DSF:一种时序约束下的快速数据分发算法

吴吉庆,彭宇行,刘 锋
(国防科技大学并行与分布处理国防科技重点实验室,湖南长沙 410073)

摘 要: 时序约束下的大规模数据分发在互联网环境下有着越来越广泛的应用.现有的系统大多采用 mesh 结构组织结点,并通过运行在请求结点的调度算法来控制数据的分发.但请求结点只依据自身的需求来调度数据,并不能保证系统的整体服务效果.本文以改善整体服务效果为目标,提出一种面向服务结点的调度算法——时序约束下的快速分发算法 DSF(Deadline Sensitive Fast distribution).该算法的基本思想是:当服务结点面临多个邻居结点的多个数据请求时,选择系统最迫切需要的数据,优先传输给继续服务能力较强的结点,以达到减少迟到数据比例、提高数据传输率、优化系统持续服务能力的目的.实验结果表明,与面向请求结点的调度算法相比,DSF 在流传输质量、分发速率、负载均衡等方面均具有较好的特性.

关键词: 数据分发; 时序约束; 调度
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2012) 02-0365-06
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2012.02.025

DSF:A Fast Data Distribution Algorithm under Timing Constraint

WU Ji-qing, PENG Yu-xing, LIU Feng
(*Key Laboratory of Science and Technology for National Defense of Parallel and Distributed Processing, National University of Defense Technology, Changsha, Hunan* 410073, *China*)

Abstract: The applications for large-scale data distribution under timing constraint have been widely deployed on Internet. Most existing systems organize the involved clients with mesh structure and control the data distribution through the scheduling algorithms running on requesting peers. However, requesting peers request data only according to their own needs, regardless of the overall service quality of the system. Therefore, we take aim at improving the overall service quality of the system and propose a scheduling algorithm running on serving peers: DSF (Deadline Sensitive Fast distribution algorithm), which transfers urgently needed data to the peers that have strong follow-up service ability with high priority, to reduce the ratio of late data chunks and improve the service capacity of the whole system. The experimental results show that DSF has better performance with respect to transmission quality, distribution rate and load balance than the frequently used scheduling algorithms.

Key words: data distribution; timing constraint; scheduling

1 引言

互联网应用,如文件共享[1,2]、开源软件分发[3]、视频直播[4,5]、视频点播[6]和视频会议等,对数据分发的能力有越来越高的要求.其中,视频数据分发对数据的传输有严格的时序约束,更需要有效利用参与应用的客户端带宽以完成数据的快速分发.

基于组播树[7,8]的数据分发是早期采用的视频分发应用技术,该技术在带宽充裕的静态网络中,能够获得良好的用户体验.但是,在异构的互联网环境中,树结构存在不能充分利用结点带宽、动态维护开销大等缺点,因此当前的视频分发系统多采用 Mesh 结构[9,10]组织结点.在 Mesh 结构中,结点之间没有严格的父子关系,各结点通过与邻居周期性地互相交换数据来实现数据分发.为满足视频数据分发的时序约束,每个请求结点维护一个滑动窗口,将数据的请求范围限制在临近播放时刻的数据中.同时,结点为使数据分发满足自己的服务质量,使用某种调度算法决定从哪个邻居请求哪些数据.常用的调度算法有:随机调度[11]、最少优先调度[12,13]和顺序调度[14,15]等.随机调度算法以请求结点随机选择数据和服务结点为特征,易于实现,鲁棒性强,但只有在带宽充裕的条件下才能保证播放质量;最少优先调度算法以随时间推进的交换窗口为粒度进行数据调度,优先请求邻居内副本最少的数据块,它实现比较

收稿日期:2011-02-22;修回日期:2011-10-14
基金项目:国家 973 重点基础研究发展规划(No.2011CB302601);国家 863 高技术研究发展计划(No.2009AA01Z142)

简单,也能获得较高的吞吐量,但不能保证请求结点优先得到最迫切需要的数据块;顺序调度算法优先请求距使用时刻最近的数据,以达到减少迟到的数据的目的,但降低了系统内数据的多样性,不利于数据的快速分发,最终影响系统的整体服务效果.

