方，并且在 NDN 数据返回至请求方后由内嵌在视频播放器的数据转换模块将数据转换为原有的 HLS 视频格式供播放器播放。PCLive 以此方式兼容了现有的互联网直播系统组件，包括视频播放器、视频转码器以及直播视频发布软件。

在系统评估方面，因为 NDN 架构对传统网络系统架构进行革新，包括吞吐率在内的许多以往使用的网络评测指标无法真正体现 NDN 数据分发能力，故我使用了较为主观的方式对网络数据分发效果进行评价——通过在数据发布方和数据接收方分别运行 PCLive 系统使用 NDN 网络传输方式进行直播数据分享，分别在双方本地搭建 Web 服务器并通过浏览器视频播放器的方式观察视频播放结果，对数据发送方和数据接收方的视频进行对比观察，考察 NDN 网络数据分发的能力。并以每一份 NDN 数据自请求发送至数据返回的 RTT 作为测量指标，从而计算出 NDN 网络数据的传输速度，作为 NDN 网络分发效率的具体体现。

值得注意的是，原文的工作仅关注于数据分发环节，所有的数据均建立在已存在于网络内的前提下进行请求和响应，并没有关注数据发现环节。为此我设计并实现了一个数据发现功能，并由 PCLive 对数据名称进行自动管理，为数据分发功能提供服务。

最终我成功将互联网直播流进行 NDN 化并通过 IP 网络进行传输，并在直播数据接收方将 NDN 数据复原为 HLS 格式，通过在接受方与发送方对比观察的方式证明了对现有互联网应用 NDN 化的可行性，并且在长达 6 个小时的记录中，发现数据传输速度基本稳定在 7MB/s 至 8MB/s 之间，可以满足目前主流的 1080p 高清视频数据实时传输。此外，作为论文复现改进部分的数据发现功能使系统各节点具备网络资源发现能力，并且可以对 NDN 网络数据名称进行有效管理，使原文提出的数据分发架构对不同的应用场景有更高的兼容性。

2 相关工作

应用程序 NDN 化意味着不重零进行 NDN 网络的构建，而是翻译应用层协议使其能够在 NDN 网络上运行，这种方式可以尽量使用 NDN 网络优势，并减少开发全套 NDN 网络架构所带来的成本开销。目前有一些工作是遵循 NDN 化的思想进行的，NDNVideo [5] 和 NDNlive/NDNtube [14] 使用 NDN 网络进行视频流的传输，ACT [18] 是 NDN 网络音频会议

工具，而使用 NDNRTC [3] 可以在 NDN 网络举行实时会议，但是这些均已经停止被维护，无法跟上 NDN 网络架构的快速发展。

在协议的转换方面，前人也已经做过了一些工作，其中 VoCCN [4] 在 VoIP 和 CCN 之间进行协议转换，mailSync [6] 将 IMAP 和 XMPP 转换为 NDN 数据进行传输，而 iVisa [2] 转换的是 DASH 协议。

3 本文方法

3.1 本文方法概述

原论文继续遵循互联网应用 NDN 化的路线，将互联网直播中进行了 HLS-NDN 的转换，PCLive 实现了在浏览器视频播放器中嵌入协议转换模块，充分利用 NDN 分发能力，在修改数据分发部分的同时复用了数据发布部分，而数据发布部分由广大的第三方进行维护，使 PCLive 具有了更高的兼容性和更长的寿命。PCLive 在真实场景部署运行，可以为后续 NDN 的相关研究提供提供大量的真实流量数据。

通过将该方法应用于互联网直播，开发出使用 NDN 分发机制的网络直播系统 PCLive，在兼容现有应用的前提下，利用 NDN 的优势，验证了该方法的有效性。

3.2 系统设计总体架构

对现有直播系统进行 NDN 化的首要任务是确定系统的边界，而常见的互联网直播架构包含两个主要部分，直播发布和直播分发。PCLive 使用 NDN 替换分发部分，同时重用发布部分，系统总体架构图如图 2 所示。根据我对拟复现论文原文的理解，首先我使用现有的 OBS 软件捕获一个直播流并通过 RTMP 协议推送到视频服务器，之后视频服务器将 RTMP 流转换成 HLS 格式并存储在本地。PCLive 在这之后加入一个过程，将 HLS 文件翻译成 NDN 数据格式，存储在数据库中，并实时地服务于 NDN 网络。消费者希望观看直播时，先使用浏览器打开直播播放页面，网页中的视频播放器嵌入了一个 NDN 转换模块，转换模块将直播请求转换成对应的 NDN 请求并发送到 NDN 网络中。NDN 网络中的任意一个具有该数据的节点将以 NDN 数据包的形式向请求方响应，并最终在请求方处转换成 HLS 文件供视频播放器播放。

