ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.4, April 2006, pp.876−884
DOI: 10.1360/jos170876

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种 P2P 环境下的 VoD 流媒体服务体系*

刘亚杰+, 窦文华

(国防科学技术大学 计算机学院,湖南 长沙 410073)

A Video-on-Demand Streaming Service Architecture in P2P Environment

LIU Ya-Jie+, DOU Wen-Hua

(School of Computer, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: Phn: +86-731-4573550, Fax: +86-731-4573550, E-mail: liuyajie@nudt.edu.cn, http://www.nudt.edu.cn

Liu YJ, Dou WH. A video-on-demand streaming service architecture in P2P environment. *Journal of Software*, 2006,17(4):876−884. http://www.jos.org.cn/1000-9825/17/876.htm

Abstract: Providing video on demand service over the Internet in a scalable way is a challenging problem. This paper proposes an architecture for video on demand streaming in peer-to-peer environment, in which each peer node has a fixed-size FIFO buffer to cache the most recent content of the video stream it receives and can provide service to subsequent reached proper peer nodes. It has the following properties: 1) it utilizes a distributed control protocol to support the joining and leaving processes of peer nodes in a scalable way; 2) it considers the issue of integrity of the received program in service recovering process of the interrupted nodes. Performance studies based on simulation are carried out, and the results show that the system architecture outperforms a recently proposed system architecture P2VoD in a number of important performance metrics such as the server's load, client join rejection probability, network resource usage ratio, program integrity ratio and so on.

Key words: peer-to-peer network; VoD; fault tolerant; algorithm; architecture

摘 要: 在 Internet 上提供大规模的 VoD 服务是一项具有挑战性的工作.提出了一种基于 P2P(peer to peer)方法的 VoD 服务体系 PeerVoD,它能够以较小的服务器代价实现大规模的 VoD 应用.PeerVoD 中的每个节点均使用定长的 FIFO 缓存队列来保存其最近所接收到的数据,以便为后续到达的节点提供服务.它具有如下特点:1)采用分布式控制协议以支持节点的加入和离开,系统具有良好的可扩展性;2) 服务被中断的节点在进行中断恢复时,考虑了节点对目标节目接收的完整性.仿真实验表明:同等条件下,PeerVoD 体系在服务器负载、节点加入时被拒绝的概率、网络资源利用率以及节目完整性等性能指标上,均优于同类体系 P2VoD.

关键词: 对等网络;视频点播;容错;算法;体系
中图法分类号: TP393 文献标识码: A

由于视频流服务对带宽资源的要求高、服务时间长,使得在 Internet 上提供 VoD 视频点播极具挑战性,特别是当某个节目趋向流行时,系统会在短时间内收到大量异步服务请求,而传统的在服务器端为每个请求单独分

* Supported by the National Natural Science Foundation of China under Grant Nos.60433040, 90104001 (国家自然科学基金); the National Grand Fundamental Research 973 Program of China under Grant No.2003CB314002 (国家重点基础研究发展规划(973))

Received 2004-12-29; Accepted 2005-06-02

配一条流的 C/S 模式无法容纳大规模的点播请求.因此,如何使系统具有高可扩展性也就成为其核心问题.IP 组播以其多路复用的方式,能够减缓服务器和网络的负载.在此基础上,针对 VoD 服务,人们提出了 Batch,Patching[1,2],Periodic Broadcast[3,4]等调度策略来满足异步的服务请求.但众多原因,如实现方面的复杂性、拥塞控制、可靠性管理等方面的不足,使基于 IP 组播的 VoD 服务在近期内难以得到广泛应用.P2P 流媒体[5−9]通过利用普通节点(peer 节点)的带宽、存储等资源为其他节点提供服务,能够减少对服务器带宽等资源的占用.而针对 VoD 点播服务,通过在 Peer 节点上分配一段定长或不定长的数据空间以缓存其所接收到的数据,并为其他请求该数据段的节点提供服务[10−12],不但可以满足 VoD 中异步模式的服务请求,而且可以缓解服务器的负载.

一般来说,基于 P2P 模式的 VoD 系统面临如下挑战:1) 节点搜索,主要是指当节点加入系统时,如何在组播树中快速、有效地搜索到合适的父节点;2) 系统容错,Peer 节点可能随时离开系统或失效,从而中断其子节点的服务.如何让被中断的节点能够快速、有效地进行中断恢复,是系统面临的核心问题;3) 协议开销,由于组播树的建立和维护依赖于控制协议,如何设计控制协议,使之具有良好的可扩展性也是系统的关键问题;4) QoS 保证,主要是指在 Peer 节点存在离开或失效的前提下,如何保证节目的播放质量,如完整性、连续性等.

本文提出了一种 P2P 环境下的 VoD 服务体系 PeerVoD,其基本思想是:在系统每个节点上分配一段固定长度的 FIFO 队列以缓存其最近所接收到的数据,并为其他点播请求该数据段的节点提供服务.PeerVoD 具有如下几个特点:1) 采用分布式的控制协议,通过在每个节点上维护有限个其他节点的状态信息,使得当节点加入时,能够在短时间内找到合适的父节点;而当节点上发生离开或者失效行为时,子节点能够通过其所维护的状态信息快速而准确地找到新的父节点;2) 节点的失效或离开一般不涉及到服务器,从而减轻了服务器的负载,系统具有良好的可扩展性;3) 服务被中断的节点在进行中断恢复时,考虑了节点对目标节目接收的完整性;4) 考虑了节点在网络带宽等资源方面所体现出来的异构性.

