ISSN 2096-742X
CN 10-1649/TP

文献DOI：
10.11871/jfdc.issn.2096-742X.2020.03.007

文献PID：
21.86101.2/jfdc.2096-742X.2020.03.007

页码：75-86

开放科学标识码（OSID）

基于P4的HTTP网内缓存方案及其实现

詹昱辰*，冯巍巍，谭小彬

中国科学技术大学信息科学技术学院自动化系，安徽 合肥 230022

摘　要：【目的】在当今的网络交互中，最常用到的是HTTP协议，并且HTTP的使用量仍在迅速增长。然而，HTTP的通常传输仍然需要基于基础的TCP / IP协议栈，这限制了其解决当前因特网上的问题的能力。在用户浏览网页时，网页的响应速度对用户的访问体验来说是很关键的。本文旨在给出一种可行的使网页浏览响应速度提升的方案。【方法】在本文中，我们提出并测试了一种解决方案，借鉴ICN（信息中心网络）协议的思路，通过使用P4（协议无关的可编程数据包处理器）语言，在HTTP协议中实现网络内缓存。首先，我们提出一种数据包转换机制，用于普通数据包与特制数据包的来回转换，其次采用P4语言程序在转发路由器中实现ICN传输过程。【结果】为了评估成果，我们用虚拟机设计搭建了一个网络拓扑。经过测试，与使用普通的HTTP协议时相比，本文中的方案使P4交换机能够缓存HTTP内容响应，聚合相同的请求，改善网络的性能。【结论】于是得出在提升用户浏览网页时的体验上，本文提出的方案是有效的。

关键词：HTTP；ICN（信息中心网络）；P4语言；网络内缓存

A Solution for HTTP In-Network Caching Based on P4

Zhan Yuchen*, Feng Weiwei, Tan Xiaobin

Department of Automation, School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui 230022, China

Abstract: [**Objective**] HTTP is the most used protocol in network traffic these days, and the use of HTTP is still growing rapidly. However, the normal transmission of HTTP requires the underlying TCP/IP protocol stack, which limits its ability to solve problems in the current Internet. The response speed of a web page is critical to users' experience when they browse the web pages. This paper aims to give a solution to improve the response speed of web browsing. [**Methods**] In this paper, we propose a solution to implement in-network cache in HTTP protocol using the idea of ICN (Information Centric Network) protocol and the P4 (Programming Protocol-Independent Packet Processors) language and conduct evaluation of the proposed solution. Firstly, we propose a packet conversion mechanism for converting a custom packet into a special packet and converting it back. Secondly, we adopt P4 language to implement the ICN transmission in the forwarding router. [**Results**] In order to verify the functionality of our design, we set up a network topology among multiple virtual machines and verify the network performance improvements when using the ICN protocol in HTTP transports. Evaluation results show our solution that enables P4 switches to cache HTTP content responses can aggregate the same requests and improve network performance. [**Conclusions**] Therefore, the solution we proposed in this paper is effective for users in web page browsings.

Keywords: HTTP; information centric network; P4 language; in-network cache

基金项目：国家自然科学基金（61673360）

*通讯作者：詹昱辰（E-mail: zyc233@mail.ustc.edu.cn）

引　言

在信息量爆炸式增长的互联网时代，人们几乎每天都在浏览网络且基本上是通过浏览器。互联网用户每天都会浏览大量的网页信息和视频信息，而目前大约80%的互联网流量都是视频。当浏览者在网络上观看视频时，大量的流量都是以HTTP流量的形式流动的。因此每天，都有巨量的网页信息或视频数据通过HTTP传输，这就促使多数网站都要投入大量的工作来优化HTTP流量的传输，以防止网站宕机。因此，优化当前的HTTP传输通信，从公司的盈利能力或是用户的浏览体验来看，是十分必要的。

HTTP（超文本传输协议）[1]是一种广泛使用的应用层协议，且针对它已经做了很多优化。例如，各种HTTP设施（例如CDN，HTTP代理和WEB缓存）早已部署在因特网之中[2]，使现有的问题得到部分缓解。但是，尽管现在已经有许多公司提供了大量的基础设施，在一定程度上改善了HTTP的使用体验，当前的HTTP工作上仍然存在着许多瓶颈：

（1）HTTP依赖于基础的TCP/IP体系结构，这导致HTTP具有TCP/IP体系结构的窄腰特性。TCP/IP体系结构中的通讯基本单位，是由IP地址所标识的两个端点之间的端到端信道。也就是说，TCP/IP体系结构的窄腰正是IP地址，而且这样的窄腰仍会限制住解决当前网络问题的能力[3]。

（2）HTTP是应用层协议，传输层需要由TCP协议来完成，这就需要端到端的网页数据传输。但是，对于静态数据例如网页，更改频率并不是特别快，因此这种端到端传输在内容分发网络的概念被提出之后，在该类型情景下的适用性不强。因为这势必导致相同的内容在网络中被反复传输。

（3）尽管HTTP传输模式与ICN传输类似，用兴趣包来检索数据，但是在传输的实现细节上，仍存在很多差异。例如，HTTP的下层传输协议是TCP，而TCP协议使用PUSH模式来传输数据。相反，在ICN网络中，兴趣包被用于发送以检索相应的数据，这称为GET模式。理所当然地，HTTP的请求和响应方法很容易地为我们带来一个优化想法：移植ICN传输模式到HTTP，允许网络中的转发路由器缓存数据并汇总请求[4]。然而TCP的PUSH模式给这个想法增添了不少麻烦。

用户对他们访问网页时的体验的需求，驱使我们利用ICN网络传输的思想来改善HTTP请求过程。由于网内缓存和请求聚合通信特性的存在，ICN传输特别适合部署在HTTP请求过程中用于提升网络性能。同时，根据P4语言[5]的能力，用户可以在可编程交换机设备上自定义转发操作；根据P4语言的特性，用户可以产生具有ICN和IP包特征的数据包，并且可在同一交换设备上兼容地工作[6]。另外，交换设备可以根据不同的传入数据包类型采用不同的转发操作。

