第 45 卷 第 7 期 计 算 机 工 程 2019 年 7 月
Vol. 45 No. 7 Computer Engineering July 2019

·体系结构与软件技术· 文章编号：1000-3428(2019)07-0046-08 文献标志码：A 中图分类号：TP393

支持变长间隔网内缓存的软件定义视频流化系统

陈金雯，姚 振，杨 坚，奚宏生

（中国科学技术大学 自动化系，合肥 230027）

摘 要：为解决大量视频片段冗余传输造成的网络带宽严重浪费问题，设计一种改进的软件定义视频流化系统。基于网络功能虚拟化技术，在网络节点上实现视频缓存和流化功能。通过抽象出可编程的缓存策略模块，部署针对实时传输视频流的变长缓存窗口（VLCW）算法，降低服务器负载。根据不同的用户接入模式自适应调节缓存视频片段长度，提高缓存资源利用率。实验结果表明，在 VLCW 算法的优化下，该系统服务器负载下降 50%，缓存资源利用率提高 3 倍～5 倍。

关键词：软件定义网络；网内缓存；多媒体流化；OpenFlow 协议；视频点播系统

中文引用格式：陈金雯，姚振，杨坚，等. 支持变长间隔网内缓存的软件定义视频流化系统[J]. 计算机工程，2019，45(7)：46-53.

英文引用格式：CHEN Jinwen，YAO Zhen，YANG Jian，et al. Variable-length interval in-network caching enabled software-defined video streaming system[J]. Computer Engineering，2019，45(7)：46-53.

Variable-Length Interval In-Network Caching Enabled Software-Defined Video Streaming System

CHEN Jinwen，YAO Zhen，YANG Jian，XI Hongsheng

（Department of Automation，University of Science and Technology of China，Hefei 230027，China）

【Abstract】In order to solve the problem of network bandwidth waste seriously caused by redundant transmission of a large number of video clips，an improved software-defined video streaming system is designed. Based on Network Function Virtualization（NFV）technology，video caching and streaming functions can be implemented on network nodes. By the abstracted programmable cache policy module，a Variable-Length Cache Window（VLCW）algorithm is deployed for real-time video streaming transmission to reduce the server load. The length of the cached video clips is adaptively adjusted according to different users' access patterns so as to improve the cache resource utilization. Experimental results show that under the optimization of the VLCW algorithm，the server load of the system drops by 50%，and the cache resource utilization is increased by 3 to 5 times.

【Key words】Software-Defined Network（SDN）；in-network caching；multimedia streaming；OpenFlow protocol；Video on Demand（VoD）system

DOI：10.19678/j.issn.1000-3428.0051225

0 概述

视频传输在当今网络流量组成中占据相当大的比例，根据 Cisco[1] 的预测，IP 视频流量占所有消费者网络流量的百分比将从 2016 年的 37% 提升到 2021 年的 82%。届时，每秒钟都将有近百万分钟的视频内容在网络上传输，如此大规模的需求会使网络带宽资源迅速饱和，从而导致延迟、抖动和丢包等一系列问题。在此情况下，视频服务提供商很难保证终端用户的服务质量（Quality of Service，QoS）和体验质量（Quality of Experience，QoE）。

视频热度的倾斜度服从帕累托准则[2-3]，即 20% 的视频对象覆盖了 80% 的访问量，而多次访问同一视频会产生相当多的流量冗余。IP 组播可以在一次发送中将数据推送给一组感兴趣的接收者，但组播只支持同步视频播放而不支持视频点播（Video on Demand，VoD）这类个性化差异服务，总体而言仍无法减少流量冗余。另一种思路是通过应用层优化来提高视频内容的传输效率，如在传统网络协议栈上部署内容分发网络（Content Distribution Network，

基金项目：国家自然科学基金面上项目（61573329）。

作者简介：陈金雯（1978—），女，硕士研究生，主研方向为网络系统建模与优化；姚 振（通信作者），博士研究生；杨 坚、奚宏生，教授。

收稿日期：2018-04-16 **修回日期：**2018-09-18 **E-mail：**yaozhen1@mail.ustc.edu.cn

CDN)[4]。但 CDN 只是一个覆盖在稀疏代理服务器网络上的逻辑网络,不能降低终端用户和选定代理服务器之间的冗余。而且,如果 CDN 服务器位于自治系统(Autonomous System,AS)之外,则会引发大量的跨域冗余流量,且部署大规模的 CDN 服务器成本过高[5]。

随着网络功能虚拟化(Network Function Virtualization,NFV)技术的普及,网络中的路由节点可以支持内容缓存,即网内缓存[6]。由于缓存磁盘被集成在网络节点上,而非 CDN 的存储服务器上,因此可实现网络层的内容缓存和分发,使其尽可能贴近用户,从而减少边缘网络的冗余,同时能避免将过多视频请求重定向到 ISP 网络之外的代理服务器上。

尽管 NFV 可减少设备开销,缩短设备开发周期,提高可扩展性、灵活性和敏捷性,但要从现有静态、无弹性的传统网络迁移到基于 NFV 的新型网络架构仍面临较多挑战。软件定义网络(Software-Defined Network,SDN)[7]提供了一个可扩展的智能框架,能在虚拟或物理基础设施上动态部署基于 NFV 的网络功能[8]。OpenFlow[9]是首个实现 SDN 原理的平台,提供了相应的接口来使能控制平面和数据平面之间的交互。

在此基础上,本文设计一个支持网内缓存且可完成高效视频内容分发的软件定义 VoD 系统,能实现从距离用户最近的网络节点完成内容交付,以降低传输时延及流量冗余,并提高 QoS 和 QoE。

1　相关工作

1.1　基于 NFV 的 CDN 和网内缓存

通过 NFV 可以改进传统 CDN 服务。文献[10]在 CDN 场景下,设计基于 NFV 和云计算的虚拟节点和应用,从而通过较小的代价实现有效的内容分发。文献[11]利用 OpenStack 设计多媒体传输框架,基于 NFV 构建 CDN 来提高服务质量。上述研究虽然取得了一定的优化效果,但是终端用户和代理服务器之间的冗余流量仍未被消除。