在已有的相关研究工作中,P2Cast[10]实际上是 Patching 策略在 P2P 环境下的一种扩展.它借鉴了传统 IP 组播中 Patching 策略,每个 Peer 节点的 Patching 流既可以从服务器获取,也可以从其他 Peer 节点获取.DirecStream[11]与 PeerVoD 相同之处在于节点数据缓存机制;不同之处是 DirectStream 通过中心节点(一般为服务器)来记录维护系统中所有节点的状态,并采用中心控制协议;而 PeerVoD 中对节点状态的记录维护以及所采用的控制协议均为分布式的.P2VoD[12]与本文工作相对而言最具有相关性,它引入“代(generation)”的概念,同一“代”中所有节点上所缓存的最旧的数据块相同,以便节点服务被中断时直接与其上一“代”节点建立链接即可恢复数据;P2VoD 与 PeerVoD 的区别主要有 3 点:一是 P2VoD 中为维持“代”的关系,其节点上所分配的 FIFO 缓存队列长度是不定长的,而 PeerVoD 中不存在“代”的关系,其节点的缓存队列长度是定长的;二是在进行中断恢复时,P2VoD 没有考虑数据接收的完整性;三是 P2VoD 中需要在服务器上记录保留系统中每个节点的信息,而在 PeerVoD 中对节点信息的保存是分布式的.

1 PeerVoD 系统结构

假设目标节目以 CBR 方式编码和传输,其播放时长为 T,目标节目的最小存储和传输单元为数据块,每块的播放时间为一个单位时间,所有数据块按播放先后顺序从 1~T 予以编号.假设节点加入系统的时间序列服从强度为λ的泊松分布,每个节点上所分配的 FIFO 缓存队列长度为τ.对 Peer 节点 p_i,用 t_i 表示其加入系统的时刻,a_i 表示其 IP 地址,min(p_i)表示 p_i 上所缓存数据块的最小编号,max(p_i)表示 p_i 上所缓存数据块的最大编号,$next(p_i)$表示 p_i 将要接收的下一个数据块的编号.另外,为了系统描述的简单,这里假设节点加入时均从目标节目中第 1 个数据块开始申请,实际上,PeerVoD 同样适用节点加入时从目标节目中任意位置开始申请的情形.首先,我们给出描述系统结构所需要的几个定义:

定义 **1**. 系统中共享从服务器 S 一个通道所流出数据的 Peer 节点集合称为一个簇.

定义 **2**. 簇中直接从服务器 S 获取数据的节点称为簇首节点;簇中没有子节点的节点称为簇叶节点.

定义 **3**. 对于系统中的任一节点,如果该节点上还缓存有目标节目的第 1 个数据块,则称该节点为开放节点;

而对于系统中的任意簇,如果其中至少还包含一个开放节点,则称该簇为开放簇,否则为闭合簇.

定义 **4**. 组播树上沿数据转发路径处于服务器 S 和 p_i 之间的所有节点(不包括 S 和 p_i),称为 p_i 的上游节点;而沿数据转发路径处于 p_i 和叶节点之间的所有节点(不包括 p_i),称为 p_i 的下游节点.

定义 **5**. 对系统中的任意节点 p_i 和 p_j,如果 p_i 在当前时刻缓存了数据块 $next(p_j)$,且 p_i 不为 p_j 的父节点,则称 p_i 为 p_j 的候选父节点.

对 Peer 节点 p_i,用 h_i 表示其所在簇的簇首节点,c_i 表示其所在簇的簇号,P_i 表示 p_i 的候选父节点集合.

由于在 Peer 节点上存在离开或失效等动态行为,而当 Peer 节点作为服务节点时,这些动态行为将导致其子节点服务的中断.考虑到系统中节点所缓存的数据块内容一般处于动态更新状态,且节点的服务被中断后一般要求从被中断处开始重新获取数据,而现有的容错机制,如直接从祖父节点[5]或父辈节点[12]恢复获取数据的方式,一般不能保证节点可以从被中断处恢复接收数据,因而,这些机制在此都不具有普遍适用性.这里,我们提出一种新的容错机制,它在每个节点上保留其候选父节点的状态信息,而当数据服务被中断时,节点能够基于这些状态信息快速而准确地搜索到新的父节点,从而避免了在整个组播树中进行搜索,且不必在每次中断恢复过程中访问服务器,有效地减轻了服务器的负载.

定理 **1**. 在不考虑节点离开或失效的前提下,对于系统中节点 p_i,其候选父节点集合 P_i 的平均大小为 $\lambda\tau-1$.

证明:由泊松分布及数学期望的计算方法可知,详细过程略. □

定理 1 从理论上指出了节点为维护其候选父节点状态信息而增加的平均负载.