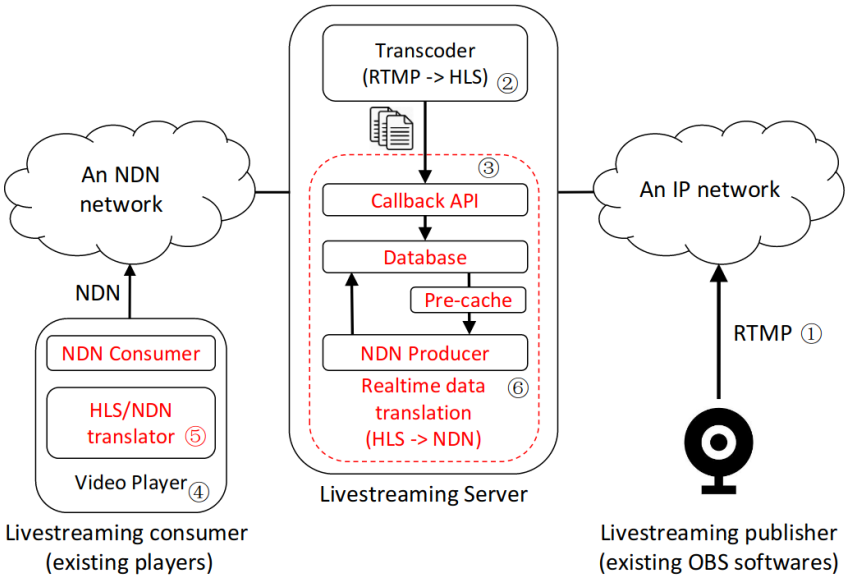

图 2. 系统总体架构图

3.3 数据命名方式

NDN 网络中一个重要的部分是数据命名，在 NDN 网络中，不同的数据必须要保证自己具有全局唯一的数据名。关于数据命名的方式，有相关的工作提出了不同的命名规则，例如，Davide Pesavento 和他的团队 [8] 提出了一种在车载网络中将双维地理区域映射到单维命名方案的命名方法，Mohammad Shahrul Mohd Shah 的团队 [11] 探讨了 NDN 网络的分层数据命名方案，希望为 NDN 数据命名提供一种规范。关于命名的规则有很多种，而该系统主要是为互联网直播提供服务，故我针对论文原文的需求，设计了更为易用的命名规则。在本系统中，主要有兴趣包和数据包两种 NDN 数据，我使用/ndn/demo/interest 标记兴趣包，使用/ndn/demo/data 作为前缀标记数据包，对于 m3u8 文件与 ts 文件需要具体区分，命名方案如图 3 所示，这是由系统工作流程决定的，具体流程将在系统具体实现一节中进行介绍。

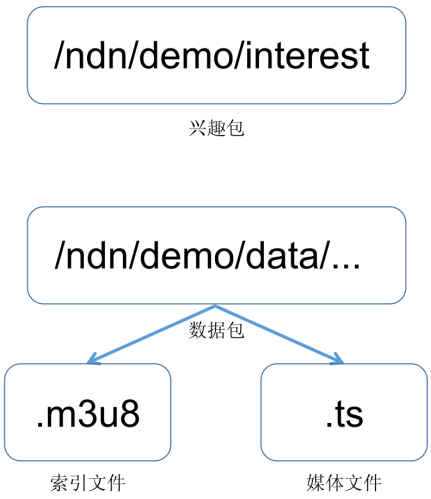

图 3. 系统 NDN 数据包命名规则

3.4 协议转换

视频 HLS 基于 HTTP 协议实现，传输内容包括两部分，一是 M3U8 描述文件，二是 TS 媒体文件。其中，M3U8 是一个具有特定格式的文本文件，记录了一个媒体列表。视频播放器首先对 M3U8 文件进行解析，得到需要播放的媒体文件信息后，按播放顺序播放媒体文件。本系统的 NDN 传输顺序与 HLS 视频的解析顺序类似，数据发布方形成新的 M3U8 文件后，先将该文件传输至直播观看方，直播观看方收到 M3U8 文件后马上对其进行解析，并对各个 TS 媒体文件发出相应的 NDN 数据请求。

流程中设计的所有文件都是使用 HTTP GET 请求来获取，这样就为转化成 NDN 数据传输提供了方便。消费者发出 NDN 数据请求，生产者监听到数据请求后将本地的数据转换为 NDN 方式对消费者做出响应。待 NND 数据到达消费者后，由协议转换模块将数据复原成 HLS 视频格式，供播放器使用。整个协议转换的过程对用户是透明的，用户不需要作任何额外的配置即可享受 NDN 带来的网络优势。