尽管上面提到了很多优势，但是想要在HTTP请求方案中使用P4语言实现ICN传输，由于TCP/IP体系结构和ICN之间的差距，仍具有挑战性。首先，由于协议的不同，两者所使用的数据包是不兼容的，需要寻求数据包的转换方法。其次，如何利用P4语言编写简单快捷的转发过程算法也是一个需要思考的问题。

在文献[19]和[20]中，是用OpenFlow尝试在TCP/IP上实现了ICN思路的缓存与负载均衡操作，但方案与操作细节较为复杂，所用到的几个流表参数很多。而P4作为正在发展中的一种优秀的语言，相比OpenFlow的协议对数据面的控制与操作更为便捷，尤其是在数据包的转发方面。这正是本文选用P4作为实现工具的原因。

数据传输过程，网络中的数据包的处理和转发设备的操作是三个主要的研究方向。在本文中，我们采用P4语言在转发交换机上实现ICN转发操作，并提出一种用于将HTTP数据包和ICN.p4数据包相互转换的数据包转换规则。本文的主要贡献如下：

（1）提出了一个数据包转换规则，该规则可以

部署在 HTTP 客户端和 HTTP 服务器上，并通过代理应用程序来保障该规则顺利工作。代理应用程序添加 ICN 特征的字段到要发送的数据包包头，将其转换为 ICN.p4 数据包；相反地，它也会删除其中 ICN 特性的字段（如有必要），使数据包转换回到 TCP / IP 数据包。

（2）在本文的方案中，通过使用 P4 语言，在可编程的转发交换机上实现了 ICN 转发操作。该可编程交换机可以实现 PIT，CS 和其他仅存在于 ICN 网络转发设备中的模块。在 P4 交换机加载 P4 程序之后，就可以完成 PIT 检查，数据包缓存等过程，然后将数据包转发到下一跳。以上所描述的整个处理流程就是 ICN 网络转发设备的转发过程。

（3）为验证提出的方案，我们在虚拟机上搭建了一个简单的网络拓扑结构。在测试中使用 HTTP 协议浏览网页，同时收集流量信息以及网络中的其他实验数据，以量化展示网络性能的提高。

1　相关工作

在本节中，我们简要介绍 ICN 和 P4 语言和 HTTP 协议的现状。

1.1　ICN 和 HTTP 概述

ICN（Information Centric Network，信息中心网络）被认为是可以取代 IP 架构的下一代网络架构，对于信息中心网络的实例，已有的例如 NDN（Named Data Network，命名数据网络）/ CCN (Content Centric Network，内容中心网络)/ PURSUIT(Publish-Subscribe Internet Technologies) 等。虽然他们实施细节不同，但设计概念是类似的，NDN 就是其中一个典型的案例。在 NDN 中，数据传输是通过两种数据包来完成的：兴趣包和数据包 [7]。兴趣包携带着请求内容的名称，同时数据包携带的是与兴趣请求相对应的内容。在这个过程中有三个主要的状态表：CS (Content Store，数据缓存)，PIT (Pending Interest Table，待处理表)，FIB（Forwarding Information Base，转发路由表）。CS 可以缓存经过的数据包 [8]；PIT 的作用是留下标记来帮助数据包返回到消费者方；FIB 用于路由兴趣包，同时 FIB 条目是由路由协议所生成的。要注意的是上面三个状态表的表项都是名称。

HTTP 是一个简单的请求 - 响应协议，通常在 TCP（一种应用程序层协议）上运行。当一位 Web 用户打开其 Web 浏览器时，该用户将间接地使用到 HTTP。到目前为止，最新版本的 HTTP 是 HTTP / 2，是于 2015 年 5 月发布的。HTTP 指定了消息的格式，它们都在 TCP 协议的有效载荷之内。在图 1 中，展示了 HTTP 的请求和响应格式。

1.2　P4 语言现状

斯坦福大学的 Nick McKeown 教授首先在 2014 年设计并提出了数据平面特定领域编程语言——P4 [5]，以充分解放数据平面的编程能力，现在已经在学术界和工业界被广泛认知。一方面，基于可编程设备

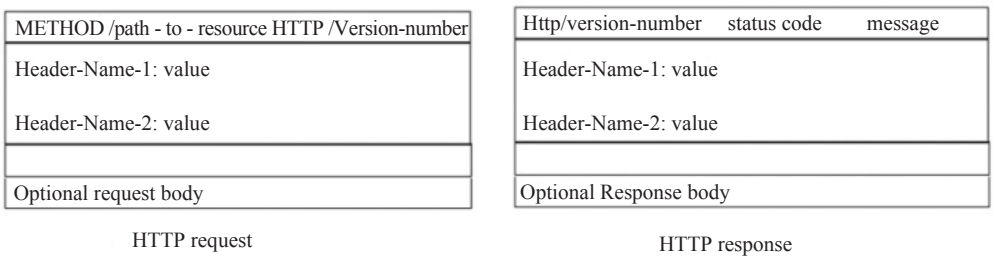


图 1 HTTP 请求格式和响应格式

Fig.1 HTTP request and response packet format

的可定制功能，它可以快速实施和验证一些新的网络架构，功能和协议，极大地加速了网络的进化与创新；另一方面，基于可编程设备的高性能特征，传统上灵活但性能低下的中间件工具，例如防火墙，负载平衡以及其它一些更简单的网络功能，就可以被转移到到可编程数据面上，从而实现显著的性能提升。

自 2014 年提出 P4 的概念以来，P4 语言一直在蓬勃发展。同年，就提出了 P4_14，随后 P4_16 版本在 2016 年被发布。根据 P4 官方网站，大多数开发中的新协议的小演示，现在都在软件交换机上完成。只要软件交换机 bmv2 被安装在所有 Linux 接口的机器上，这台计算机就可以用作可编程交换机。尽管有许多公司已经开发了支持 P4 语言的可编程硬件交换机 [9]，但本文中之后提到的所有环境，都是搭建在 Linux 之上的软件交换机。