在上述基于请求结点的数据调度算法中,请求结点只根据自身的需求申请数据,并不能保证整个系统获得较优的服务效果.本文基于服务结点进行数据调度,以整体服务效果为目标,提出时序约束下的快速分发算法 DSF(Deadline Sensitive Fast distribution).其基本思想是,在服务结点面临多个邻居结点的多个数据请求时,将调度过程分为数据选择和优选邻居选择两个阶段.数据选择阶段以选择系统最迫切需要的数据为目标,达到最小化系统内迟到数据比例的目的;优选邻居选择阶段以选择继续服务能力强的结点为目标,优化系统的持续服务能力.实验表明,与传统算法相比,DSF 在时序约束的满足、流传输质量、分发速率、负载均衡等方面均具有较好的特性.

2 紧急数据的快速分发

由于流媒体应用具有时序性约束,每个数据块的分发都有其 deadline,因此当一个服务结点同时面临多个数据请求时,它选择的要进行服务的数据一定与其对应的 deadline 相关;另外,当该结点选择了要服务的数据后,面对若干个请求结点,它所选择分发的请求结点也与其对应的继续服务的能力相关.

结点在同一时刻面临多个数据服务请求时,数据的选择策略直接影响服务的效果.如图 1 所示,结点 P_1 是数据块 C_1、C_2 的拥有者;P_2 向 P_1 申请 C_1,deadline 为 t_2;P_2、P_3、P_4、P_5 向 P_1 申请 C_2,deadline 是 t_3.如果 P_1 在 t_0 时刻分发 C_1,数据在 t_1 时刻到达 P_2,则 C_1 的传输满足 P_2 的 deadline 要求;随后,P_1 在 t_1 时刻分发 C_2 给 P_2,数据在 t_2 到达,同样也满足 P_2 的 deadline 要求.在 t_2 时刻,可分发 C_2 的结点有 P_1 和 P_2,而需要 C_2 的结点还有 P_3、P_4、P_5.因此,无论怎样传输,在 t_3 时刻,P_3、P_4、P_5 中总有一个结点得不到数据 C_2.

如果 P_1 在 t_0 时刻分发 C_2,如图 2 所示,则 C_2 于 t_1 时刻到达 P_2;在 t_1 时刻,P_1 向 P_2 分发 C_1,P_2 向 P_4 分发 C_2,数据在 t_2 时刻到达;在 t_2 时刻,P_1 向 P_3 分发 C_2,P_2 向 P_5 分发 C_2,数据在 t_3 时刻到达.这样的数据选择策略,可使得所有的数据在各结点的 deadline 之前分发到位.

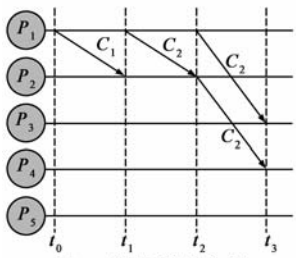


图1 数据选择实例1

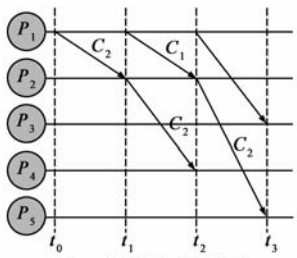


图2 数据选择实例2

为描述数据块的紧急特性,我们给出如下定义.

定义 1 紧急程度

设 t 为当前时刻,P_a 为数据块 C_k 的拥有结点,$R_k^{(t)}=\{P_1,P_2,\cdots,P_{j_k}\}$ 为 t 时刻 P_a 的邻居结点中请求 C_k 的结点集合,D_i 为 P_i 请求 C_k 的 deadline,$i=1,2,\cdots,j_k$,则 P_a 中的 C_k 在 t 时刻的紧急程度定义如下:

$$\Delta_k^{(t)}=\frac{\frac{1}{j_k}\cdot\sum_{i=1}^{j_k}(D_i-t)}{j_k}$$

$\Delta_k^{(t)}$ 的物理意义为:当结点 P_a 面临多个数据块请求时,$\Delta_k^{(t)}$ 越小的数据块 C_k,其紧急程度越高,越应该优先分发.