4 复现细节

4.1 与已有开源代码对比

原文并没有公开 PCLive 系统源码，我在对原文理解的基础上对系统进行重新编码复现，成功使用 NDN 网络的方式替换了传统的 IP 网络数据分发部分。其中，我按照自己复现的实际情况重新设计了数据命名规则，并通过持续解析 M3U8 文件并请求缺失的 TS 文件的方式保证直播的连贯性，同时避免发送接收双方之间数据版本号同步的难题。在复现原文的基础上，我具体设计并实现了一个数据发现的功能，将原系统打造得更具实用性与兼容性，并在测试中证明了该功能的可行性。在数据发布部分，我基本按照原文的介绍使用同种现有的软件，包括使用 OBS Studio [12] 进行直播流的推送，使用 SRS [10] 作为视频服务器进行 RTMP 流与 HLS 视频的转换，使用 Nginx [9] 搭建 Web 服务器以供用户打开直播播放页面等。原论文选择在全国多个城市部署真实的网络节点运行 PCLive 系统，本人因条件有限而选择使用建立多台虚拟机进行通信。我必须阐明的是，使用虚拟机进行通信的方式与原文的初衷并无背道而驰，依然可以体现原论文希望证明的直播应用 NDN 化可行性。

4.2 系统具体实现

我首先建立了一个 SRS(Simple Real-time Server) 服务器，用于接收实时推送的 RTMP 直播流，并将直播流转换为 HLS 格式。我将每一个视频段的时间设置为 4s，是因为经过我多次测试，发现 4s 是最佳的数值。假如每一个视频段的时间太短，将导致视频分块太多，从而导致频繁的磁盘读写，影响了总体性能；而视频段太长又会导致用户观看直播的实时性受到影响。我使用了市面常用的直播软件 OBS Studio 用 RTMP 协议的视频流推至 SRS 服务器并实时转换为 HLS 文件。我实现了一个 HLS 向 NDN 的转换功能，可以实时将 HLS 格式的视频文件转换成 NDN 数据包并向 NDN 网络发送。

系统执行的具体过程如下：接收方首先向 NDN 网络发送带有/ndn/demo/interest 前缀的兴趣包，其他节点监听到请求后，检查本地有无该数据，若有，则将对应的 M3U8 文件进行 NDN 化并原路返回。接受方监听/ndn/demo/data 前缀的 NDN 数据，在收到相应 NDN

数据包后将其复原为 M3U8 文件并开始解析，获取到 TS 文件列表后继续向 NDN 网络发送 TS 文件的 NDN 数据请求，并以同样的方式获取对应的数据文件。

网络中的每一个节点分别运行一个用于监听 NDN 网络请求的进程，当接收到来自其他节点的请求时，它会在检查本地是否存在该数据，若有该数据则返回，若没有，则忽略该请求，让网络中有该数据的节点进行返回。当网络中的某个节点需要数据时，该节点主动发出 NDN 请求，等待 NDN 网络返回数据。例如在本文中用户点击观看视频时，转换模块会将视频请求转换为 NDN 请求，并发送到 NDN 网络中，等待其他监听 NDN 网络请求的节点返回数据并在播放器上显示。

因为受到现实条件约束，我无法像原论文一样，在全国多地部署真实的网络节点进行真实实验。因此我部署了三台虚拟机用于实验，系统具体实现架构如图 4 所示。首先，我使用我的主机进行直播流的生成，并及时将 HLS 格式数据文件复制到虚拟机 1 中。由虚拟机 3 发出直播观看请求，虚拟机 1 作为数据响应方，在两台虚拟机中间设置虚拟机 2，其任务仅仅是用于接力数据传输。我通过同样的 NDN 的逻辑方式形成网络，除了网络传播时延以外的其他条件是相同的，而传播时延取决于网络各个节点真实部署时的物理距离以及所使用的传输介质等客观条件，与 NDN 架构本身并无关系。因此我选择在虚拟环境下实现 NDN 数据转发依然能证明应用程序 NDN 化的有效性。