在 P4 语言的提出之后，相关研究非常活跃，许多小型演示的尝试旨在快速实现一些非常创新的协议和转发策略。在文献 [10] 中，实现了网络内缓存的想法，这使得转发交换机可以缓存数据包内容。只要重新设计数据包的格式，在 P4 交换机上设计转发过程，缓存，获取，更新数据内容或其他一些操作，都可以根据数据包中的某些特定字段来实现。另外，根据交换机上数据包的统计信息，还可以提出一种新的缓存策略。

使用 P4 语言，许多复杂的转发协议可以在数据平面上轻松实现。在文献 [11] 和 [12] 中，ICN 在转发方面的一些优势正在逐渐由 P4 语言实现，但由于要求所有转发节点都是 P4 交换机，所以大多数还无法与当前的网络协议或应用相结合。在分阶段部署转发设备的前提下，是无法保证所有设备都是 P4 交换机的，因此仍然需要改进。尽管 ICN 传输的思想在文献 [11] 中已经基本实现了，但该文章仍明确指出：缓存模块功能尚未完成，并且文中的实现仅适用于完全配置了 P4 路由器的链路。

1.3 HTTP 的网内缓存

如果能在 HTTP 传输过程中，存在中间节点的数据缓存，就能够大大减少传输的流量，提高访问效率。这就是 HTTP 的网内缓存。

现有的缓存方案主要分为缓存整个文件与记录文件放置位置两种。直接缓存整个文件开销太大，又需要占用大量存储资源，并非合适的选择。记录文件位置可以方便内容的集中管理，提供控制和管理网络资源的机会。更新内容位置数据库的开销相对于存储文件低了很多，但就存储位置来说灵活性不如就地缓存高。

2 架构描述

本节描述了我们提出的解决方案：受 ICN 的思想启发，通过 P4 在路由器转发过程中实现网络内缓存，实现对 HTTP 的访问效果提升，减少网络冗余流量。

2.1 动机和概述

如上一节所述，研究 HTTP 协议的优化是很有意义的，但是解决方案的方向受到了端到端通信这一基础机制的限制。端到端的通信会导致许多相似的内容在复数链接中被反复传输，这产生了很多冗余的网络流量，提高了网络的负担，而为优化网站的浏览而设置的代理服务器相对来说可是很昂贵的。尽管 ICN 网络架构中“网内缓存”的提出为优化的研究思路带来了一些好的想法，但由于 ICN 和 TCP / IP 之间的差异，仍然很难在实际部署中与以前网络的体系结构相兼容。但是，P4 语言的发展应用拓宽了解决问题的思路视野，使 ICN 的特性在 HTTP 协议中运转不再那么困难。所以，依然借助 ICN，我们提出了一个使用 P4 语言来帮助实现 ICN 的网内缓存以优化 HTTP 请求过程的想法。有了网内缓存的存在，就能够大大降低网络中的冗余流量。

我们的基本思路是，通过部署一些可编程的路由设备，实现 ICN 网络内缓存 [13] 与请求聚合等等功能，更进一步，实现 ICN 传输的模式，而其中最关键的部分正是网内缓存。并且，希望可以与现有的路由设备和网络协议相兼容。为了实现这个想法，在这里首先要提出一个数据包转换代理的模块，它一方面具有代理服务器的功能，另一方面实现数据包的转换。

这样的数据包转换代理分别部署在 HTTP 客户端与服务器端的接口处，使每个进入网络传输的数据包接受转换，每个离开网络到达客户端或服务端的数据包也会接受转换。

具体来说，HTTP 客户端发送的每个请求均由代理转换，代理将转换后的数据包转发给路由寻找答复；反之，响应也通过这样的代理应用程序进行传递。服务器上的代理应用程序会转换整个 HTTP 响应。由于转换前后的数据包结构不同，在传输时需要重新切片。

路径上在代理应用程序之后的每个响应数据包都包含名称，即 HTTP 响应的 URL 和相应的段号。因为转换后的数据包包含附加字段，可以轻松地利用可编程转发设备上的 P4 语言程序，设计 ICN 传输中的转发操作，包括但不限于缓存与聚合。每当一个数据包到达时，可编程转发设备需要将其与 PIT，CS，FIB 等匹配比对，这些表需要执行的常规操作是与 ICN 类似的。存在不同的操作例如有在 PIT 中存在聚合时的多播分发，由于某些中间转发节点可能不是经 P4 编程的转发设备，此处需要添加目标 IP 地址的信息。另一个是我们使用 IP 表而不是 FIB，并且表项由控制面来发送。

根据以上描述，代理应用程序将转换后的数据包发送到网络进行路由，这些数据包可以根据名称在可编程交换机上进行缓存和聚合。该设计中的代理应用程序和数据包传输过程如图 2 所示。

另一方面，在图 2 中可以看到，网络中存在着一部分 P4 路由器，正是在网络中部署好的可编程路由器。相对于原来的网络，也就是将普通路由器进行了替换，使之可以执行更多的功能。要部署在实际的网络中的话，可以采用安装了 P4 环境的树莓派或其他可配置环境的智能路由器来担当可编程路由器的角色。

2.2　包转换

为了使转发设备能够依照 ICN 转发过程和谐地工作，将数据包中的相关操作归档是很有必要的，例如名称、序列号、操作等。图 3 显示，ICN-TCP / IP 协议的数据包格式，同时包含着 TCP / IP 和 ICN 的特性，也就是说，数据包格式为 TCP / IP 和 ICN 的集成。虽然本质上是 IP 数据包，但是它也拥有一些 ICN 字段，并且是通过使用 P4 语言来实现 ICN 协议的，因此我们将其称为 ICN.p4 数据包。由于此数据包的独特结构，在可编程交换机上，可以轻松