依据紧急程度选择待分发的数据后,当有多个邻居结点请求该数据时,优先分发给哪一个结点呢?为描述结点的服务特性,我们给出如下定义.

定义 2 分发速率

在时刻 t,若系统中拥有数据块 C_k 的结点可以在单位时间内分发 $m_k^{(t)}$ 个 C_k,则定义 $m_k^{(t)}$ 为 C_k 在时刻 t 的分发速率.

在时刻 t,对于拥有的数据集为 $A_i^{(t)}$ 的结点 P_i,若它在单位时间内能传输 O_i 个数据块,则它用于分发 $A_i^{(t)}$ 中任意数据的相对带宽可近似为 $\frac{O_i}{|A_i^{(t)}|}$.设系统中拥有数据块 C_k 的结点集合为 $H_k^{(t)}$,则 C_k 当前的分发速率 $m_k^{(t)}$ 等于拥有 C_k 的结点用于分发 C_k 的带宽之和,即:

$$m_k^{(t)}=\sum_{P_i\in H_k^{(t)}}\frac{O_i}{|A_i^{(t)}|}\tag{1}$$

定义 3 预估完成时间

在时刻 t,若系统中未拥有数据块 C_k 的结点集合为 $N_k^{(t)}$,则 C_k 在时刻 t 的预估完成时间定义如下:

$$T_k^{(t)}=\frac{|N_k^{(t)}|}{m_k^{(t)}}$$

$T_k^{(t)}$ 的物理意义为:若按照分发速率 $m_k^{(t)}$ 分发数据块 C_k 到 $N_k^{(t)}$ 中的所有结点,还需要 $\frac{|N_k^{(t)}|}{m_k^{(t)}}$ 的时间.C_k 的分发过程就是 $T_k^{(t)}$ 逐渐下降至 0 的过程.若 $T_k^{(t)}$ 能更快地下降至 0,则能更早地完成 C_k 的分发.

将式(1)代入定义 3,可得

$$T_k^{(t)}=\frac{|N_k^{(t)}|}{\sum_{P_i\in H_k^{(t)}}\frac{O_i}{|A_i^{(t)}|}}\tag{2}$$

分析 $T_k^{(t)}$,当在时刻 t' 从 $H_k^{(t)}$ 中的某个结点向 $N_k^{(t)}$ 中的某个结点 P_u 传输 C_k 时,则需要 C_k 的结点减少 1 个,拥有 C_k 的结点集合增加了 P_u,即 $|N_k^{(t')}| = |N_k^{(t)}| - 1$, $H_k^{(t')} = H_k^{(t)} \cup P_u$,亦即 $\sum_{P_i \in H_k^{(t')}} \frac{O_i}{|A_i^{(t')}|} = \sum_{P_i \in H_k^{(t)}} \frac{O_i}{|A_i^{(t')}|} + \frac{O_u}{|A_u^{(t')}|}$.由定义 3,$t'$ 时刻 C_k 的预估完成时间计算如下:

$$T_k^{(t')} = \frac{|N_k^{(t)}| - 1}{\sum_{P_i \in H_k^{(t)}} \frac{O_i}{|A_i^{(t')}|} + \frac{O_u}{|A_u^{(t')}|}} = \frac{|N_k^{(t)}| - 1}{\sum_{P_i \in H_k^{(t)}} \frac{O_i}{|A_i^{(t')}|} + \frac{O_u}{|A_u^{(t)}| + 1}} \quad (3)$$