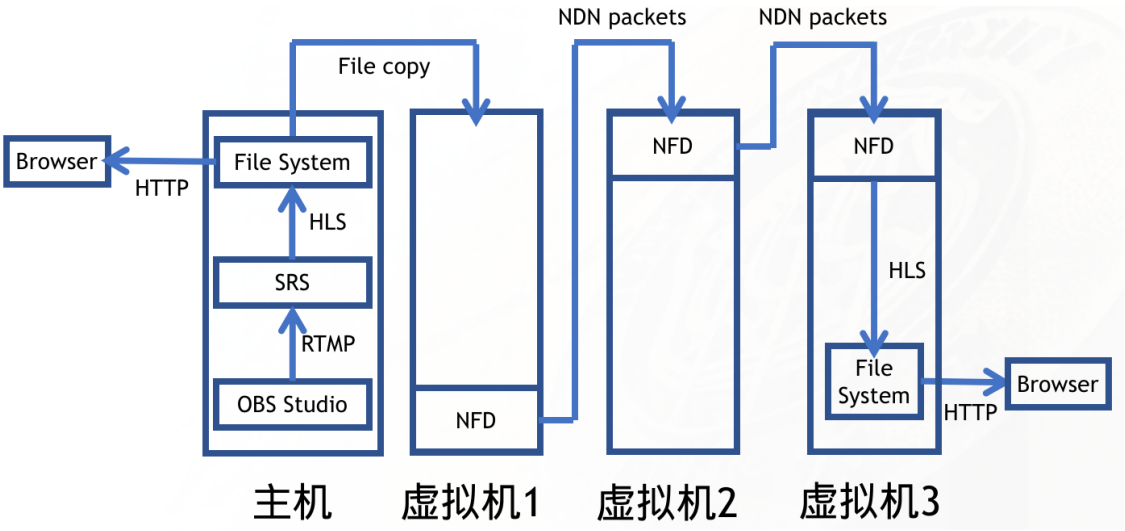

图 4. 系统具体实现架构图

4.3 实验环境搭建

本节用于介绍本次复现的环境搭建过程，包括搭建虚拟机、发布直播、协议转换和搭建用户端视频播放服务器的全过程。

4.3.1 虚拟机环境搭建

Linux 内核提供了容器技术，能提供轻量级的虚拟化能力，并隔离进程和资源。我使用 Linux Container 搭建了三台虚拟机，并使用它们作为同一 NDN 网络下的多个节点，所有虚拟机均使用默认配置进行创建，并配置在同一个子网下，保证可以使用 IP 地址相互通信。虚拟机需要安装 NFD [7] 作为 NDN 协议转换工具。

4.3.2 NDN 逻辑网络构建

在保证几个虚拟机可以使用 IP 地址互相通信的前提下，使用 nfdc route add 命令在虚拟机之间架设 NDN 通信路径，之后的 NDN 数据请求于分发依赖于此时连接的逻辑路径。我在复现时将三台虚拟机通过串联的方式进行连接，未设计复杂的拓扑结构，而这已足够证明 NDN 数据分发的能力以及互联网应用 NDN 化的可行性。我在项目中写了一个构建该链式拓扑结构的功能，编程人员可以使用 run_PCLive -init 命令轻松生成。

4.3.3 RTMP 直播流生成

在生成直播流的环节，我使用流行的直播软件 OBS Studio 将本地的视频文件生成实时的 RTMP 直播流，并将其推送到预先开启的 Simple Real-time Server 服务器中。

4.3.4 RTMP 转换 HLS 视频格式

我在 Simple Real-time Server 中配置了一个将 RTMP 流转换成 HLS 视频数据的服务，以便视频文件可以被 NDN 化并且在互联网中传输，相关配置如下：

```
1         hls_fragment    2;
2         hls_td_ratio    1.5;
3         hls_aof_ratio   2.0;
4         hls_m3u8_file   [stream].m3u8;
5         hls_ts_file     [stream]-[seq].ts;
6         hls_acodec      aac;
7         hls_vcodec      h264;
```

4.3.5 HLS 视频 NDN 化

得益于 NDN 团队的努力，数据文件 NDN 化的功能已经不需要我们另外开发。我们可以轻易地使用 NFD 将视频文件进行 NDN 化并且开启一个进程用于监听即将到来的请求，以便随时对 NDN 请求做出响应，具体的命令是 ndnputchunks，我已经将该命令集成在我复现的 PCLive 系统中。

4.3.6 HLS 文件复原

对于 NDN 数据的复原也同样容易，数据接收者通过 NFD 收到 NDN 数据后与普通的数据文件无异，我们可以直接将其写入磁盘，计算机会根据用户指定格式将其进行解析使用。我将该功能集成在了 PCLive 系统中，并且将其与数据 NDN 化结合成一套完整业务逻辑，不需另外单独配置。

4.3.7 用户端 Web 服务器

对于使用用户来说，只需要打开浏览器下载普通的网页即可使用 NDN 网络直播服务。而根据通常的 C/S 架构，我们需要一个服务器来提供 Web 服务。我根据本次复现的具体需求，

在主机和虚拟机 3 两处分别使用流行的 Nginx 搭建了一个 Web 服务器，为用户提供视频播放网站的服务，Nginx 不需任何额外配置，只提供浏览网页功能。

4.3.8 视频播放网页

网页需要内嵌一个视频播放器，该播放器需要能播放 HLS 格式的直播视频，我使用了一个开源的视频播放器 video.js [13]，用于展示实时直播质量。

4.4 界面分析与使用说明

首先，我们需要使用 SRS 启动一个视频服务器，我设置 SRS 监听的端口为 1935。然后使用 OBS Studio 生成 RTMP 直播流推向 SRS 服务器 rtmp://localhost:1935/并实时将其转换为 HLS 格式，具体步骤可以查看软件官方文档，我不再赘述。

主机将不断生成最新的 HLS 文件，我们需要将该文件迁移进虚拟机 1 中，可以在主机简单执行 run_PCLive -d 命令达到此效果，此时会出现“主机正在向容器分发数据”的提示。

在虚拟机 1 中执行 run_PCLive 命令，用于监听来自 NDN 网络其他节点的数据请求，执行命令后会出现“正在监听请求”的提示。

在虚拟机 3 中执行命令向 NDN 网络发送数据请求，并接收数据。以命令 run_PCLive -i movie 为例，系统会利用该名字根据设计好的命名规则构建 NDN 请求发送到网络中，并等待对应直播数据返回。