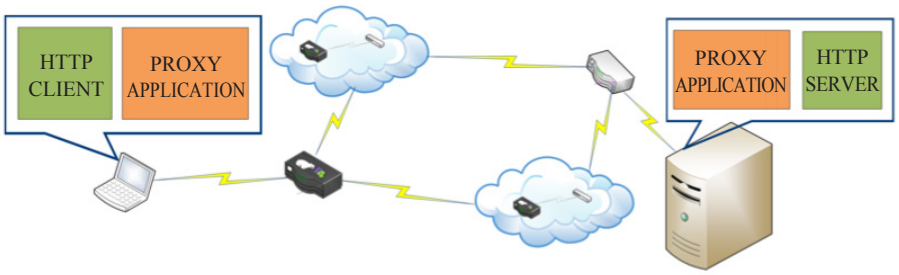


图 2 结合代理应用程序的 HTTP 请求体系结构

Fig.2 HTTP request architecture with Proxy Application

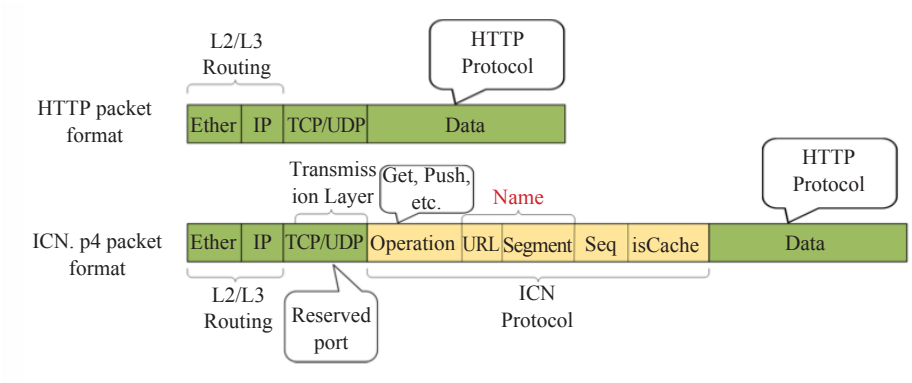


图 3 HTTP 和 ICN.p4 数据包格式

Fig.3 HTTP and *ICN.p4* Packet Format

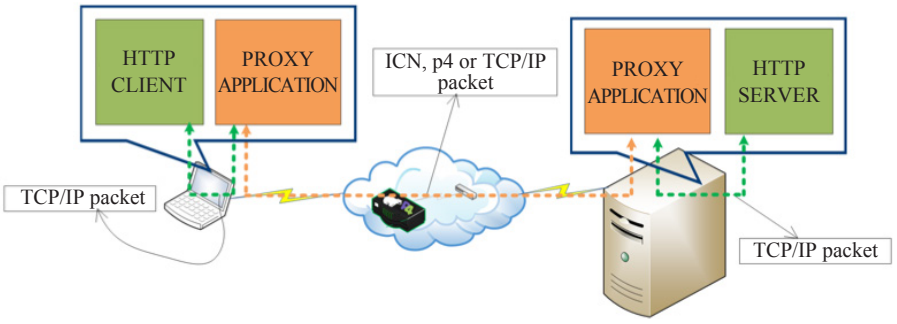


图 4 代理应用程序转换数据包

Fig.4 Proxy Application converts packets

地使用 P4 语言实现 ICN 转发过程和 TCP / IP 转发过程。ICN.p4 数据包中的主要标头字段是 OP，URL，Segment，Seq 和 isCache。OP 代表操作，表示它是一个 Get 包还是 Push 包，它等同于 ICN 中的数据包类型，例如兴趣包和数据包。所以我们称 OP 字段为 Push 的是 ICN.p4 内容数据包，OP 字段为 Get 的是 ICN.p4 兴趣包。由 URL 和 Segment 字段组成的名称部分类似于 ICN.p4 名称，但是区别是，在此方案中并不使用名称进行路由，而是用于匹配判断。isCache 字段用于决定是否要缓存数据包数据。

ICN 协议字段位于 TCP 有效负载之内，并且有一个为此数据包保留的专用的 TCP 端口。P4 交换机使用该特殊端口来调用 ICN 数据包处理逻辑，其他端口的数据包则由 P4 交换机视为普通 IP 数据包进行转发，这样其他的交换机就不需要理解 ICN.P4 数据包的格式，P4 交换机也不需要将所有进入的数据包视为普通 IP 数据包。这样一个保留的 TCP 端口正是被用于确定数据包是 TCP / IP 包还是 ICN.p4 数据包，这同样会应用在代理应用程序当中。在客户端和服务器端均设置了应用程序代理 [14]，使用此代理，可以将 HTTP 数据包转换为 ICN 数据包，也可以将其转换回去。一旦 HTTP 客户端和服务器都使用代理应用程序，对于客户端，代理应用程序将被视为 HTTP 服务器，反之，对于服务器端，代理应用程序被认为是 HTTP 客户端。对于代理应用程序，它扮演的角色以及它在通信过程中所达成的功能如图 4 所示。因此，当 ICN 数据包在网络中流动时，只要使用了代理应用程序，就会由代理应用程序完成 TCP

和 ICN 数据包的转换。这奠定了实现网内缓存，请求聚合以及 ICN 的其他特性的基础。

2.3 转发过程

从代理应用程序发出的数据包会进入网络，并且开始传输，剩下的工作就是逐跳进行数据包转发。而顺利进入网络的数据包均是转换后的数据包，即 ICN.p4 数据包。P4 交换机上的转发操作与数据包的处理流程都与 ICN 是类似的，完整的 HTTP 请求流程如图 5 所示。