1.1 控制协议

在 PeerVoD 中,为了维护组播树的拓扑结构以及在组播树的节点之间正确地转发数据,需在每个节点上维持其父、子节点的有关信息;而为了让服务被中断的节点能够快速地进行中断恢复,需在每个节点上维持其候选父节点的有关信息.此外,为支持节点的快速加入和中断恢复,对于每个簇首节点,还需维护其簇中部分节点的有关信息.

以节点 p_i 为例,对于其父节点,仅需在 p_i 上记录维护其 IP 地址、节点状态这两类信息.对于其任一子节点,需在 p_i 上记录维护 3 类信息:IP 地址、节点状态、子节点当前所请求的数据块,后者作为下一时刻向该子节点转发哪个数据块的依据.对于其簇首节点,仅需要维护其 IP 地址信息.对于其任一候选父节点,除了记录其 IP 地址外,还需维持其和 p_i 之间的"心跳线",以确认其当前是否仍处在系统中.另外,考虑到节点的失效或离开将影响其下游节点缓存内容的更新进度,从而导致在整个服务过程中集合 P_i 所包含的节点对象也可能需要更新,即随着系统的运行,P_i 中某些节点因已不能提供数据块 $next(p_i)$而需从 P_i 中删除,而原先某些不属于 P_i 的节点因可以提供数据块 $next(p_i)$而需要加入 P_i.因此,对任意节点 $p_j(p_j\in P_i)$,在每次心跳检测过程中,还需查看其缓存的数据是否对 p_i 有用,即是否有 $\min(p_j)\le next(p_i)\le\max(p_j)$,若否,则把 p_j 从 P_i 中删除,并从 p_j 的上游节点($next(p_i)\le\max(p_j)$情形)或下游节点($next(p_i)<\min(p_k)$情形)中,为 p_i 查找新的候选父节点.另外,如果 p_i 为簇首节点,还需在其上维护两类节点的信息:一类是簇中那些还属于开放节点的节点 IP 地址信息;另一类是簇中每个簇叶节点的 IP 地址、缓存数据块范围等信息.这两类信息都可以分别通过让开放节点或簇叶节点定时向其簇首节点发送状态更新消息的途径来获取.对于服务器 S,除了记录每个簇首节点的 IP 地址信息以外,还需在其上记录每个簇所缓存的数据块范围,以便在节点中断恢复过程中提供可能的帮助,该信息可以通过让簇首节点定时地向服务器发送更新消息来获取.

1.2 节点加入

节点(设为 p_i)在加入系统之前,首先向服务器 S 发送一条加入系统请求消息;S 在收到该消息后,分两种情形进行处理:

情形 **1**. 如果系统中没有开放簇,则查看其本身是否还有空闲的数据通道:如果有,则向 p_i 返回加入成功的消息并指明其父节点为 S,p_i 在收到该信息后直接加入 S,并从 S 分配获取一个新的簇号,置 p_i 为空;如果没有,则向 p_i 返回加入失败的消息,节点加入失败;

情形 **2**. 如果系统中还有开放簇,则向所有开放簇的簇首节点转发 p_i 申请加入系统的消息,簇首节点在收到该消息后,直接向 p_i 返回簇中所有的开放节点.假设 p_i 所收集到的开放节点集合为Ω,p_i 根据预先定义的某种父节点选取原则,从Ω中挑选一个父节点.若父节点选取成功(设为 p_j), p_i 直接加入 p_j,置 p_i 为$\Omega-\{p_j\}$,置 c_i 为 c_j,置 h_i 为 h_j;若父节点选取失败,则直接申请加入服务器 S,其过程类似情形 1,不同之处是置 P_i 为Ω.

对于节点 p_i 和集合Ω,这里提出 3 种不同的父节点选取原则,倾向于 3 种不同的目的:

- 最快匹配原则:即在Ω中选择第一个有充足剩余带宽容纳 p_i 加入的节点.这种选择方式不必在Ω中查询、比较每一个节点,从而可以让 p_i 快速加入系统;
- 最小延迟原则:即在Ω中选择有充足剩余带宽容纳 p_i 加入,且和 p_i 之间延迟最小的节点.由于组播树建立在应用层,其拓扑结构与物理网络的实际拓扑普遍存在不一致.最小延迟选择方法能在某种程度上减少这种拓扑的不一致,同时也可以减少对骨干网络带宽的消耗;
- 最小高度原则:即在Ω中选择有充足剩余带宽容纳 p_i 加入,且其在组播树中高度最小的节点.最小高度原则通过选择高度最小的节点作为父节点,能够减小整个组播树的高度,从而减少节点离开或失效对其他节点的影响,提高节目在节点上的平均播放质量.

1.3 节点离开与失效

节点的离开或失效都将引起子节点服务的中断,而为了恢复被中断的服务,必须解决两个问题:一是中断节点的检测,二是重新加入时父节点的搜索选择.对于中断节点的检测,如果节点为主动离开,则在离开前可通知其所有的子节点;如果节点为失效,由于其在离开前无法通知其子节点,因此约定,如果节点在规定时间内没有收到任何从其父节点转发而来的数据,则给其父节点发送检测消息,并最终可定位已失效的节点.为减少节点失效或离开给系统带来的开销,规定只有失效或离开节点的子节点才能发起重新加入请求.