分析 $T_k^{(t')}$,在时刻 t',若 $N_k^{(t)}$ 中接受 C_k 的是另一结点 P_v,则此刻的 C_k 预估完成时间为:

$$T_k^{(t')} = \frac{|N_k^{(t)}| - 1}{\sum_{P_i \in H_k^{(t)}} \frac{O_i}{|A_i^{(t')}|} + \frac{O_v}{|A_v^{(t')}|}} = \frac{|N_k^{(t)}| - 1}{\sum_{P_i \in H_k^{(t)}} \frac{O_i}{|A_i^{(t')}|} + \frac{O_v}{|A_v^{(t)}| + 1}} \quad (4)$$

比较式(3)和式(4),$T_k^{(t')}$ 的分子不变,分母中 $\sum_{P_i \in H_k^{(t)}} \frac{O_i}{|A_i^{(t')}|}$ 不变,只有式(3)中分母的 $\frac{O_u}{|A_u^{(t)}| + 1}$ 变为 $\frac{O_v}{|A_v^{(t)}| + 1}$.如果 $\frac{O_v}{|A_v^{(t)}| + 1} > \frac{O_u}{|A_u^{(t)}| + 1}$,则式(4)的值小于式(3)的值.亦即,当有多个结点同时请求数据块 C_k 时,将 C_k 优先传输给相对带宽最大的结点,C_k 的预估完成时间下降更快,可以更早地完成 C_k 的分发.

定义 4　优选结点

在时刻 t,若 C_k 是服务结点 P_1 中紧急程度最高的数据块,P_v 是 P_1 邻居结点中请求 C_k 服务的相对带宽最大的结点,则称 P_v 为 P_1 在该时刻的优选结点.

3　数据分发算法

数据分发由两个部分构成:数据请求与数据服务.

对请求结点而言,如果其滑动窗口的容量为 n 块数据,其中 k 块数据已下载完成或正在下载中,则需要请求的 $n-k$ 块数据以一张申请表的方式传送给邻居结点,表中的每一项包括需要请求的数据 ID 和其 deadline,如图 3 所示.

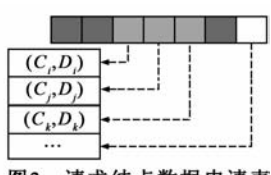


图3　请求结点数据申请表

请求端算法如算法 1 所示.

算法 1　请求端算法

```
when 滑动窗口内有未下载的数据 {
  按时间顺序找到前 k 个数据块;
  for i from 1 to k do {
      C_i = 第 i 个数据块的 ID;
      D_i = 第 i 个数据块的 deadline;
      将(C_i, D_i)加入申请表;
  }
}
when 申请表不为空{
   向邻居结点广播申请表;
   清空申请表;
}
when 有邻居结点开始向自己传输数据块 C_k' {
   向邻居结点广播自己已拥有 C_k';
}
```

服务结点维护一张数据服务表,表中记录了每一个被请求的数据的 ID、请求该数据的结点和对应的 deadline,如图 4 所示.

C_i	(P_a,D_{ia})	(P_b,D_{ib})	
C_j	(P_a,D_{ja})	(P_c,D_{jc})	(P_d,D_{jd})
C_k	(P_e,D_{ke})		
…	…	…	

图4　服务结点的数据服务表

数据服务过程由两个阶段构成:①计算数据服务表中每一个数据的紧急程度,并选择紧急程度值最小的数据进行服务.②当有多个结点请求被选择的数据块时,将该数据块发送给优选结点.

为支持算法 2 正确运行,每个结点根据其物理带宽以及运行时所传输的数据量周期性地计算其相对带宽 $\frac{O_i}{|A_i|}$,并广播给邻居结点.每个结点动态维护一个邻居结点的相对带宽表,用于分发数据时选择优先结点.