当数据包传输的路径上存在一个 P4 交换机时，就有必要检查 P4 交换机中的状态表，例如 CS 和 PIT。如果 CS 表命中了（根据数据包的名称进行匹配），则 P4 交换机会直接使用网络内缓存进行响应。另一方面，P4 交换机检查 PIT，来判断是否要汇总请求。如果名称已经存在于 PIT 中，P4 交换机就更新 PIT，然后丢弃数据包；如果没有，那么就比对 IP 表并转发数据包。因为 P4 交换机是分阶段部署的，所以并不能保证所有链路上都是 P4 交换机。也就是说，并非所有中间转发节点都是有状态的，是记录了状态表的。因此，根据 IP 地址来进行路由才是可靠的。

于是，当 ICN.p4 数据包在网络上转发时，存在有两种可能性。第一种情况，数据包由 P4 交换机来执行转发，P4 交换机会基于预留的 TCP 接口来判断传入数据包是否是一个 ICN.p4 数据包。接下来，如果它是一个 ICN.p4 数据包，则该交换机将调用由 P4 语言书写的 ICN 数据包处理程序。如果不是，则交换机就将传入数据包视为普通的 IP 数据包，并将其转发出去。而在另一种情况中，是由普通交换机来执行 ICN.p4 数据包的转发。由于 ICN.p4 协议字段位于 TCP 有效负载内，因此对于仅执行 L2 / L3 层转发的的普通路由器来说，是不可见的。

2.4 P4 交换机的细节

经过设计的整个 HTTP 请求过程已经在上面完成了讨论，本节将介绍具体的操作以及每个 P4 交换机的实现细节。对于一台转发设备，其转发过程如算法 1 所示。

算法 1 中所使用到的参数，如 2.2 节与图 3 中所说的，是 ICN.p4 数据包的特有字段，用于辅助 ICN 转发过程的执行。而 groupcast 操作代表了一种广播分发操作，将数据包分发到可能存在的多个对其发出了请求的地方去。

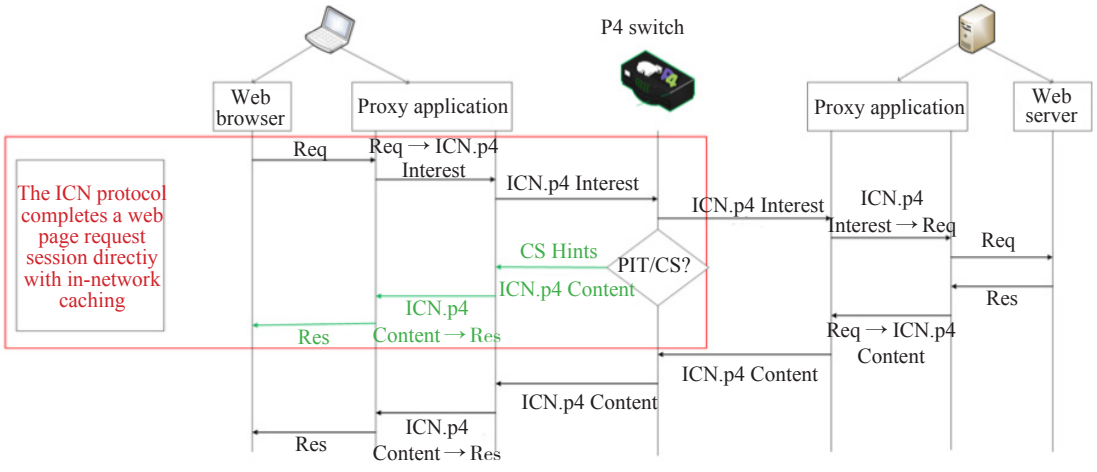


图 5 数据包转发过程

Fig.5 Packet forwarding process

```
Algorithm 1: Packet forwarding process
  Input: pkt
  Output: Forwarding action
1 Parse pkt header
2 if pkt. TCP.dport != reserved port then
3 |     IPV forwarding
4 else
5 |     Hash(pkt.name)
6 |     if pkt.op == Get then
7 |     |   if CS_Register[Hash(pkt.name)] == 1 then
8 |     |   |     Send back cached data
9 |     |   |     Break
10|     |   end
11|     |   if PIT_Register[Hash(pkt.name)]! == 0 then
12|     |   |     Update PIT_Register and drop pkt
13|     |   else
14|     |   |     IPV4 forwarding
15|     |   end
16|     end
17|     if pkt.op == Push then
18|     |   if pkt.isCache == 1 then
19|     |   |     Cache packet
20|     |   end
21|     |   if PIT_Register[Hash(pkt.name)]! == 0 then
22|     |   |     Update packet header, delete PIT entry and
  |     |   |     groupcast packet
23|     |   else
24|     |   |     IPV4 forwarding
25|     |   end
26|     end
27 end
```

就像 ICN 网络一样，在交换机上需要构建几个有状态表，例如用于包数据缓存的 CS 和用于记录请求的 PIT，这些操作都需要上面提到的算法。有状态表在算法中被抽象为每个交换机上的寄存器数组，例如 PIT 表，就使用寄存器数组来记录传入的请求。由于同时可能会有复数待回复的请求被记录在 PIT 中，这没有办法直接以 P4 语言实现，因此我们使用多个一维数组来实现多维数组记录。如图 6 所示，首先使用一个数组记录所有传入的请求。索引是每个请求名称的哈希值 [15]，该值是记录的请求信息的表项索引。如图 6 的简单示例上所说，索引为 13 425 的请求，其值 101 表示有两个请求被记录，并且两个请求的信息均记录在辅助阵列中。很容易知道，请求是来自 IP 地址为 10.0.0.1 和 10.0.0.2 的计算机。如上所述，由于并不能保证整个路径上的所有交换机设备都是 P4 交换机，所以需要在记录请求的同时缓存请求的发送者，以便我们可以直接使用多播进行回复。将目的 IP 地址放在数据包的 IP 标头位置，以确保通过多播分发的数据包可以到达预期的目的地。

除了 PIT，还需要另一个有状态表 CS。为了重现 ICN 中 CS 表的缓存功能，这里选择了一些小技巧来实现该模块的功能。由于 P4 语言只是一种数据平面上的网络编程语言，许多繁琐的文件存储操作并没有办法直接完成。我们使用 P4 语言在数据面上