网内缓存技术可减少重复流量。文献[12]在网内缓存场景下引入资源管理的概念,在减少冗余的同时可使资源得到更有效的利用。文献[13]利用 CCN/NDN 网络的网内缓存能力,根据特定的内容缓存策略来预缓存 MPEG-DASH 编码的视频片段。上述研究较传统基于代理服务器的缓存架构有明显的性能改善,但仍较难实现有效的全局管控,因此需引入 SDN 技术做进一步性能优化。

1.2　支持 SDN 的视频流化和缓存管理

文献[14]利用 SDN 实现分层组播视频会议系统,提高异构终端接收可伸缩视频编码(Scalable Video Coding,SVC)视频流质量,降低时延。文献[15]提出基于 SDN 的 DASH 路由规划和码率调节联合算法来提升视频分发效率。文献[16]在 SDN 背景下考虑了视频多媒体服务的接入控制和路由优化算法。然而,以上方案未将网内缓存和 SDN 相结合。文献[17]提出 OpenCache 框架以提升缓存分发效率,但 OpenCache 节点部署在网络边缘的独立服务器上,而本文提出的网内缓存是集成在 OpenFlow 交换机中,视频流传输不需要遵循端对端原则。

1.3　缓存置换策略

传统缓存系统有许多经典的内容置换算法,如最近最少使用(Least Recently Used,LRU)算法和最低频率使用(Least Frequently Used,LFU)算法均得到广泛应用[18]。文献[19]提出一种在线学习缓存置换方法,学习视频随时间变化的流行度并预测其变化趋势。以上算法尽管都在不同程度上达到性能改善的目的,但都需要缓存整个视频文件,且视频流行度预测准确度有待提高。

文献[20]考虑利用最近的用户行为信息预测将来的视频片段请求,提出不同匹配度水平下的 2 种视频片段缓存策略,以提高缓存命中率和降低带宽消耗。文献[21]扩展了文献[22]中为处理多媒体服务系统中混合负载而提出的间隔缓存策略,本文在此基础上提出针对支持网内缓存的 SDN 视频流化系统,设计变长间隔缓存策略,以降低冗余流量和提高资源利用率。

2　SDN 多媒体流化构架

2.1　系统构架

如图 1 所示,本文多媒体流化系统针对内容提供商在 ISP 的小规模边缘网络上部署服务的场景进行设计。该系统由 OpenFlow 控制器、多媒体流化服务器(Media Streaming Server, MSS)、管理服务器(Management Server,MS)、OpenFlow 交换机和终端组成。图 1 中带箭头虚线表示从流媒体服务器发出的源视频流,而带箭头实线表示从节点缓存中流化出的缓存视频流。由于 SDN 具有中心控制管理、信息收集及状态监测能力,且可以支持网络节点和上层应用之间的协作,因此本文系统基于 SDN 架构进行设计。支持缓存的软件定义多媒体流化系统架构如图 2 所示。

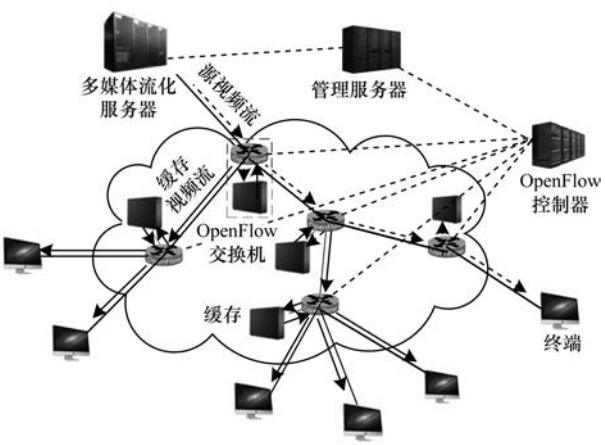

图 1　支持网内缓存的软件定义多媒体流化系统

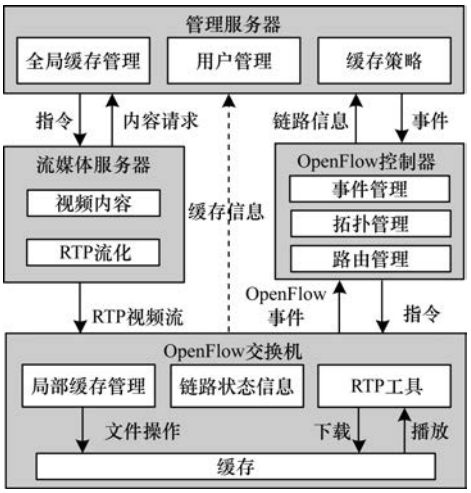

图 2 支持网内缓存的软件定义多媒体流化系统架构

在 MSS 上有全部视频内容,还部署了视频服务应用来接收用户请求并为每个会话产生实时传输协议(Real-time Transport Protocol,RTP)视频流。其基本功能是提供 VoD 服务(视频内容管理和 RTP 流化)。相应地,终端上会有视频播放器触发请求,接收视频流并解码播放。MS 像整个系统的大脑,从交换机获得拓扑、缓存分布和连接状态信息,并运行特定的缓存策略,指导数据缓存和视频流化。OpenFlow 控制器负责根据 MS 的决策控制交换机动作。路由管理模块和事件管理模块使得 MS 能控制交换机来推送、缓存和流化视频。Cache 是集成在交换机内的网内缓存。交换机可以进行文件管理和内容传输,具备本地缓存管理、链路状态监测、RTP 数据处理的能力。LCM 支持基本的文件操作来缓存视频流,并提供关于链路状态及缓存分布的信息作为事件周期性地上传到 OpenFlow 控制器,再交给 MS 分析处理。部署在交换机中的 RTP[23] 工具支持 RTP 数据缓存、读取输出和发送 RTP 流到终端。为使得 OpenFlow 控制器能实现网内缓存管理和内容传输的功能,本文分别定义了新的与存储管理和视频流化相关的事件来扩展 OpenFlow 协议。位于传输路径上的交换机可以复制和缓存相应的 RTP 视频,使得被请求过的高热度内容能够遍布全网。当后续的用户发起相同的内容请求,可以直接从网络节点上获取内容,间接降低了 MSS 的负载。为高效地利用缓存,需要一个有效的针对 SDN 网络场景视频流化服务的缓存管理策略。本文将采用间隔缓存技术优化视频流的网内缓存,而非缓存整个视频对象。