对于中断节点的重新加入过程,可能会涉及到其候选父节点、服务器及其他节点.以节点 p_i 重新加入为例,表 1 给出了在执行节点重新加入分布式算法过程中所涉及到的主要消息及其参数所代表的意义.下面分别描述在各类节点上所执行的算法.

Table 1 Messages used in rejoining algorithm

表 **1** 重新加入算法中所用到的消息

Message name	Message parameters	Description
REJOIN_SEARCH_BIDIRECTION	a_i,$next(p_i)$	Send from p_i to p_k ($p_k \in P_i$), it meaning is to search new father node(s) on multicast tree bidirectionally which starts from the destination node
REJOIN_SEARCH_DOWN	a_i,$next(p_i)$	Sent by p_i, it meaning is to search new father node(s) on multicast tree in the data retransmitting direction
REJOIN_SEARCH_UP	a_i,$next(p_i)$	Sent by p_i, its meaning is to search new father node(s) on multicast tree in opposite direction of the data retransmitting
REJOIN_QUERY	a_i,$next(p_i)$,C_i	Send from p_i to S, it meaning is to query S that if any node locating in other clusters (excluding the cluster C_i) has cached the data block $next(p_i)$
REJOIN	a_j	Send from p_i to S, it meaning is that wish to join the server S directly
REJOIN_BE_CANDIDATE	a_i	Send from p_j to p_i, its meaning is that p_j can be a candidate father node of p_i
CLUSTER_HEAD_CHANGE	h_i,c_i	Sent by p_i, its meaning is to notify the destination node to change its cluster number and its cluster header

1. 节点 p_i

Step 1. 查看其候选父节点集合 P_i,如果 P_i 为空,转 Step 3,否则给 P_i 中的每个节点发送 REJOIN_SEARCH_BIDIRECTION 消息,置集合Ω为空,启动超时计时器,转 Step 2;

Step 2. 等待接收消息:如果收到的消息为 REJOIN_BE_CANDIDATE,则把消息中包含的节点加入到Ω.如果计时器超时,则查看Ω是否为空:如果是,则转 Step 3;否则,按第 1.2 节所述 3 种父节点选取原则之一,从Ω中挑

选一个父节点.如果父节点选取成功,转 Step 6;否则,置Ω为空,转 Step 3;

Step 3. 置集合 C_i 为$\{c_k|p_k\in P_i\}$,C_i 为 P_i 中成员所在簇的簇号集合.p_i 与服务器 S 建立连接并向 S 发送 REJOIN_QUERY 消息请求,如果 S 返回的消息为 REJION_QUERY_WAIT,启动超时计时器,转 Step 4;如果 S 返回的消息为 REJOIN_QUERY_NONE,转 Step 5;

Step 4. 等待接收消息:如果收到的消息为 REJOIN_BE_CANDIDATE,则把消息中包含的节点加入到Ω.如果计时器超时,查看Ω是否为空:如果是,则转 Step 5;否则,按第 1.2 节所描述的某种父节点选取原则,从Ω挑选一个父节点.如果父节点选取成功,转 Step 6;否则,转 Step 5;

Step 5. p_i 与服务器 S 建立连接并向 S 发送 REJOIN 消息请求,如果 S 返回的是 REJOIN_SUCCESS,转 Step 6;如果 S 返回的是 REJOIN_FAIL,转 Step 7;

Step 6. p_i 加入系统并从新父节点获取数据,置 P_i 为 $P_i\cup\Omega$.如果新父节点为服务器 S,则从 S 获取一个新的簇号并向其所有子节点发送 CLUSTER_HEAD_CHANGE 消息,算法结束;如果新父节点为 Peer 节点(假设为 p_j),比较 c_i 和 c_j:如果相等,则算法结束;如果不等,则置 c_i 为 c_j,置 h_i 为 h_j,并向其所有子节点发送 CLUSTER_HEAD_CHANGE 消息,算法结束;

Step 7. p_i 重新加入失败,p_i 向其所有子节点发送重新加入失败消息,并退出系统.

2. 服务器 S

Step 1. 循环接收消息并检查消息类型:如果为 REJOIN_QUERY,转 Step 2;如果为 REJOIN,转 Step 3;

Step 2. 取消息中的参数 a_i,$next(p_i)$和 C_i.检查系统中的每个簇:如果该簇的节点上缓存了数据块 $next(p_i)$且其簇号不属于 C_i,则向该簇簇首节点发送消息 REJOIN_SEARCH_DOWN,向 p_i 发送消息 REJOIN_QUERY_WAIT;如果系统中不存在满足上述条件的簇,向 p_i 发送消息 REJOIN_QUERY_NONE;

Step 3. 检查其本身是否还有剩余空闲通道:如果有,向 p_i 发送消息 REJOIN_SUCCESS;如果没有,则向 p_i 发送消息 REJOIN_FAIL.

3. 其他节点

用 p_k 代表 p_i 重新加入过程中的任一其他节点.在下述算法中,对从 p_k 所发出的消息,如不加特殊说明,其参数值等同于 p_k 所接收到的上一个消息的参数值.