此时可以打开浏览器访问部署于虚拟机 Nginx 服务器的 HTML 页面，观看使用 NDN 网络作数据分发的视频直播，网页仅需包含一个普通的 HLS 视频播放器，如图 5 所示。

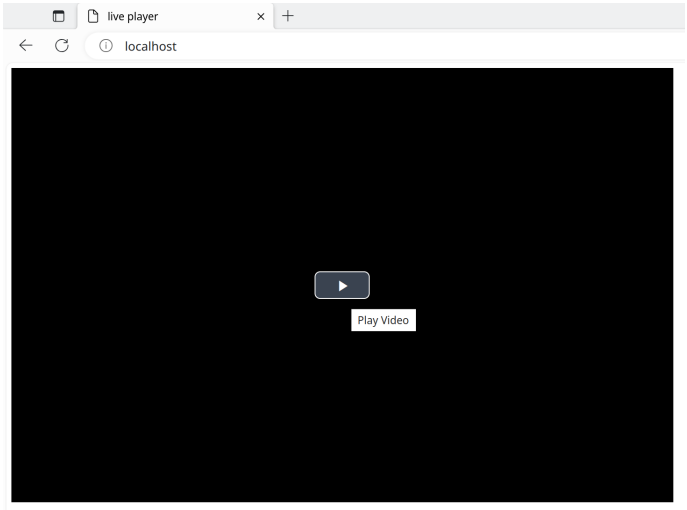

图 5. 观看直播页面

每当系统运行 15 分钟后，系统会计算这一时间段下已传输所有数据的平均速度，并记录在请求方本地的文件系统内，可以查看系统运行期间 NDN 网络的数据分发效率，图 6 是系统执行时记录下来的真实传输效率数据。

round	speed
1	21.38907232
2	18.50588258
3	20.90660755
4	18.8219206
5	19.21700308
6	21.75727477
7	19.41129291

图 6. 数据传输速度记录

4.5 创新点

我在复现原文的基础上设计并实现了数据发现功能，并对网络数据名称进行有效管理。在此方案中，当某一个数据的生产者成功生产数据后，应当向 NDN 网络广播数据名，以告知其他节点可以请求此数据。其他节点监听到新数据发布后，将该名称存在本地，以便以后根据该名称向 NDN 网络发送兴趣包，图 7 展示了数据发现的过程。

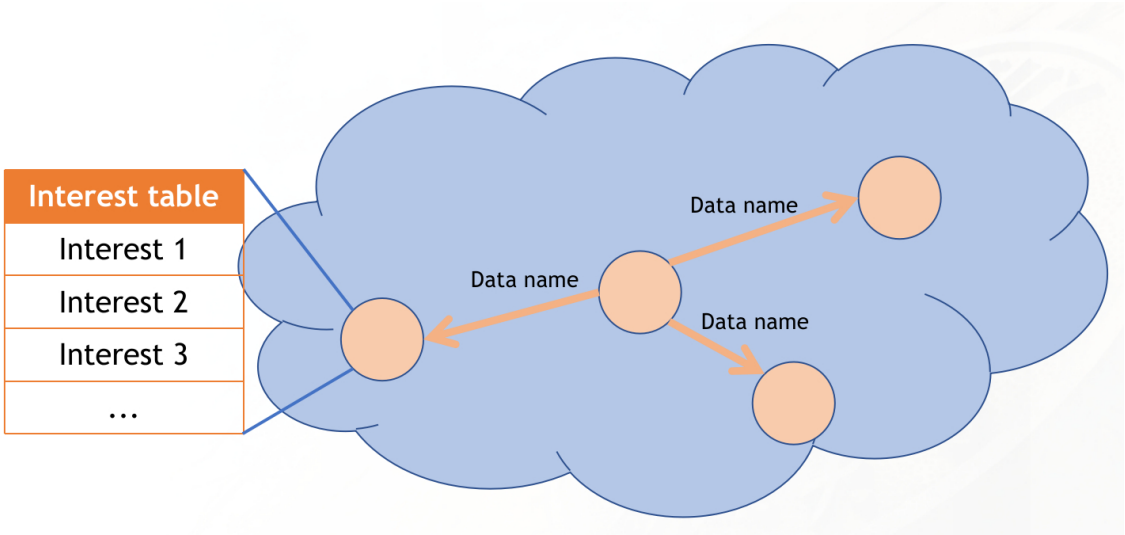

图 7. 数据发现流程

然而，我在设计数据发现功能的过程中发现该功能依然不够可靠，节点只能获取到生成的数据名，而不对这些数据名进行有效管理。假如某个数据在整个网络中消失，节点已经不能检索到该数据，此时应该将该数据名取消。因为 NDN 网络数据缓存的特殊特性，数据可能在网络中的任意一个节点存在，如果仅仅是某一个节点内的数据消失并不能说明该数据已经在全网络内消失，所以数据源驱动的方式不可行。我使用请求者驱动的方式，图 8 表示当请求者请求该数据并经过若干次尝试失败后，该数据名应当在数据接收者本地清除，表示某数据已在全网络内消失，图中红色字体为欲请求但无法获取对应数据的数据名称。我成功实现了该数据名管理的功能，在实际测试中显示能达到预期效果。