2.2 变长间隔缓存技术

变长间隔缓存策略中的数学符号如表 1 所示。交换机中的缓存由 MS 中的全局缓存管理模块控制,由其发出缓存特定视频流的指令后,交换机会创建一个本地文件缓存 RTP 数据包。变长间隔缓存窗口作为缓存的基本单元,其机制如图 3 所示,其中,$R_{i,j}$ 表示第 i 个视频的第 j 个请求,$L_{i,j}$ 表示第 i 个视频的第 j 个窗口的长度。

表 1 变长间隔缓存策略中的数学符号说明

符号	定义
T	缓存窗口初始时长
T'	更新后的缓存窗口时长
$T_{\max}$	缓存窗口长度上界
n	到达缓存窗口的请求数量
C_i	第 i 个视频对象
N	流媒体服务器中可获得的视频数量
$W_{i,j}$	第 i 个视频对象的第 j 个缓存窗口
$K_{i,j}$	窗口 $W_{i,j}$ 中的请求数量
L_i	第 i 个视频对象的缓存窗口总数
$R^k_{i,j}$	窗口 $W_{i,j}$ 中的第 k 个请求
$T^k_{i,j}$	$R^k_{i,j}$ 的到达时间
$R^n_{i,j}$	缓存窗口 $W_{i,j}$ 中最后一个请求的到达时间
$I^k_{i,j}$	缓存窗口 $W_{i,j}$ 中第 k 个连续到达请求之间的到达时间间隔
$I^n_{i,j}$	缓存窗口 $W_{i,j}$ 中最后一个请求和视频结束时间之间的时间间隔
$\overline{I}_{i,j}$	缓存窗口 $W_{i,j}$ 中的平均到达时间间隔

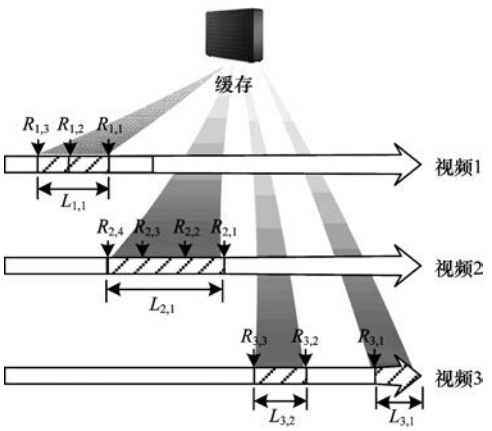

图 3 变长间隔缓存示意图

整个视频文件被当作一个固定长度的时间轴,被交换机缓存的部分则是在视频时间轴上滑动的一定长度的窗口。由于窗口尺寸调整的粒度为 GoP,比窗口长度的典型值小多个数量级,因此可忽略视频编码带来的影响,而合理地假定窗口长度仅取决于视频请求的到达模式,即到达时间的概率分布。同一窗口中的视频请求会重复利用上游交换机中的同一个视频流,即在时间域上聚集冗余的视频流量。此外,值得注意的是间隔缓存技术仅缓存窗口中的视频流量。

MS 中的全局缓存管理模块控制交换机中的缓存。基于图 4 所示的链表和 STL 图动态地维护存储的组织结构。

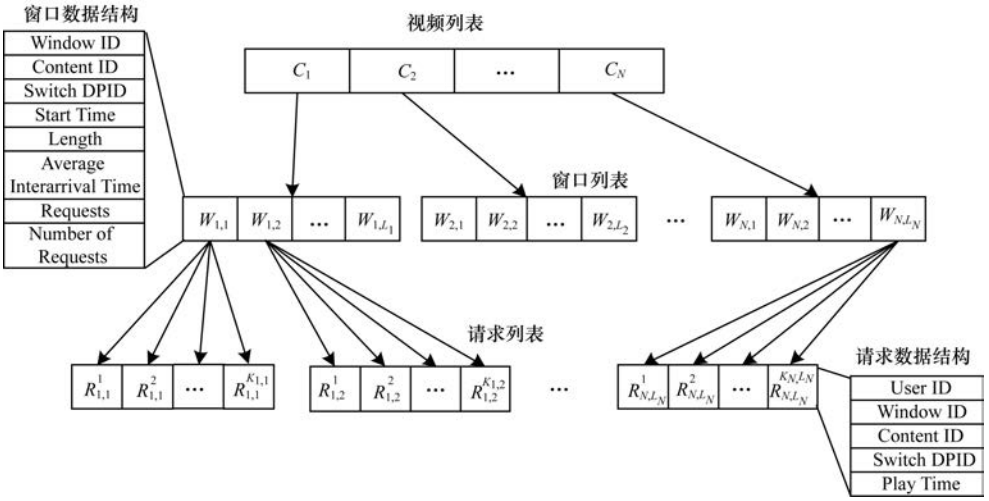

图 4　存储组织的数据结构

图 4 包含视频对象和窗口的匹配关系。视频对象列表包括若干 C_i,一个对象 C_i 与一个包含 $W_{i,j}$ 的窗口列表相关,而 $W_{i,j}$ 与不同的交换机相关。一个 $W_{i,j}$ 包含一个请求列表 $R^k_{i,j}, k=1,2,\cdots,K_{i,j}$,$K_{i,j}$ 为窗口的请求总数。每个窗口有一个记录,包含 Window ID、Content ID、交换机标识(DPID)等。此外,开始时间、长度、请求数量和平均到达间隔时间都是窗口对象的属性。