Step 1. 循环接收消息并检查消息类型:如果为 REJOIN_SEARCH_BIDIRECTION,转 Step 2;如果为 REJOIN_SEARCH_UP,转 Step 3;如果为 REJOIN_SEARCH_DOWN,转 Step 4;如果为 CLUSTER_HEAD_CHANGE,转 Step 5;

Step 2. 如果 $next(p_i)<\min(p_k)$,向其所有子节点发送消息 REJOIN_SEARCH_DOWN;如果 $next(p_i)>\max(p_k)$,向其父节点发送消息 REJOIN_SEARCH_UP;如果 $\min(p_k)\le next(p_i)\le\max(p_k)$,向 p_i 发送消息 REJOIN_BE_CANDIDATE,消息参数值为 a_k,并进一步向其父节点发送消息 REJOIN_SEARCH_UP,向其所有子节点发送消息 REJOIN_SEARCH_DOWN;

Step 3. 如果 $\min(p_k)\le next(p_i)\le\max(p_k)$,向 p_i 发送消息 REJOIN_BE_CANDIDATE,消息参数值为 a_k,并进一步向其父节点发送消息 REJOIN_SEARCH_UP;如果 $next(p_i)>\max(p_k)$,向其父节点发送消息 REJOIN_SEARCH_UP;如果 $next(p_i)<\min(p_k)$,丢弃本消息;

Step 4. 如果 $\min(p_k)\le next(p_i)\le\max(p_k)$,向 p_i 发送消息 REJOIN_BE_CANDIDATE,消息参数值为 a_k,并进一步向其所有子节点发送消息 REJOIN_SEARCH_DOWN;如果 $\min(p_k)>next(p_i)$,向其所有子节点发送消息 REJOIN_SEARCH_DOWN;如果 $\max(p_k)<next(p_i)$,丢弃本消息;

Step 5. 重置 p_k 的簇号和簇首为 CLUSTER_HEAD_CHANGE 消息中所包含的簇号和簇首,并把该消息转发给所有的子节点.

对于上述算法中超时计时器阈值的设置:如果其值设置得太小,可导致系统来不及检测到某些合适的候选父节点;而如果设置得太大,会导致节点服务被中断的时间太长,影响节点的播放质量.但由于系统中的每个节点在定时执行其候选父节点心跳检测的过程中,均对其候选父节点进行了筛选,因此,算法中的超时计时器的设

置不用太大,一般为节点之间平均延迟的 3~5 倍即可.

下面分析节点重新加入算法的性能及正确性.从节点 p_i 上所执行的算法可知,在新父节点的搜索过程中,首先以所有候选父节点(如果有)为起点,在其所在簇中沿组播树进行双向局部搜索;如果不能搜索到合适的父节点,再通过服务器查询其他簇上是否缓存了 p_i 所请求的数据块:若否,则试图直接加入服务器;否则,通过相关簇的簇首沿组播树自上而下进行局部搜索,如果能搜索到合适的父节点,则直接加入,否则试图直接加入服务器.因此,上述分布式算法能够帮助被中断的节点恢复数据服务.此外,在上述算法中,服务器(中心节点)只需记录维护系统中每个簇首节点的状态信息,而且只有部分节点在重新加入过程中需要依赖于中心节点,因而极大地提高了系统的可扩展性.

引理 **1**. 对系统中任意节点 p_i 和 p_j,如果 p_i 为 p_j 的上游节点,则有 $\min(p_i)>\min(p_j)$且 $\max(p_i)>\max(p_j)$;如果 p_i 为 p_j 的下游节点,则有 $\min(p_i)<\min(p_j)$且 $\max(p_i)<\max(p_j)$.

定理 **2**. 节点的重新加入过程,不会在组播树中产生环或被分离的子树.

证明:在组播树中能产生环或被分离子树的情形有两种:一种是节点 p_i 在重新加入的过程中选择了其下游节点中的某个节点作为父节点;另一种是节点 p_i 在重新加入过程中选择了 p_j 作为父节点,而 p_j 或其上游节点也同时在经历重新加入的过程,且选择了 p_i 或其下游的某个节点作为其父节点.对于第 1 种情形,由 $next(p_i)=\max(p_i)+1$ 及引理 1 可知,p_i 的下游节点上不可能缓存有数据块 $next(p_i)$,因而情形 1 不会产生;对于第 2 种情形,以 p_i 在重新加入过程中选择 p_j 作为父节点,p_j 也同时在经历重新加入过程且选择了 p_i 作为父节点为例:由于 p_i 选择 p_j 作为父节点,则有不等式 $\min(p_j)\le next(p_i)\le\max(p_j)$成立;而 p_j 也选择 p_i 作为父节点,则有不等式 $\min(p_i)\le next(p_j)\le\max(p_i)$成立;把上述两个不等式相加,则有不等式 $\min(p_i)+\min(p_j)\le next(p_i)+next(p_j)\le\max(p_i)+\max(p_j)$,而 $next(p_i)+next(p_j)=\max(p_i)+\max(p_j)+2$,矛盾;情形 2 中的其他情况可用类似方法推出矛盾.因此,情形 2 也不会产生.证毕. □

2 系统性能评估

2.1 服务器负载理论分析

这里先不考虑节点的失效或离开行为,且假设每个节点至少能够容纳一个子节点,并从理论上对服务器的负载进行分析.