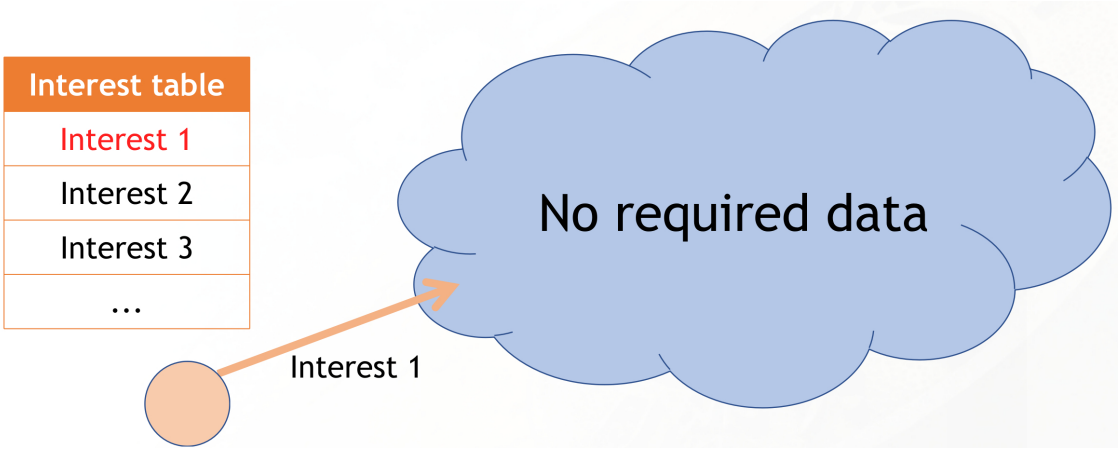

图 8. 数据名管理功能图

5 实验结果分析

在长时间系统运行的实际直播效果观察中，NDN 数据命名网络可以良好地完成实时数据分发任务，直播几乎没有出现卡顿情况，直播效果如图 9 所示。

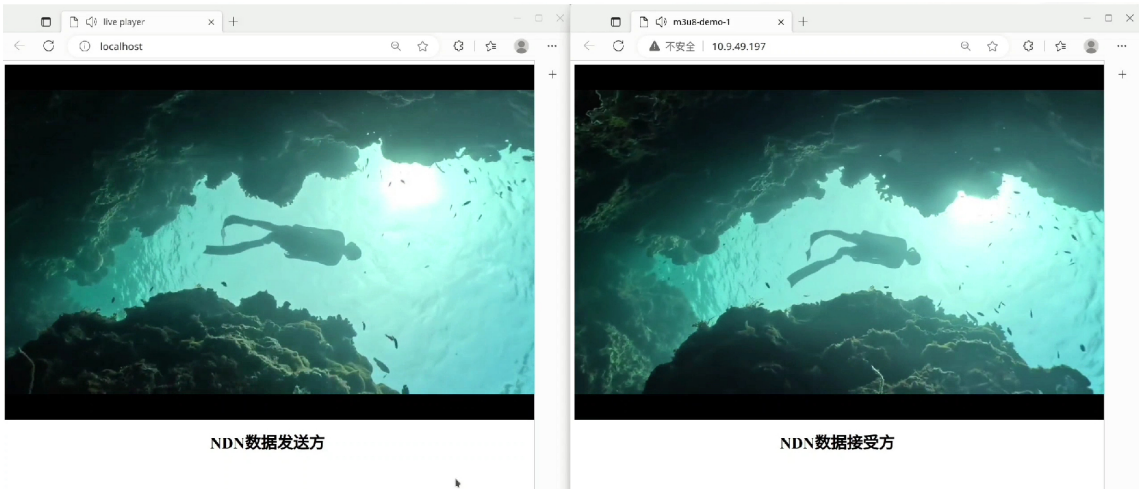

图 9. 直播系统实际效果展示图

请求节点在发送 NDN 数据请求时开始计时，到 NDN 数据成功返回后终止计时，由接受者计算该数据的平均接收速度，以此方式来监测实时的直播数据接受速度。我连续运行该直播系统，并且记录下所有 HLS 数据文件的传输速度，以 15 分钟作为一个周期，连续运行了 50 轮，并计算出每个周期内传输的平均速度，以此来评定 NDN 网络直播的传输速率。实验过程的数据传输速度变化由图 10 展示，可以看到，传输速度基本在 18 至 22Mb/s 之间波动，以足以支持市面流行的 1080p 高清视频传输。此外，我在代码中将数据拉取失败后等待下一次尝试的时间设置为 300ms，而这 300ms 对于 1MB 左右的 TS 文件来说是很长的等待时间，这严重影响传输速度的计算。我认为真正的传输速度远远没有这么慢，下一步还应该继续优化代码，使多次请求之间的等待时间更短，或是继续完善服务逻辑，使传输速度计算得更准确。