算法 2　服务端算法

```
float emergency(C_k)  {
   Δ_k = 0;
   j_k = 请求 C_k 的结点数;
   for i = 1 to j_k {
      Δ_k = Δ_k + (D_i - sysTime);
   }
   Δ_k = (Δ_k / j_k) / j_k;
   Return  Δ_k;
}
int emergencyChunk(){
   min_emergency = ∞;
   for C_i in the Table {
      temp: = emergency(C_i);
      if min_emergency > temp {
         min_emergency = temp;
         C_k = C_i;
      }
```

```
    }
    Return C_k;
}
void removeFromTable(C_k, P_i){
    将 C_k 记录中 P_i 及其对应的 deadline 删除;
    if C_k 的记录中请求结点列表为空 {
        将 C_k 的记录从数据服务表中删除;
    }
}
when 数据服务表不空 {
    C_k = emergencyChunk();
    将 C_k 对应的数据块发送给其优选结点 P_i;
    removeFromTable(C_k, P_i);
}
when 接收到邻居结点的数据申请表 {
    for 申请表中的每个数据 C_k{
        修改数据服务表;
    }
}
when 接收到邻居结点 P_i"已有数据块 C_k"的通知 {
    removeFromTable(C_k, P_i);
}
```

4 实验结果

4.1 实验配置

为评估算法性能,我们用 EPSS[16] 进行了仿真实验.仿真规模为 1000 个结点,会话持续 10 分钟.结点在仿真开始后 10 秒内按泊松分布加入系统.加入系统后,结点开始寻找邻居并进行数据交换.每个结点维护 8 个邻居结点.源结点以每秒 2 个数据块的速度发出连续的数据流,系统内结点的平均带宽为每秒发送 3 个数据块,每个数据块有自己的时戳.

本文的算法在实验中标记为 DSF.我们将 DSF 与 P2P 流媒体系统中常用的最少优先调度(RF)[7,15]、随机调度(Random)[13,14]和最近时间调度(DS)三种算法进行了比较.这三种算法的算法思想如下所述:

(1)最少优先调度算法:请求端先将要请求的数据根据拥有它的邻居结点的数量非降序排列,再对各个数据块进行调度;若有多个邻居结点可以提供同一数据块,则由能最快完成传输的邻居发送.服务端不进行调度.

(2)随机调度算法:请求端随机选择要请求的数据;如果该数据块被多个邻居结点拥有,则随机向一个邻居结点请求.服务端不进行调度.

(3)最近时间调度:选择服务数据的算法与 DSF 相同;当多个结点请求同一数据时,优先服务 deadline 最小的请求结点.该算法用于验证 DSF 优先服务优选结点的合理性.

4.2 性能比较

图 5 给出了各个算法的迟到数据块的增长情况.图中,横轴是会话时间,纵轴是平均每个结点随时间增长的迟到数据块的数量.整个会话过程传输 1200 个块,会话结束时,DSF 迟到 69 块,最少优先调度算法迟到 136 块,随机调度算法迟到 204 块,最近时间调度算法迟到 104 块.实验表明,DSF 满足实时数据流时序约束的效果最好.

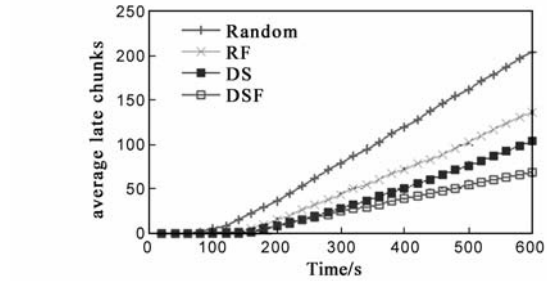


图5 迟到的数据块数

图 6 给出了各个算法的流传输质量,用及时接收的数据与应到数据的比例来衡量.横轴是会话时间,纵轴是流传输质量.很明显,在会话的每个时刻点,DSF 的流传输质量均高于其它三种算法;会话结束时,DSF 流传输质量的平均值为 94%,最少优先调度流传输质量的平均值为 89%,随机调度流传输质量的平均值为 83%,最近时间调度流传输质量的平均值为 91%并有持续下降的趋势.实验表明,与其它三种算法相比,DFS 能取得了最好的流传输质量.