定理 **3**. 假设节点加入系统过程服从强度为 λ 的泊松分布,目标节目的时长为 T,每个节点上所分配的 FIFO 缓存队列长度为 τ,那么,目标节目在服务器 S 上平均占用的通道数为 $\lambda T\mathrm{e}^{-\lambda\tau}$.

证明:假设 p_1 为第 1 个到达系统的节点,其到达时刻为 0,p_1 直接从服务器 S 获取数据即创建一个新簇,p_1 为簇首.由于 p_1 上缓存了其最近 τ 时间内接收到的数据,对于随后到达的节点 p_2,如果 p_2 的到达时刻属于 $(0,\tau)$,那么 p_2 可直接从 p_1 获取数据且同样缓存 τ 时间的数据;否则,p_2 将直接从 S 上获取数据,即 p_2 也将创建一个新簇且成为新簇簇首;如此类推.用 x 表示 S 上为目标节目所分配的通道数,y 表示簇首之间的时间间隔,那么,在 S 上为目标节目所分配的平均通道数 $E(x)$可表示为

$$E(x)=\frac{T}{E(y)} \tag{1}$$

用 z 表示节点的到达间隔,p 表示事件 $z<\tau$发生的概率,$E(z|z\le\tau)$表示到达间隔小于 τ的这一类节点到达间隔的平均值,$E(z|z>\tau)$表示到达间隔大于 τ的这一类节点到达间隔的平均值,那么 $E(y)$可表示为

$$E(y)=\sum_{i=0}^{\infty}p^i(1-p)(iE(z\mid z\le\tau)+E(z\mid z>\tau)) \tag{2}$$

而 $E(z|z\le\tau)$和 $E(z|z>\tau)$的表达式分别为

$$E(z\mid z\le\tau)=\frac{1}{P\{z\le\tau\}}\int_0^{\tau}zf(z)\mathrm{d}z=\frac{1}{\lambda}-\frac{\tau\mathrm{e}^{-\lambda\tau}}{1-\mathrm{e}^{-\lambda\tau}} \tag{3}$$

$$E(z \mid z>\tau)=\frac{1}{P\{z>\tau\}}\int_{\tau}^{+\infty} zf(z)\mathrm{d}z=\frac{1}{\lambda}+b \tag{4}$$

把式(3)、式(4)代入式(2),并把计算结果代入式(1)得:

$$E(x)=\lambda T\mathrm{e}^{-\lambda\tau} \tag{5}$$

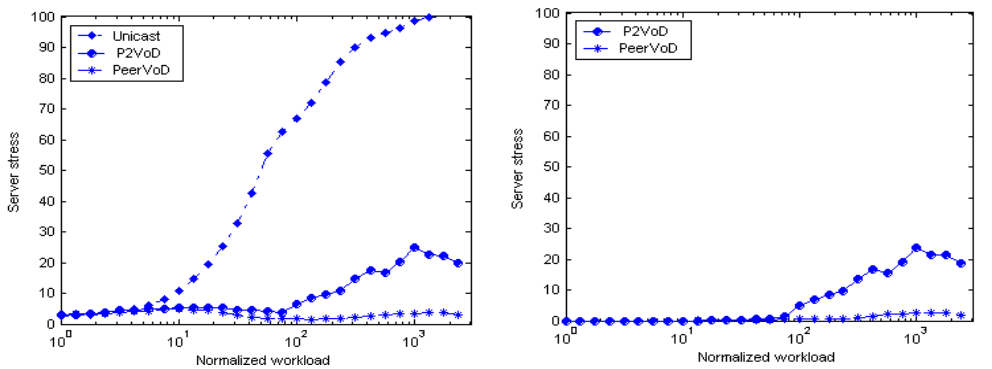
定理 3 证毕. □

从定理 3 可知,服务器的平均负载只与泊松分布参数λ、节目时长 T、FIFO 缓存队列长度τ等因素有关,与系统的规模无关.表明系统具有良好的可扩展性.

2.2 仿真实验及结果分析

在仿真实验中,我们首先用工具 GT-ITM[13]生成一个具有两个层次的网络拓扑,分别代表核心路由层和边缘路由层.核心路由层由一个域组成,它包含 4 个核心路由节点;边缘路由层由 12 个域组成,每 3 个域附属于一个核心路由节点,边缘路由层的每个域中平均包含 8 个边缘路由节点,每个边缘路由节点代表一个网络区域.网络拓扑中的带宽定义如下:核心路由节点之间、核心路由节点和边缘路由节点之间均可同时容纳 25 条流,边缘路由节点之间可同时容纳 7 条流,同属一个网络区域的主机节点之间的网络带宽不受限制.节点间的路由路径由最短路径算法 Dijkstra 决定.其次,在每次实验中,服务器均连接在一个固定的核心路由节点上,其他主机节点以泊松分布加入系统并随机连接在某个边缘路由节点上,实验时间和节目时长 T 均为 100 分钟.对每种参数组合,实验均进行 10 次,实验结果取其平均值,并主要和 P2VoD 的实验结果进行对比.另外,在实验中定义工作负载 W 为λT,缓存比率ρ为τ/T.