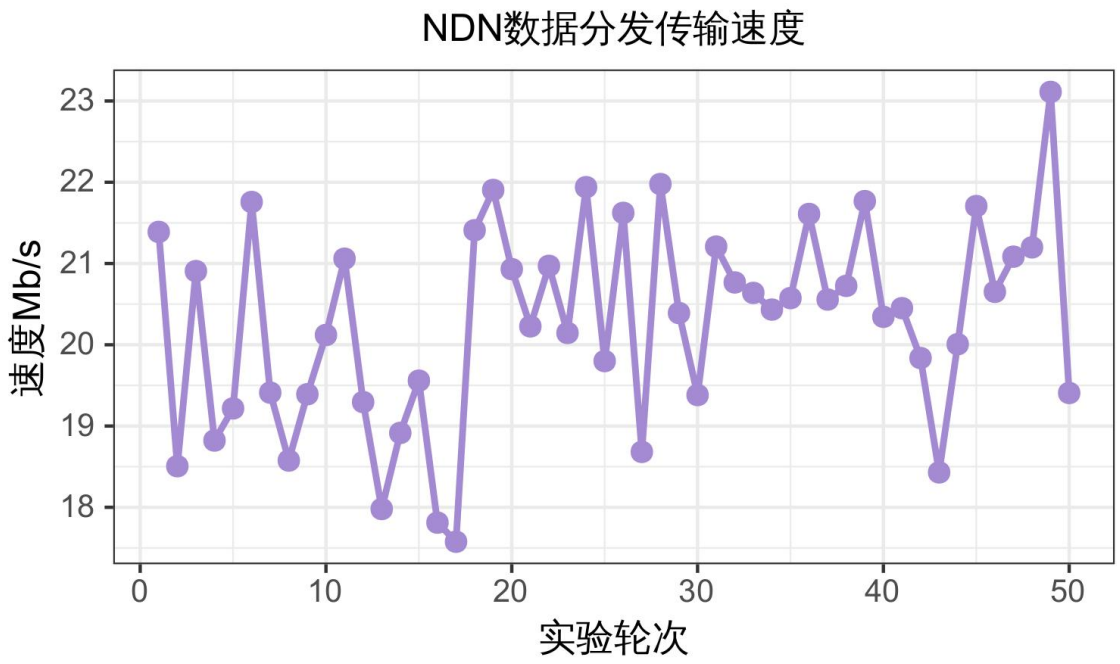

图 10. 数据分发速度

6 总结与展望

原文的工作主要围绕互联网直播应用 NDN 化展开，直接在视频播放器中嵌入 NDN 转换模块来达到 NDN 化的效果，省去了繁杂的代理配置流程，实验证明这种方法简单有效，可以说明 NDN 化在直播视频应用上的可行性。并且该论文是一个兼容现有直播系统组件的良好典范，仅仅替换了直播系统的数据分发部分，而重用了数据发布部分。

在本次论文复现过程中我收获颇丰。首先，我在实际的系统复现过程中，更加全面地认识了 NDN 数据命名网络的全面架构。其次，在系统开发过程中我遇到了许多阅读论文时无法注意到的实际困难，我在导师的指导下将这些实际开发难题一一解决，让我的系统开发能力得到较大提升。最重要的一点是，作为 NDN 数据命名网络的初学者，我对该领域的认识依然浅薄，论文复现让我有机会拓宽科研视野，了解到目前该领域的最前沿的研究工作进行到何种程度，面临着哪些挑战，有哪些工作是值得我下一步开展的。

本文在数据安全方面只是简单地要求消费者信任 PCLive 的数据生产者，并完全信任具有特定前缀的数据包，使用数据签名的的方式确保数据的可靠性，而没有讨论到数据的保密性。我认为在 NDN 网络中，数据包携带的消息可能包含重要内容，而数据可能受到任何一个网络节点获取并缓存，因此，数据泄露的风险明显增大。更何况，在目前设计的命名规则下，数据包名字本身已经揭示出数据本身的内容，这使得有心人容易推断出用户的部分敏感信息。其中一种思想是对数据进行加密，这样可以保证只有对应秘钥的用户才获才能获得数据，但这导致 NDN 网络的缓存特性受到削弱，丢失了该架构天然的优势。这是 NDN 网络继续解决的一个问题 [17]。

另外，原文仅专注于数据分发的功能，没有关注内容缓存在 NDN 网络中的作用。而缓存作为 NDN 网络的一个重要特性，要发挥 NDN 网络架构的数据分发最大潜能，缓存功能应当被精妙设计，因此缓存策略也可以是下一步研究的方向，目前已有一些专家研究这方面的工

作，包括 Institut Teknologi Bandung 等人研究改变内容存储容量大小对网络性能的影响 [1]，Yan-Hong Liu 团队更是将 NDN 内容缓存机制与网络拥塞相结合 [15]，提出一种拥塞解决方案：当拥塞发生时，使用内容存储 CS 动态扩展发送队列的缓冲区，降低数据包和相应兴趣包的转发速率，以缓解网络拥塞。