2.3　基本操作流程

用户触发一个视频请求,MSS 解析后提取用户的 IP 地址、视频 ID 和到达时间发送给 MS。MS 的用户管理模块基于与 IP 地址相关的用户 ID 鉴定用户,并授权对视频服务的接入。对授权用户,MS 使用 OpenFlow 控制器的链路状态与全局缓存管理模块的缓存分布信息共同确定最优的内容传输路径。一旦确定,MS 将把所有的行动映射为交换机中可执行的指令,更新全局的缓存信息。交换机接到指令后会开启新进程传输 RTP 包,除非缓存未命中,此时继续由 MSS 提供内容。

在被请求视频的传输路径的节点上对其进行缓存,并根据本地热度相应地自适应调整缓存长度。如果有新的用户请求相同的内容,则优先在最近的网络节点上查找是否有相应的缓存。如果有,则直接从该节点取回视频内容,实现同一份视频缓存的异步复用,提高缓存资源利用率;如果没有,则继续从源服务器取回内容。

3　基于变长间隔窗口的网内缓存策略

3.1　状态机

假定一个如图 5 所示初始长度为 T 的缓存窗口 $W_{i,j}$ 中有 n 个请求到达,$R^n_{i,j}$ 为最后一个到达的请求。定义 $I^k_{i,j}$ 为第 k 个(连续 2 个到达的请求之间)的间隔时间。

$$I^k_{i,j}=\begin{cases}\infty, k=n=1\\ T^{k+1}_{i,j}-T^k_{i,j}, 1<k<n\\ T+T^1_{i,j}-T^n_{i,j}, k=n>1\end{cases} \tag{1}$$

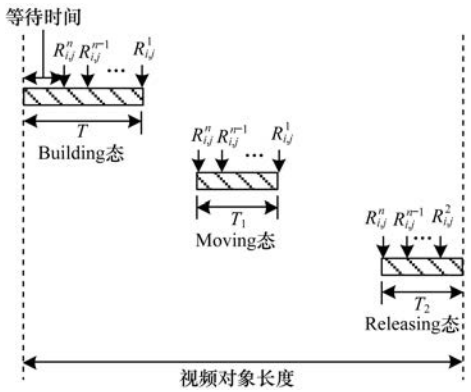

图 5　缓存窗口的状态示意图

如果 $n=1$,则 $I^1_{i,j}$ 被指定为 ∞,即没有间隔存在。最后一个请求到达时间和时间区间结束点之间的间距被称为等待时间,即为等待下一个请求的时间。另外,定义 $\bar{I}_{i,j}$ 为缓存窗口 $W_{i,j}$ 的平均到达间隔:

$$\bar{I}_{i,j}=\begin{cases}\infty, n=1\\ \sum_{k=1}^{n-1} I^k_{i,j}/(n-1), n>1\end{cases} \tag{2}$$

如果在时间区间 T 中只有一个请求到达,则不存在平均间隔时间,设置为 ∞。$\bar{I}_{i,j}$ 越小意味着对视频的访问越频繁,所以 $I^k_{i,j}$ 序列或 $\bar{I}_{i,j}$ 序列刻画了视频的流行度和当前的用户请求模式。

本文用状态机来描述一个缓存窗口的生命周期。如图 5 所示,定义缓存窗口的 3 个状态:1) Building 态。一旦有一个请求到达,如果没有包含需要的缓存片段的缓存窗口存在,则会初始化一个长度为 T 的初始缓存窗口,其大小在“冻住”以前可调。“冻住”是指该窗口不再接收新的视频请求。在缓存窗口“冻住”以前的状态称为 Building 态。如果被请求的对象没有缓存窗口,则分配第 1 个下标为 1 的缓存窗口,后续创建的缓存窗口下标依次增加。2) Moving 态。在缓存窗口冻住后,进入 Moving 态。缓存窗口随着视频对象的流化进程在时间轴上移动。需注意 Moving 态下缓存窗口的大小

只会在提前终止或类 VCR 操作[24]（暂停、前进、后退等）发生时发生改变。3）Releasing 态。当对应缓存窗口中第 1 个请求的视频会话进行到视频流的终点时，缓存窗口进入 Releasing 态。

3.2 变长间隔网内缓存策略

一个缓存窗口的所有请求会被分派到同一个交换机中以重复利用缓存中的同一个数据块，从而降低冗余的网络流量。当第 i 个对象的用户请求到达时，如果包含所求视频片段的缓存窗口在 Building 态，且相应的交换机有输出视频流的可用带宽，则该请求就被安排到最优的交换机中，从而得到最短的传输路径。传输路径上的每一个没有该缓存内容的交换机都要创建一个初始长度为 T 的新的缓存窗口；否则，该请求就被安排到 MSS 中建立新的 RTP 会话。同样地，此时传输路径上的每个交换机也都会创建一个长度为 T 的新的缓存窗口。

下文讨论缓存窗口大小的调整。在创建一个缓存窗口时，会分配一个初始的长度 T 并开始一个三阶段的生命周期。在这些阶段中，缓存窗口长度会根据以下规则进行调整，如图 6 所示。