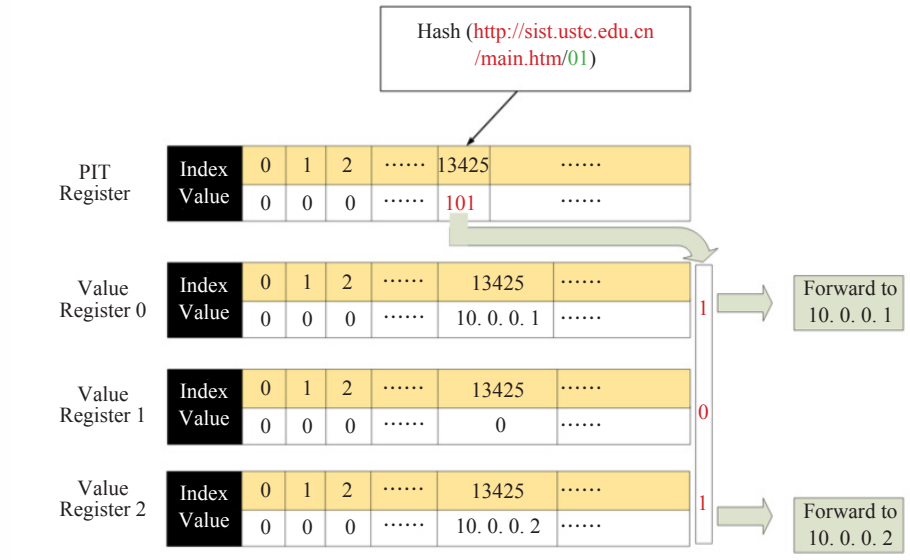


图 6 用寄存器数组实现的 PIT

Fig.6 PIT implemented with register array

申请一个寄存器数组空间，来记录本地缓存的状态，而实际的缓存功能是由特定的应用程序提供的。因此，从数据面的角度来看，交换机只记录状态，并没有实际数据存储，存储的数据被转发到提供缓存服务的应用程序。当接收到 OP 字段为 Get 的 ICN.p4 数据包，如算法 1 中所描述的，去查询与数组索引相对应的值，检查该值是否等于 ICN.p4 数据包名称的哈希散列值，以确定是否存在本地缓存。如果该值表示，存在一个本地数据缓存，那么就将 ICN.p4 数据包转发到应用层，同时交由提供缓存数据功能的应用程序去检索对应的数据并构造一个数据包发送回去。否则，就将直接转发数据包到下一跳。另一方面，如果到达的数据包的 OP 字段是 Push，那么只需要解析 isCache 字段值，来决定是否缓存数据包。图 7 展示了 P4 交换机的体系结构，包括交换机内各个模块实现的角色和功能的细节。

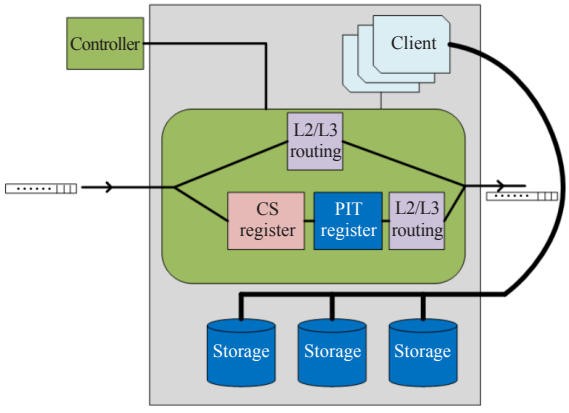


图 7 交换机架构

Fig.7 Switch architecture

如图 7 所示，寄存器与转发层都在交换机的数据面上，扮演着转发算法的核心——控制器角色。另外交换机内还有存储模块，用于交换机的本地存储功能，可以直接在需要时与用户进行交互。

3 评估

为了演示我们为完整的整合了 ICN 传输的优势的 HTTP 请求过程设计的场景，以多台虚拟机，构建了一个网络拓扑。每台虚拟机上都安装了 Ubuntu16.04 操作系统，也预装了 P4 软件交换机 BMV2。这样，每个虚拟机都可以看作一个 P4 交换机，交换机上的操作可以通过 P4 语言程序进行配置，流表项可以通过控制面来发送。

据了解，在类似领域中，有一些大多采用 Mininet 模拟器进行实验的研究。而在这里使用虚拟机来完成实验，就相当于使用每个交换设备作为一台计算机或作为一个 P4 交换机。因此，本文提出的网内缓存可以使用虚拟机的本地内存来实现。在实验中，可以使用每个虚拟机的 Ubuntu16.04 终端作为控制平面来配置流表，发送表项。类似地，代理应用程序的部署也是通过在 Ubuntu 上配置虚拟网络适配器来完成的 [17]，添加一个虚拟隧道，于是 HTTP 请求由已部署的代理申请人完全代表，这等效于将请求转发到应用程序层。此功能也可以由单独的代理网关来实现。

为了验证这一想法，我们构建了如图 8 中所示的拓扑来进行实验。在测试过程中，主要关注对象是服务器的响应负载。

如图 8 所示，PC_1 和 PC_2 是由 SERVER 提供的有意愿去浏览网页的消费者，并且在链路中间有两个转发设备，S_1 和 S_2。但是区别在于，S_2 是 P4 交换机，而 S_1 不是。因此，在 S_2 上存在请求聚合和网络内缓存发生的可能性。在这里，将 CS 寄存记录设置为 10 秒内有效，也就是说，只要同名的请求数据包在交换节点刷新缓存后 10 秒内到达 P4 交换机，交换机就可以将其直接转发到应用层，应