首先不考虑系统节点的失效或离开行为.图 1(a)给出了当ρ=10%,在不同工作负载下,两种系统中服务器负载情况的比较,并与 Unicast 系统进行对比.从图 1(a)可以看出,相比于 Unicast,P2VoD 和 PeerVoD 都能够较大程度地节约服务器的带宽资源,而 PeerVoD 占用的服务器带宽比 P2VoD 更少.当 W 小于 100 时,PeerVoD 和 P2VoD 系统中的服务器负载均在 W=10 左右达到最大,这是因为当 W 达到 10 左右时,由于节点加入系统的时间间隔较大,后续到达的节点不能把系统中已有的 Peer 节点选作父节点,即当节点加入时只能选择服务器作为其父节点;而当 W 进一步增大时,节点在加入时可以选择其他 Peer 节点作为父节点,因而服务器的负载反而会减小;当 W 到达 100 后,两种系统的服务器负载均开始出现一定程度的异常反弹增长,通过对节点加入情况的跟踪,发现这种情况主要是由于 Peer 节点间网络资源不够而引起的.图 1(b)给出了那些节点加入时,虽然系统中有 Peer 节点缓存了第 1 个数据块,但由于和这些 Peer 节点间的网络资源均不充足,因而只有直接加入服务器并导致服务器负载增长情况的统计.通过对图 1(a)和图 1(b)进行比较,可以说明上述对服务器负载异常增长现象的解释是准确的.



(a) Server stress comparison (ρ=10%)

(b) Server stress comparison caused by insufficient network resource (ρ=10%)

(a) 服务器负载对比(ρ=10%)

(b) 因网络资源不够而增加的服务器负载对比(ρ=10%)

Fig.1 Server stress comparison (ρ=10%)

图 1 服务器负载对比(ρ=10%)

图 2 给出了当ρ=10%,在不同的工作负载情况下,节点加入系统时被拒绝的概率.可以看出,在相同的网络条件下,P2VoD 中节点加入时被拒绝的概率大于 PeerVoD,特别是随着工作负载的增加,这种差距更为明显.这是因为随着工作负载的增加,边缘路由器之间的最短路径上无空闲带宽资源的概率也随之增加,而在同等系统参数条件下,节点加入时 PeerVoD 中的候选父节点数目多于 P2VoD,从而有更多的 Peer 节点可供选择作为父节点,在网络资源相对不充足的条件下,导致 PeerVoD 中节点加入时被拒绝的概率小于 P2VoD.图 3 给出了ρ=10%时,不同工作负载情况下的网络资源利用率.从图 3 可以看出,P2VoD 和 PeerVoD 对网络资源的利用率均大于 Unicast,而前两者相比较,PeerVoD 能够更好地利用网络资源,因而也更能够节约服务器的带宽.

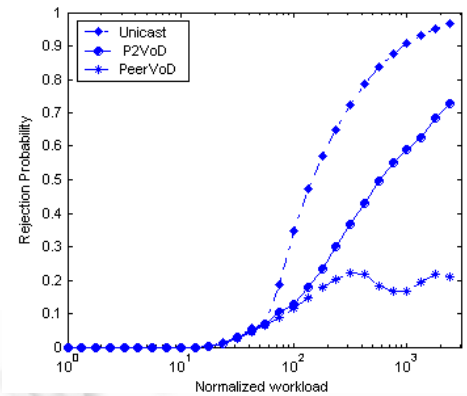


Fig.2 Client rejection probability (ρ=10%)

图 2 节点加入失败概率(ρ=10%)

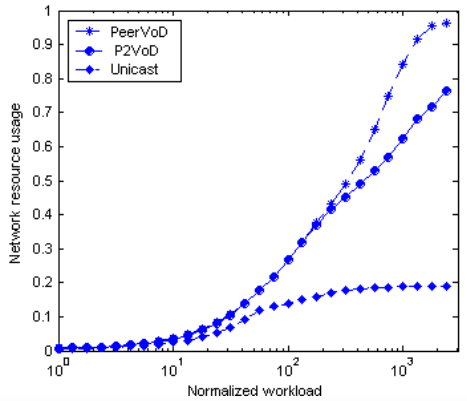


Fig.3 Network resource usage (ρ=10%)

图 3 网络资源利用率(ρ=10%)