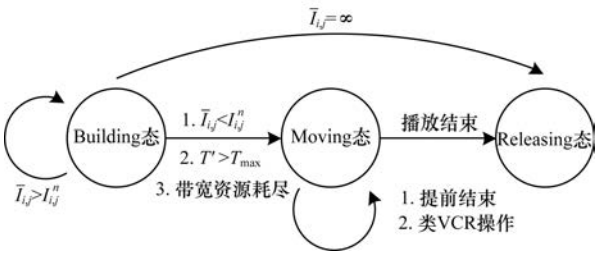

图 6 缓存窗口的状态转换及相应条件

为防止缓存窗口过大，缓存窗口的尺寸由一个上界 T_{max} 限定。会话结束也可能导致缓存大小的调整，包括 2 种不同类型的进程结束，完成后结束和提前结束。完成后结束时会话在视频流末端结束，可以根据 Releasing 态结束进行处理。对于提前结束的情形，后来的请求可能比前面的请求结束得还早，此时缓存窗口大小的调整就需要特殊的处理机制。不失一般性，本文仅讨论 Moving 态的缓存窗口调整规则，其他 2 个状态可以进行类似推导。

考虑处在 Moving 态的缓存窗口 $W_{i,j}$ 有连续的请求 $R^1_{i,j},R^2_{i,j},\cdots,R^n_{i,j}$，如果请求 $R^k_{i,j}$（$1\leqslant k\leqslant n$）的会话先于其他会话结束，则缓存窗口的大小调整遵循以下原则：1）如果 $R^k_{i,j}$ 处于缓存窗口的中间，即 $1<k<n$，在同一个缓存窗口中有前后请求，则缓存窗口保持目前状态并移除 $R^k_{i,j}$。2）如果 $R^k_{i,j}$ 处于缓存窗口的头部，即 $k=1$，则移除 $R^1_{i,j}$；如果该请求是缓存窗口中的唯一请求，则删除该请求后该缓存窗口也被销毁；否则该窗口的头部会收缩到 $R^2_{i,j}$，且 $R^1_{i,j}$ 和 $R^2_{i,j}$ 之间的区间会被移除；3）如果 $R^k_{i,j}$ 处于缓存窗口的尾部，即 $k=n$，从请求队列中删除 $R^n_{i,j}$，且缓存窗口的尾部收缩，$R^{n-1}_{i,j}$ 成为最后一个请求。

算法 1 基于最短路径的视频传输算法

输入 被请求的视频 C_i，视频 C_i 的第 j 个缓存窗口 $W_{i,j}$，缓存窗口 $W_{i,j}$ 所处 OpenFlow 交换机的 ID 号 $N_{i,j}$，请求用户和交换机 $N_{i,j}$ 之间的跳数 $h_{i,j}$，交换机 $N_{i,j}$ 维护的并行会话数目 $m_{i,j}$，从源媒体服务器到请求用户的跳数 h_0，单个交换机可以维护的最大会话数 M，获得的被请求视频的缓存窗口集合 $W(i)$

输出 最优传输视频的缓存节点 N^*

初始化 $W(i)=\phi$，$h_{min}=h_0$，$m_{min}=M$

1. for all $W_{i,j}\in C_i$ do
2. if $W_{i,j}$ is in Building State and $m_{i,j}<M$ then
3. $W(i)=W(i)\cup\{W_{i,j}\}$
4. if $h_{i,j}<h_{min}$ then
5. $h_{min}=h_{i,j}$
6. end if
7. end if
8. end for
9. if $W(i)==\phi$ or $h_{min}==h_0$ then
10. N^* = Media Streaming Server
11. else
12. for all $W_{i,j}\in W(i)$ do
13. if $h_{i,j}==h_{min}$ and $m_{i,j}<m_{min}$ then
14. $m_{min}==m_{i,j}$
15. $N^*=N_{i,j}$
16. end if
17. end for
18. $m_{i,j}=m_{i,j}+1$
19. end if
20. return N^*

经过以上步骤，缓存窗口参数如 $\bar{I}_{i,j}$ 应立即更新。对类 VCR 操作的情形则更复杂。由于请求的相对接入位置会变化，也导致相应缓存窗口的状态变化，因此需要扩展上述缓存策略来支持类 VCR 操作，例如暂停和继续播放。特别地，暂停操作和提前终止的情形类似，而继续播放和跳转会被当做接收到新的请求。于是，变化的请求会被分配给一个合适的缓存窗口（可能不同于原来的缓存窗口），最坏的情况是被重新安排给 MSS。结束类 VCR 操作后，需要注意的是所有相关的窗口状态也要相应更新。

3.3 最短路径传输策略

除了要讨论交换机中的缓存策略，本文还需选择合适的交换机传输视频内容。将算法 1 应用在本文提出的软件定义视频流媒体系统中。具体来说，该轻量级算法联合考虑了缓存窗口状态、交换机吞吐量和网络拓扑信息来选择一个最优的交换机节点作为视频源实现就近服务。

算法 1 的第 1 行～第 8 行是搜索可用的交换机，第 12 行～第 17 行是搜索离用户最近且能达到负载均衡的维持最少会话连接数的交换机。一旦根据算法 1 得出最优交换机或源服务器，传输路径上的交换机就需要创建新的窗口来缓存视频流。

4　原型系统实现

为验证本文系统设计的有效性,建立一个 SDN 实验平台来搭建如图 1 所示的变长网内缓存的多媒体流系统。选择 OpenFlow 交换机作为 NFV 硬件。32 GB 内存的 Kingston's DataTravalerSE9G2 作为外部 USB 闪存来提供视频缓存能力,每个交换机安装有 RTP 工具。选择 POX[25] 作为轻量级的 OpenFlow 控制器,在一个有 3 张网卡的 Linux 服务器上运行,第 1 个 NIC 连到一个标准的 24 端口 TL-SG1024DT 交换机,该交换机再连接到所有 OpenFlow 交换机的 WAN 端口,从而构建一个 out-of-band 控制平面网络。第 2 个 NIC 连接到 MS 来共享网络和缓存信息并接收控制信息。OpenFlow 控制器的模块也在 POX 中实现。原型系统的网络拓扑按图 7 所示的网络拓扑进行搭建。