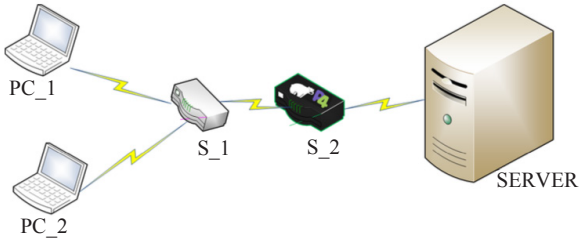


图 8 用于评估的拓扑网络

Fig.8 Topology for evaluation

用程序使用本地缓存来响应请求。同时，我们将 PIT 可以聚合两个请求的时间间隔设定为 10ms。那么，只有当两个同名的请求在 10ms 内到达同一交换机时，它们才会被聚合。

我们让 PC_1 和 PC_2 每 5 秒发送一次请求，重复这个操作 30 次。通过观察的数据，可以知道两个 PC 请求的数量之和与图 9 中的 SERVER 的响应之间的关系。从图中清晰地发现，当请求数相同时，对应的 ICN-HTTP 数方法比 HTTP 小得多，可以看出，ICN-HTTP 的响应负载仅为 HTTP 的一半，这主要归功于 S_2 节点的缓存和聚合。不过，由于聚合发生的条件十分严格，所以聚合的可能性很低，主要对网络优化起效的是网络内缓存功能。

图 9 服务器负载的小提琴图

Fig.9 Violin plot of server load

4 改进方案

在上面介绍了的方案的基础上，针对其兼容性，我们还可以进行一些改进。由于数据包的转换由代理应用程序执行，所以在调整链路环境时，除了需要将普通路由器替换为 P4 路由器，还需要在链路的起点与终点处配置代理应用程序。这样的情况，对现存网络的兼容性来说是不方便的。于是在新方案中，假如让数据包转换的过程也集成到 P4 交换机中，就能使得全部操作都由 P4 交换机完成。在对链路进行升级时，就只需要替换路由器，即可实现这一新方案的应用。

在文献 [18] 中提出了一种操作，它使用 P4 修改了 TCP/IP 包头，随后由 Linux 内核重新计算校验和进行封包。这给了我们启发，在 P4 交换机中，是可以实现前面提出的数据包转换操作的。根据这一思路，提出的改进方案具体如下。

当 P4 交换机收到传入数据包时，首先对数据包的 TCP 头进行检查，根据内容获取并计算 ICN 转发过程判断所需要的字段，并放入 TCP 负载当中，将普通数据包在 P4 路由器内部转换为 ICN.p4 数据包。接下来，ICN.p4 数据包经由 P4 程序设计的转发操作过程，判断之后的去向与内容操作，这与原方案中的 P4 路由器的任务是相同的。假如该数据包需要被传出 P4 交换机，那么在传出之前要将 ICN.p4 数据包转换回普通的数据包，使得进入网络传输的数据包都是一般格式的数据包，保证传输的兼容性。在去除 ICN.p4 特有的字段之后，交由内核重新计算校验和，执行重新封包，完成到普通 TCP 包的转换。这样，在链路中不全是 P4 路由器的情况下，也可以局部地实现优化 HTTP 传输，是一种可行的提高兼容性的改进方案。

5 结语

HTTP 协议在网页浏览中使用非常普遍，优化请求过程就显得尤为重要和有意义。本文提出了一种在 HTTP 请求过程的具体环境中采用 ICN 的思想进行通信的解决方案。由于网内缓存和请求聚合是 ICN 的固有特性，我们主张通过使用 P4 语言对 HTTP 请求过程实现 ICN 传输模式，也就是实现 HTTP 的网内缓存与请求聚合。主要的挑战在于不同数据包的转换和识别，以及在转发节点上进行的缓存和聚合。通过应用一个代理应用程序来代理 HTTP 请求过程并实现数据包转换，并且使用 P4 语言来配置转发节点，使它们能够缓存数据和聚合请求。

相比于其他网络内缓存的实现方案，如文献 [10]

和 [21]，本文所提出的方案由于使用了 P4，算法更为轻便，相对于其他研究，本文提出的方法重点关注了对响应速度的优化提升，经过验证，其改善的效果也比较显著。

本文中的实验是在几台安装了 Ubuntu 操作系统的虚拟机上进行的，为了评估在文中提出的方案的功能和性能而设计了一个网络拓扑结构。实验结果表明，在 HTTP 请求过程中采用 ICN 传输的思路，实现 HTTP 的网内缓存是合理的，该方案可以提高网络的传输效率，减少网络中的冗余流量。这些都有利于提高用户浏览网页的体验。

关于第四部分中所描述的改进方案，计划在未来对其进行更深入的实现方式设计，实施测试实验与性能评估，可能的话，不仅在虚拟机上，也计划在一些硬件设备上配置文中的方案进行测试评估。

利益冲突声明

所有作者声明不存在利益冲突关系。

参考文献

[1] I. D. R. T. Fielding and H. F. Nielsen,Hypertext transfer protocol – http/1.0 [M] .Computer Science & Communications Dictionary, vol. 7, no. 4, pp. 595–599, 1999.

[2] X. Qiao, G. Nan, P. Yue, G. Lei, J. Chen, Y. Sun, and J. Chen,Ndnbrowser: An extended web browser for named data networking [J] .Journal of Network & Computer Applications, vol. 50, pp. 134–147, 2015.

[3] M. N. O. Sadiku, A. E. Shadare, and S. M. Musa, Named data networking [C] .AcmSigcomm Computer Communication Review, vol. 44, no. 3, pp. 66–73, 2014.

[4] M. Rabinovich and H. Wang, Dhttp: an efficient and cache-friendly transfer protocol for web traffic [C] .Proc IEEE Infocomthe Conference on Computer Communications, 2001.

[5] P.Bosshart,D.Dan,M.Izzard,N.Mckeown,J.Rexford,C. Schlesinger, T. Dan, A. Vahdat, G. Varghese, and D. Walker, Programming protocol- independent packet processors [C]. AcmSigcomm Computer Communication Review, vol. 44, no. 3, pp. 87–95, 2014.