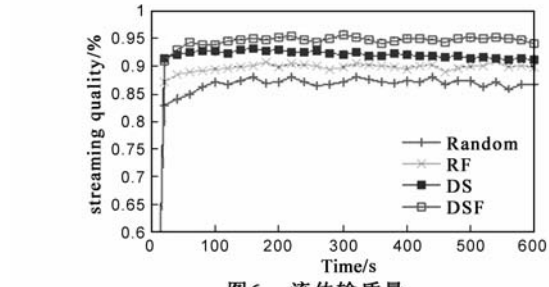


图6 流传输质量

图 7 给出了各个算法下单个数据块的数据分发速率.图中给出了 ID 号分别为 10,100,500 的数据块在各种算法下到达 90% 的结点所花费的时间.横轴是采样的数据块的 ID,纵轴是采样的数据块到达 90% 结点用掉的时间.对每一个被抽样的数据,DSF 的分发延时均小于其它三种算法.实验结果表明,对单个数据块而言,DSF 有更快的分发速率.

图 8 比较了四种算法的负载平衡特性.我们用随时间变化的负载的方差来评估各算法的负载平衡特性.方差越小,负载平衡特性越好.实验中,我们直接用结

点发送的数据块数量表示结点的负载 L,用 $\bar{L}$ 表示平均负载,则负载方差为 $\frac{\sum_{i=1}^{N}(L_i-\bar{L})^2}{N-1}$.各算法随时间增长的负载方差如图 8 所示:最近时间调度算法只以 deadline 为依据选择目标结点,总是依赖播放进度较快的结点来分发数据,造成严重的负载不均衡,有最差的负载均衡特性;最少优先调度算法的负载均衡特性比随机调度差,这是因为随机调度对服务结点的选择趋近于均匀分布,自然有较好的负载均衡特性;DSF 算法总是优先把紧急程度最高的数据分发给相对带宽最大的结点,既考虑了当前的服务请求状况,又考虑了下阶段的服务能力,因此有最好的负载均衡特性.

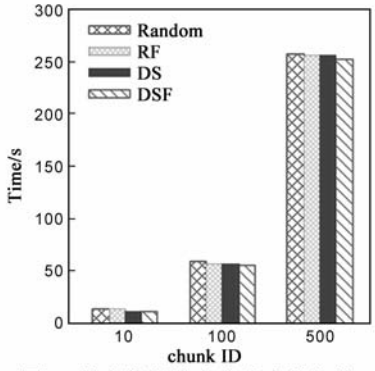


图7　单个数据块分发速度的比较

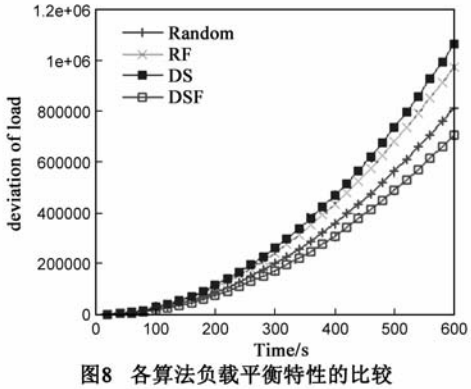


图8　各算法负载平衡特性的比较

最后比较了系统的带宽资源变化时各算法的性能.当系统内结点的平均带宽变化时,系统带宽资源的充裕程度会发生变化,系统的带宽资源越紧张,越需要充分利用结点的带宽以保证服务质量.图 9 比较了当结点的平均带宽为每秒 5 个数据块和每秒 2 个数据块时各算法随时间增长的迟到数据块数.当结点的平均带宽升为每秒 5 个数据块时,因为系统带宽资源非常充裕,各算法的迟到数据比例的差异并不明显;而当结点的平均带宽降为每秒 2 个数据块时,系统带宽资源紧张,四种算法的性能差距变得显著:随机调度和最少优先调度的迟到数据比例上升幅度尤为明显,远高于 DSF、DS.这说明在系统带宽资源紧张的情况下,更能体现 DSF 和 DS 根据紧急程度分发数据的决定在满足时序约束方面的性能优势.DSF 优于 DS 的性能说明,将选定的数据优先发送给后续服务能力强的结点比发送给时间上更迫切的结点能更有效地利用系统带宽资源,更快地完成系统范围内的数据分发.