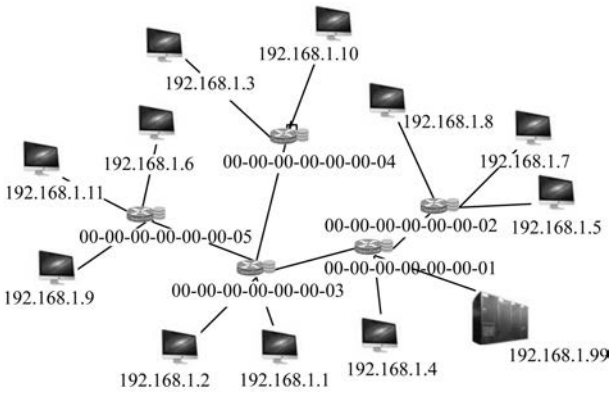

图 7　原型系统的网络拓扑

MS 也在一个有 2 块 NIC 的 Linux 服务器上实现。一块连到 OpenFlow 控制器来构建控制平面,另一块连到 MSS。MSS 是一台安装了 VLC 2.0.8 媒体播放器的 Linux 服务器,作为 RTP 流化模块来封装数据到 RTP/UDP 中。本文还构建了不同端口的多 UDP socket 来流化视频数据给用户,使得交换机能识别特定请求的数据。终端用 Linux 客户端实现,在其上同样部署有一个 VLC 媒体播放器来解码收到的 RTP 流并播放视频。

基于以上原型系统研究本文提出的变长间隔缓存窗口(Variable-Length Cache Window,VLCW)算法性能,并与其他广泛应用的调度算法相比较,如 LRU,其被认为较适合 CDN 和 ICN 网络。考虑 2 种不同形式的 LRU,即 NNC + LRU[26] 及其简单变形 Rdm + LRU[27] 算法。对 NNC + LRU 而言,内容传输路径上的所有节点都会按照 LRU 算法尽可能地缓存数据包,而 Rdm + LRU 则只随机选择一个有足够缓存空间或可置换目标的节点。这 2 种缓存置换算法在 SDN 场景下的性能要优于其在传统 CDN 缓存架构中的情况[10],因此后续实验不再与 CDN 网络作对比。

在对比实验中,本文考虑有限的网络带宽和硬件性能的情况。请求到达率 λ 设置为 2 请求/min,且视频片段的长度设置为10 min。使用服务器的实时吞吐量、当前整个网络中同时发生的 RTP 流的实时数量、从各个缓存节点传输过来的实时 RTP 会话数量来研究已实现的原型系统性能。

图 8 显示同样条件(请求序列、存储空间等)下 3 种算法的比较结果。图 8(a)表明 VLCW 算法在大部分时间内具有最低的吞吐量,即有最少的请求需要由服务器响应。相较于 2 种对比算法,VLCW 的平均服务器负载下降了 50%。本文将属于同一个 RTP 会话的不同物理链路上的数据流作为不同的流来量化整个网络的带宽消耗。图 8(b)表明 VLCW 算法有最少数量的 RTP 流,即其消耗了最少的带宽。图 8(c)说明了 VLCW 算法中被请求的视频有更大的概率从缓存节点获得,即其有更高的缓存命中率。

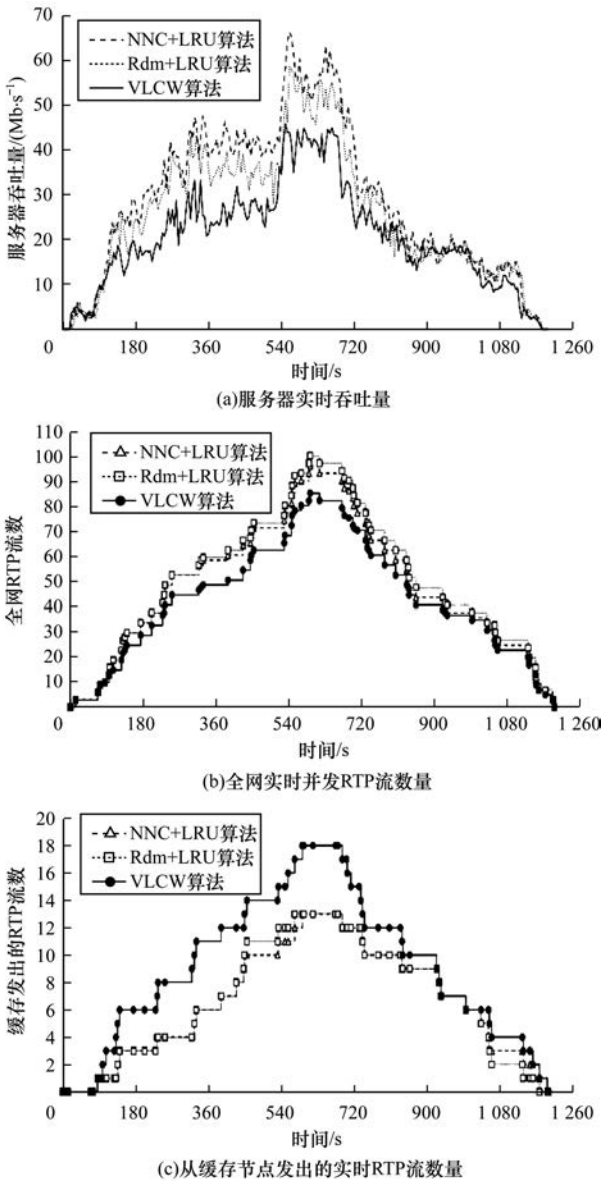

图 8　原型系统性能比较结果

5 仿真及性能评估

5.1 评估指标与仿真环境

为全面评估本文算法的性能,采用的测量指标如表 2 所示。本文使用 Mininet 2.2.1[28] 来创建一个符合实际场景的虚拟 SDN 网络。选择一棵深度为 4(服务器除外)的完全 4-叉树作为网络拓扑进行仿真,如图 9 所示,包括 1 个服务器、21 个交换机、64 个客户端。假定到达服务器的请求服从 Poisson 分布,平均到达率为 λ,且视频对象的热度服从类 Zipf 分布[3]。