[6] A. Sivaraman, C. Kim, R. Krishnamoorthy, A. Dixit, and M. Budiu, Dc.p4: programming the forwarding plane of a data-center switch [Z] .2015.

[7] H. Yuan, S. Tian, and P. Crowley,Scalable ndn forwarding: Concepts, issues and principles [C].International Conference on Computer Communications & Networks, 2012.

[8] I. Psaras, K. C. Wei, and G. Pavlou, Probabilistic in-network caching for information-centric [Z] .2012.

[9] H. Wang, R. Soule, H. T. Dang, K. S. Lee, and H. Weatherspoon, P4fpga: A rapid prototyping framework for p4 [C] .Symposium on Sdn Research, 2017.

[10] J.Xin,X.Li,H.Zhang,R.Soule,J.Lee,N.Foster,C.KimandI. Stoica, Netcache: Balancing key-value stores with fast in-network caching [C] .Proceedings of the 26th Symposium on Operating Systems Principles, 2017.

[11] S. Signorello, R. State, J. Francois, and O. Festor,Ndn.p4: Programming information-centric data-planes [C] .Netsoft Conference & Workshops, 2016.

[12] Z. Ma, J. Bi, Z. Cheng, Z. Yu, A. B. Dogar, Z. Ma, J. Bi, Z. Cheng, Z. Yu, and A. B. Dogar, Cachep4: A behavior-level caching mechanism for p4 [C] .Sigcomm Posters & Demos, 2017.

[13] B. A. Ramanan, L. M. Drabeck, M. Haner, N. Nithi, T. E. Klein, and C. Sawkar, Cacheability analysis of http traffic in an operational lte network [Z] .2013.

[14] X.Marchal,M.E.Aoun,B.Mathieu,W.Mallouli,T. Cholez,G.Doyen, P. Truong, A. Ploix, and E. M. D. Oca, A virtualized and monitored ndn infrastructure featuring a ndn/http gateway [C] .Acm Conference on Information-centric Networking, 2016.

[15] W.So,A.Narayanan,D.OranandY.Wang, To ward fastndn soft ware forwarding lookup engine based on hash tables [C] .Eighth Acm/ieee Symposium on Architectures for Networking & Communications Systems, 2012.

[16] H.YuanandP.Crowley, Experimentalevaluationofcontentd

istribution with ndn and http [C] .IEEE Infocom, 2013.

[17] R. S. V. Eiras, R. S. Couto, and M. G. Rubinstein, Performance evaluation of a virtualized http proxy in kvm and docker [C] .Network of the Future, 2017.

[18] Benoît Pit-Claudel, Yoann Desmouceaux, Pierre Pfister, Mark Townsley and Thomas Clausen, Stateless Load-Aware Load Balancing in P4 [C] .IEEE 26th International Conference on Network Protocols, 2018.

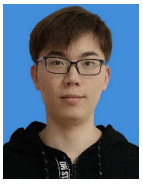
[19] Sergio Charpinel, Celso Alberto Saibel Santos, Alex Borges Vieira, Rodolfo Villaca and Magnos Martinello, SDCCN: A Novel Software Defined Content-Centric Networking Approach [C] .IEEE 30th International Conference on Advanced Information Networking and Applications, 2016.

[20] Markus Vahlenkamp, Fabian Schneider, Dirk Kutscher and Jan Seedorf, Enabling Information Centric Networking in IP Networks Using SDN [C] .IEEE SDN for Future Networks and Services, 2013.

[21] Erick B. Nascimento, Douglas D. J. de Macedo, Edward David Moreno, Luis Carlos Erpen de Bona and Miriam A. M. Capretz, Evaluation of Cache for Bandwidth Optimization in ICN Through Software-Defined Networks [C] .IEEE Symposium on Computers and Communications, 2018.

收稿日期：2020 年 4 月 10 日

詹昱辰，中国科学技术大学信息科学技术学院，主要研究方向为未来网络架构领域。
本文主要负责文献调研与文章撰写。
Zhan Yuchen is now an undergraduate student at the University of Science and Technology of China(USTC) and will continue his graduate studies in the Department of Automation, USTC. His research interests include future Internet architecture related areas.
In this paper he is mainly responsible for literature research and article writing.
E-mail: zyc233@mail.ustc.edu.cn



冯巍巍，中国科学技术大学自动化系硕士研究生，主要研究方向为未来网络架构领域。
本文主要承担实验设计与实施工作。
Feng Weiwei is currently working toward MS degree in the Department of Automation, USTC, China. He received his B.S. degree at BUPT (Beijing University of Posts and Telecommunications) in 2018. His research interests include future Internet architecture related areas.
In this paper he is mainly responsible for the design and implementation of the experiments.
E-mail: fengww@mail.ustc.edu.cn

谭小彬，中国科学技术大学信息科学技术学院自动化系，博士，副教授，研究方向包括未来网络架构、自适应传输流和信息安全。
本文主要工作为作为指导老师指导文章完成。
Tan Xiaobin is an associate professor in the Department of Automation, School of Information Science and Technology, USTC, China. He received his B.S. and Ph.D. degree at the University of Science and Technology of China (USTC) in 1996 and 2003 respectively. His research interests include future Internet architecture, adaptive streaming, and information security.
In this paper he is mainly responsible for guidance of paper competition.
E-mail: xbtan@ustc.edu.cn

引文格式：詹昱辰,冯巍巍,谭小彬. 基于P4的HTTP网内缓存方案及其实现[J].数据与计算发展前沿, 2020,2(3): 75-86.DOI:10.11871/jfdc.issn.2096-742X.2020.03.007.PID:21.86101.2/jfdc.2096-742X.2020.03.007.
Zhan Yuchen, Feng Weiwei, Tan Xiaobin. A Solution for HTTP In-Network Caching Based on P4 [J].Frontiers of Data & Computing, 2020,2(3): 75-86. DOI:10.11871/jfdc.issn.2096-742X.2020.03.007.PID:21.86101.2/jfdc.2096-742X.2020.03.007.