为考察节点离开或失效对其他节点所接收节目完整性的影响,我们设计实现了如下实验过程:首先,与上述实验过程相同,让节点加入系统并持续 70 分钟,然后让节点以和加入系统等同的概率分布随机退出系统并持续 30 分钟.在退出系统的节点中,有 50%的节点属于正常的离开,其他节点属于非正常的失效,服务被中断的节点在中断恢复过程中以最快匹配原则选取新的父节点.网络拓扑中,所有路由边的延迟均设置为 50μs,超时计数器的阈值设置为 3s.对于系统中任一节点,如果由于上游节点的退出使其所接收到的节目数据内容出现不完整之处,则称其为数据空洞.定义节点数据完整率为节点所应接收数据的总长度减去节点上所有数据空洞的总长度后所得差值与节点所应接收的数据总长度的比率.定义系统数据完整率为系统中所有节点的数据完整率的均值.图 4 给出了两种系统的数据完整率.可以看出,当系统中有节点发生失效或离开行为时,PeerVoD 中的数据完整性明显大于 P2VoD.这是因为前者在搜索新父节点的过程中,考虑了被搜索的节点上是否缓存了中断开始时的数据块;而后者并没有考虑这一点.从图 4 也可以看出,PeerVoD 中的数据完整率并没有达到 100%.这是因为节点被搜索的时刻和其被真正选作父节点的时刻还存在一定的间隔,导致个别节点上所缓存的目标数据块已被新的数据块所置换,因而也会在极少数进行中断恢复的节点上出现数据空洞,但这种事件发生的概率远小于 P2VoD.

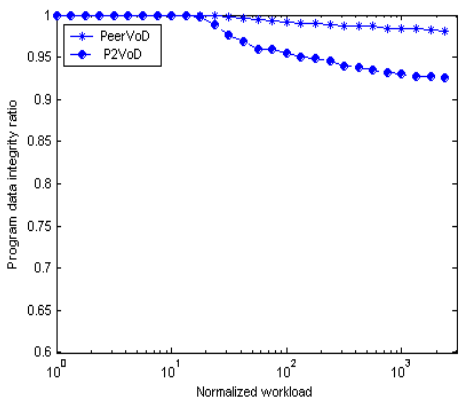


Fig.4 Program data integrity ratio (ρ=10%)

图 4 数据完整率(ρ=10%)

3 结 论

如何在 Internet 上提供大规模的 VoD 点播服务极具挑战性.本文提出了一种基于 P2P 环境的 VoD 服务体系 PeerVoD,它通过在每个节点上分配固定长度的 FIFO 队列缓存最近收到的数据,并为其他合适的节点提供数据服务.该体系具有良好的性价比.仿真实验表明:与同类体系 P2VoD 比较,PeerVoD 在服务器负载、节点加入系统时被拒绝的概率、网络资源利用率以及节目完整性等方面均优于 P2VoD.

References:

[1] Cai Y, Hua KA, Vu K. Optimizing patching performance. In: Dilip D, ed. Proc. of the MMCN'99. Washington: SPIE Press, 1999. 204−216.

[2] Gao L, Towsley D. Threshold-Based multicast for continuous media delivery. IEEE Trans. on Multimedia, 2001,3(4):405−414.

[3] Hu A. Video-on-Demand broadcasting protocols: A comprehensive study. In: Sengupta B, ed. Proc. of the IEEE INFOCOM 2001. New York: IEEE Computer and Communications Societies, 2001. 508−517.

[4] Hua KA, Sheu S. Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems. Computer Communication Review, 1997,27(4):89−100.

[5] Deshpande H, Bawa M, Garcia-Molina H. Streaming live media over a peer-to-peer network. Technical Report, CS-2001-31, Stanford University, 2001.

[6] Tran D, Hua K, Do T. Zigzag: An efficient peer-to-peer scheme for media streaming. In: Proc. of the IEEE INFOCOM 2003. New York: IEEE Computer and Communications Societies, 2003. 1283−1293.

[7] Hefeeda M, Habib A, Botev B, Xu D, Bhargava DB. PROMISE: A peer-to-peer media streaming using collectcast. In: Proc. of the ACM Multimedia 2003. New York: ACM Press, 2003. 45−54.

[8] Xu D, Chai H, Rosenkerg C, Kulkarni S. Analysis of a hybrid architecture for cost-effective media distribution. In: Proc. of the SPIE/ACM MMCN 2003. New York: Elsevier North-Holland, 2003. 353−382.

[9] Rejaie R, Ortega A. PALS: Peer-to-Peer adaptive layered streaming. In: Christos P, Kevin CA, eds. Proc. of the ACM NOSSDAV 2003. New York: ACM Press, 2003. 153−161.

[10] Guo Y, Suh K, Kurose J, Towsley D. P2Cast: P2P patching scheme for VoD service. In: Proc. of the WWW 2003. New York: ACM Press, 2003. 301−309.

[11] Guo Y, Suh K, Kurose J, Towsley D. A peer-to-peer on-demand streaming service and its performance evaluation. In: Proc. of the IEEE ICME 2003. Maryland: IEEE Computer Society, 2003. 649−652.

[12] Do T, Hua K, Tantaoui M. P2VoD: Providing fault tolerant video-on-demand streaming in peer-to-peer environment. In: Proc. of the IEEE ICC 2004. Paris: IEEE Communications Society, 2004. 1467−1472.

[13] Calvert K, Doar M, Zegura E. Modeling internet topology. IEEE Communication Magazine, 1997,35(6):160−163.

刘亚杰(1975－),男,湖南桃源人,博士,主要研究领域为 P2P 网络,流媒体传输调度.

窦文华(1946－),男,教授,博士生导师,主要研究领域为计算机网络.