表 2 仿真测量指标

指标	定义
流化效率 r_{server}	从服务器中送出单个流的聚合效率
缓存效率 r_{cache}	缓存系统中单位长度视频片段的复用效率
效率因子 η	缓存负载(实时服务的视频请求数)与服务器负载的比值

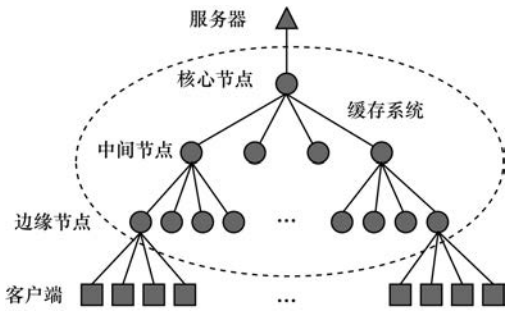

图 9 仿真网络拓扑

5.2 仿真结果

5.2.1 请求接入速率

实验 1 研究不同请求接入速率 λ 条件下的 VLCW 性能,结果如表 3 所示。实验结果表明,VLCW 在视频请求接入速率较大的情况下显著优于其他 2 种算法,说明其缓存视频内容的复用率更高。这得益于其基于视频片段的缓存算法在缓存空间有限的情况下可以缓存更多种类的视频内容,达到更高的缓存命中率和缓存利用率。在 λ 较大的情况下,相对于其他 2 种算法的优势更明显。随着 λ 的逐渐增大,VLCW 的缓存空间利用率提升了 3 倍~5 倍。

表 3 不同请求接入速率下的算法性能比较

算法	性能指标	$\lambda=5$	$\lambda=10$	$\lambda=15$	$\lambda=20$	$\lambda=25$
NNC + LRU	r_{cache}	0.005 0	0.001 2	0.001 3	0.002 3	0.002 2
	r_{server}	2.300 0	3.720 0	3.040 0	4.150 0	5.180 0
	η	0.910 0	1.970 0	1.700 0	2.680 0	3.320 0
Rdm + LRU	r_{cache}	0.000 7	0.001 6	0.001 7	0.002 6	0.002 6
	r_{server}	1.770 0	3.610 0	3.430 0	3.860 0	5.760 0
	η	0.480 0	1.690 0	1.830 0	2.520 0	2.820 0
VLCW	r_{cache}	0.011 6	0.014 3	0.015 4	0.016 3	0.016 2
	r_{server}	2.940 0	8.500 0	12.380 0	20.360 0	21.580 0
	η	1.110 0	4.840 0	7.040 0	10.390 0	12.200 0

5.2.2 视频数量

实验 2 研究不同视频数量 N 下的 VLCW 性能。λ 值固定为 20 请求/min,N 的范围为 5~25,实验结果如表 4 所示。实验结果表明,VLCW 在缓存资源利用率 r_{cache} 上比其他算法优越。对于 r_{cache} 和 η,其他 2 种算法在 N 较小时也能达到较好的性能,因为视频请求集中在小部分视频内容上。然而,VLCW 在视频数量增加时仍能获得较好的性能,因为基于小颗粒度视频片段的缓存算法能够更高效地利用缓存资源。

表 4 不同视频数量下的算法性能比较

算法	性能指标	$N=5$	$N=10$	$N=15$	$N=20$	$N=25$
NNC + LRU	r_{cache}	0.003 7	0.001 9	0.001 2	0.001 1	0.000 8
	r_{server}	30.860 0	4.180 0	3.190 0	2.510 0	1.860 0
	η	15.990 0	2.510 0	1.620 0	1.340 0	0.830 0
Rdm + LRU	r_{cache}	0.004 0	0.002 3	0.001 6	0.001 4	0.000 8
	r_{server}	21.600 0	5.690 0	3.010 0	2.420 0	1.620 0
	η	10.890 0	2.790 0	1.500 0	1.280 0	0.590 0
VLCW	r_{cache}	0.015 0	0.014 0	0.014 8	0.013 2	0.014 5
	r_{server}	43.200 0	20.500 0	10.850 0	7.920 0	5.120 0
	η	22.170 0	9.850 0	6.140 0	4.260 0	3.250 0

5.2.3 缓存空间大小

实验 3 研究单个交换机的缓存空间 M 下的 VLCW 性能。缓存空间在整个视频大小的 1 倍~5 倍范围内变化,λ 和 T 值都与实验 2 中一样,N 设置为 20,实验结果如表 5 所示。结果表明,当 M 增大时,NNC + LRU 和 Rdm + LRU 都能获得显著的性能提升,但是缓存效率仍保持较低的增长趋势,即视频缓存总长度的增长相对较快。而对于 VLCW,所有 3 个指标都几乎保持不变,这意味着 VLCW 可以在降低冗余流量和节省存储资源方面获得更好的性能。

表 5 不同缓存空间下的算法性能比较

算法	性能指标	$M=1$	$M=2$	$M=3$	$M=4$	$M=5$
NNC + LRU	r_{cache}	0.000 8	0.000 9	0.000 8	0.000 9	0.001 0
	r_{server}	1.520 0	2.200 0	2.780 0	3.980 0	4.760 0
	η	0.480 0	1.070 0	1.490 0	2.240 0	2.670 0
Rdm + LRU	r_{cache}	0.000 8	0.001 3	0.001 2	0.001 4	0.001 4
	r_{server}	1.410 0	2.150 0	2.470 0	3.980 0	3.630 0
	η	0.320 0	0.870 0	1.080 0	2.050 0	1.700 0
VLCW	r_{cache}	0.014 6	0.014 8	0.014 1	0.015 6	0.015 7
	r_{server}	7.500 0	6.590 0	7.410 0	5.970 0	7.130 0
	η	4.070 0	3.690 0	4.090 0	3.820 0	4.330 0