我在复现论文过程中注意到的问题另一个问题是，在 NDN 网络中，生产者提供的内容可能被缓存在任何一个路由器的缓存中，而一旦内容进行了更新，要及时告知所有拥有该数据的用户是非常困难的。有人提出使用版本号来代表数据最新版本，然而在我复现过程中，我发现版本号本身也需要在网络中同步，而同步版本号本身的难度较高，且网络开销较大。在现代瞬息万变的世界中，频繁更新的数据比比皆是，假如每一个更新的数据都需要在全网络范围内进行广播，并替换原来的缓存，这会导致网络内经常充斥着各种更新数据，随之而来的是网络拥塞导致整体效率不佳等问题，这样的现象应如何避免？我认为这是值得探讨的问题。

因为其端到端连接的特性，传统互联网的效率可以使用吞吐量、带宽利用率等指标正确衡量，但是这些指标在 NDN 数据命名网络中的某些场景下似乎不能很好地展示网络真实状况。我认为需要针对 NDN 网络设计一套新的测量指标，这样利于 NDN 的长远发展。

本文使用 NDN 实现视频流数据分发功能并证明其可行性，除此之外我认为还有更多的协议可以被 NDN 化并且比传统架构展现出更好的效果，例如 OIDC 身份验证和授权协议、WebRTC 实时通信以及 WebSocket 实时全双工长连接通信都是值得被 NDN 化的应用，这些协议被广泛应用于视频会议、在线游戏、身份验证等覆盖我们生活工作的多个方面，可以继续测试 NDN 化在这些常用协议上的表现。这些都是我认为以后可以开展的相关工作。

参考文献

[1] Muhammad Najib Dwi Satria and Sigit Haryadi. Effect of the content store size to the performance of named data networking: Case study on palapa ring topology. In *2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA)*, pages 1–5, 2017.

[2] Chavoosh Ghasemi, Hamed Yousefi, and Beichuan Zhang. Internet-scale video streaming over ndn. *IEEE Network*, 35(5):174–180, 2021.

[3] Peter Gusev and Jeff Burke. Ndn-rtc: Real-time videoconferencing over named data networking. In *Proceedings of the 2nd ACM Conference on Information-Centric Networking*, pages 117–126, 2015.

[4] Van Jacobson, Diana K Smetters, Nicholas H Briggs, Michael F Plass, Paul Stewart, James D Thornton, and Rebecca L Braynard. Voccn: voice-over content-centric networks. In *Proceedings of the 2009 workshop on Re-architecting the internet*, pages 1–6, 2009.

[5] Derek Kulinski and Jeff Burke. Ndn video: Live and prerecorded streaming over ndn. *The NDN Project Team, Tech. Rep*, 2012.

[6] Teng Liang and Beichuan Zhang. Enabling off-the-grid communication for existing applications: A case study of email access. In *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2018.

[7] NFD. https://docs.named-data.net/NFD/current/.

[8] Davide Pesavento, Giulio Grassi, Claudio E. Palazzi, and Giovanni Pau. A naming scheme to represent geographic areas in ndn. In *2013 IFIP Wireless Days (WD)*, pages 1–3, 2013.

[9] Will Reese. Nginx: the high-performance web server and reverse proxy. *Linux Journal*, 2008(173):2, 2008.

[10] SRS(Simple Realtime Server). https://github.com/ossrs/srs, 2013.

[11] Mohammad Shahrul Mohd Shah, Yu-Beng Leau, Zhiwei Yan, and Mohammed Anbar. Hierarchical naming scheme in named data networking for internet of things: A review and future security challenges. *IEEE Access*, 10:19958–19970, 2022.

[12] OBS Studio. https://github.com/obsproject/obs-studio.

[13] videojs. https://github.com/videojs/video.js.

[14] Lijing Wang, Ilya Moiseenko, and Lixia Zhang. Ndnlive and ndntube: Live and prerecorded video streaming over ndn. *NDN, Technical Report NDN-0031*, 2015.

[15] Feng-Hua Huang Yan-Hong Liu and Hua Yang. A fair dynamic content store-based congestion control strategy for named data networking. *Systems Science & Control Engineering*, 10(1):73–78, 2022.

[16] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, kc claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. Named data networking. *SIGCOMM Comput. Commun. Rev.*, 44(3):66–73, jul 2014.

[17] Zhiyi Zhang, Yingdi Yu, Haitao Zhang, Eric Newberry, Spyridon Mastorakis, Yanbiao Li, Alexander Afanasyev, and Lixia Zhang. An overview of security support in named data networking. *IEEE Communications Magazine*, 56(11):62–68, 2018.

[18] Zhenkai Zhu, Sen Wang, Xu Yang, Van Jacobson, and Lixia Zhang. Act: audio conference tool over named data networking. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, pages 68–73, 2011.