6 结束语

为减少 VoD 服务引起的冗余流量,本文设计一种具有网内缓存的软件定义多媒体流化系统,并提出一种变长间隔的缓存算法,提高存储资源的利用率和视频流的聚合效率。在此基础上,搭建一个原型系统来验证该算法的可行性和有效性。为全面评

估所提算法的性能,本文定义了流化效率、缓存效率和效率因子等性能指标,并将 VLCW 算法与已有的 NNC + LRU 和 Rdm + LRU 算法进行对比实验。结果表明,在 VLCW 算法的优化下,服务器负载下降了 50% ,缓存空间利用率提高了 3 倍 ~ 5 倍。由于 VLCW 算法仅适用于 VoD 系统中的 RTP 视频流化,因此下一步将研究网内缓存在其他应用场景下的优化算法,例如超清视频直播、VR 视频点播等,通过缓存减少此类高网络资源需求应用的带宽消耗,提升服务质量。

参考文献

[1] Cisco systems. Cisco visual networking index: forecast and methodology [EB/OL]. [2018-03-17]. https://www. cisco. com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490. html.

[2] ULLAH I, DOYEN G, BONNET G, et al. A survey and synthesis of user behavior measurements in P2P streaming systems [J]. IEEE Communications Surveys and Tutorials, 2012, 14(3): 734-749.

[3] MITRA S, AGRAWAL M, YADAV A, et al. Characterizing Web-based video sharing workloads [J]. ACM Transactions on the Web, 2011, 5(2): 1-27.

[4] PENG Gang. CDN: content distribution network [EB/OL]. [2018-03-17]. https://arxiv. org/abs/cs/0411069.

[5] 李昕冉,周金和. 基于复杂网络的绿色 CDN 社团结构划分[J]. 计算机工程,2018,44(3):119-126.

[6] EISLEY N, PEH L S, LI Shang. In-network cache coherence[C]//Proceedings of the 39th Annual IEEE/ACM International Symposium. Washington D. C., USA: IEEE Press, 2006: 321-332.

[7] SEZER S, SCOTT-HAYWARD S, CHOUHAN P K, et al. Are we ready for SDN? implementation challenges for software-defined networks [J]. IEEE Communications Magazine, 2013, 51(7): 36-43.

[8] 谢凡,李勇,柳嘉强,等. 基于 SDN-NFV 的移动核心网联动机制研究[J]. 计算机工程,2017,43(2):144-149.

[9] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. OpenFlow: enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.

[10] MANGILI M, MARTIGNON F, CAPONE A. Stochastic planning for content delivery: unveiling the benefits of network functions virtualization [C]//Proceedings of IEEE International Conference on Network Protocols. Washington D. C., USA: IEEE Press, 2014: 344-349.

[11] KIM J, GEMOH T, PARK S, et al. An efficient multimedia transmission control methodology based on NFV [C]// Proceedings of the 5th International Conference on IT Convergence and Security. Washington D. C., USA: IEEE Press, 2015: 1-4.

[12] PSARAS I, CHAI W K, PAVLOU G. In-network cache management and resource allocation for information-centric networks[J]. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(11): 2920-2931.

[13] KANAI K, MUTO T, KATTO J, et al. Proactive content caching for mobile video utilizing transportation systems and evaluation through field experiments [J]. IEEE Journal on Selected Areas in Communications, 2016, 34(8): 2102-2114.

[14] 张琳凯,杨恩众,姚振,等. SDN 分层组播视频会议系统设计与实现[J]. 小型微型计算机系统,2017,38(3):425-430.

[15] 姜俊超,朱坤杰,张云飞,等. SDN 中 DASH 路由规划和码率调节联合决策算法[J]. 小型微型计算机系统,2017,38(6):1169-1174.

[16] YANG Jian, ZHU Kunjie, RAN Yongyi, et al. Joint admission control and routing via approximate dynamic programming for streaming video over software-defined networking [J]. IEEE Transactions on Multimedia, 2017, 19(3): 619-631.

[17] GEORGOPOULOS P, BROADBENT M, FARSHAD A, et al. Using software defined networking to enhance the delivery of video on-demand [J]. Computer Communications, 2015, 69(C): 79-87.

[18] 李聪,温东新. 一种基于突发集中性访问模式的缓存替换算法[J]. 计算机工程,2017,43(1):105-108.

[19] LI Suoheng, XU Jie. Trend-aware video caching through online learning [J]. IEEE Transactions on Multimedia, 2016, 18(12): 2503-2516.

[20] CLAEYS M, BOUTEN N, DE VLEESCHAUWER D, et al. Cooperative announcement-based caching for video on-demand streaming [J]. IEEE Transactions on Network and Service Management, 2016, 13(2): 308-321.

[21] DAN A, SITARAM D. Generalized interval caching policy for mixed interactive and long video workloads [M]. San Francisco, USA: Morgan Kaufmann Publishers Inc., 1996.

[22] DAN A, SITARAM D. Buffer management policy for an on-demand video server: US5572645[P]. 1996-11-05.

[23] SCHULZRINNE H. RTP tools [EB/OL]. [2018-03-17]. http://www. cs. columbia. edu/irt/software/rtptools/.

[24] CHEN H Y, LIU N B, SHIAH C W, et al. A novel multimedia synchronization model and its applications in multimedia systems[J]. IEEE Transactions on Consumer Electronics, 1995, 41(1): 12-22.

[25] Open networking lab. POX Wiki[EB/OL]. [2018-03-17]. https://openflow. stanford. edu/display/ONL/POX + Wiki.

[26] JACOBSON V, SMETTERS D K, THORNTON J D, et al. Networking named content[J]. Communications of the ACM, 2009, 55(1): 117-124.

[27] CHAI W K, HE Diliang, PSARAS I, et al. Cache "less for more" in information-centric networks [C]// Proceedings of International Conference on Research in Networking. Berlin, Germany: Springer, 2012: 27-40.

[28] Mininet team. Mininet project [EB/OL]. [2018-03-17]. http://mininet. org//.

编辑　陆燕菲