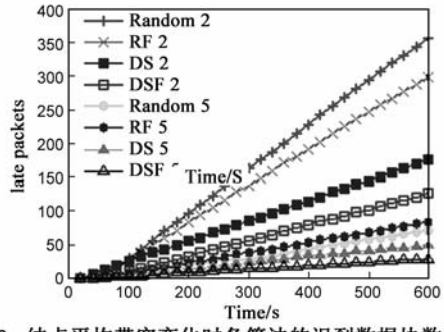


图9　结点平均带宽变化时各算法的迟到数据块数

4.3　算法复杂性分析比较

随机调度、最少优先调度、最近时间调度(DS)以及 DSF 均为基于 P2P 计算模式的调度算法.算法独立运行在系统中的每个结点中,目标是利用自己和其它结点的带宽完成数据的分发.

对请求结点来说,无论采用随机调度还是最少优先调度,其算法作用范围仅涉及滑动窗口内的数据和该请求结点的邻居结点.设数据窗口的大小为 W,邻居结点的个数为 N,则随机调度算法是在数据窗口内随机选择一个数据以及在拥有该数据的邻居结点中随机选取一个结点,根据随机计算的特点,完成一个数据块调度的计算复杂度与 W、N 无关,为常数复杂度.最少优先调度算法先将数据窗口内的数据按照副本数排序,复杂度为 $O(W\cdot\log W)$,然后按顺序将每个数据指派给能最快完成数据交付的邻居结点,指派每个数据的复杂度为 $O(N)$.由于调度一个滑动窗口内的数据只需进行一次排序,因此最小优先调度算法请求一个数据的计算复杂度为 $(O(W\cdot\log W)+O(W\cdot N))/W=O(\log W)+O(N)$.

与随机调度、最少优先调度等运行在请求结点的调度算法不同,运行在服务结点的最近时间调度(DS)和 DSF 的作用范围涉及服务队列内的数据和该服务结点的邻居结点.服务队列是各邻居结点请求队列的并集,因为各结点播放进度相近,可认为服务队列长度不超过滑动窗口的长度 W.首先选择要服务的数据,DS 与 DSF 都选择紧急程度最小的数据.计算服务队列中每个数据的紧急程度,与请求该数据的邻居数相关,而请求该数据的邻居个数不会超过邻居总数 N,因此,计算单个数据块紧急程度的复杂度不超过 $O(N)$,计算服

务队列中所有数据块的紧急程度的复杂度不超过 $O(W\cdot N)$.然后从服务队列中选择紧急程度最小的数据块,复杂度为 $O(W)$.至此,选择数据的阶段结束,该阶段的复杂度不超过 $O(W\cdot N)+O(W)$.然后,从请求该数据的邻居结点中选择目标结点,DSF 选择相对带宽最大的结点,DS 选择 deadline 最小的结点,复杂度都小于 $O(N)$.因此,DSF 和 DS 的计算复杂度都不超过 $O(W\cdot N)+O(W)+O(N)$,可近似为 $O(W\cdot N)$.与面向请求结点的随机调度和最少优先调度相比,运行在服务结点的 DSF 和 DS 具有较高的计算复杂度.

DSF 取得更好性能的代价除去较高的计算复杂度,还包括更多的通信开销,如:请求结点将自己的数据申请表向邻居结点广播后,为避免从邻居结点接收重复的数据,当从一个邻居结点收到数据 C_k 后,要立即通知其它邻居自己已有 C_k.为支持优选结点的选择,各结点也要周期性地向邻居结点广播自己的相对带宽.

5 结论

本文研究分析了动态环境中时序约束下的快速数据分发问题,得到了以下两个结论:(1)优先分发紧急程度高的数据,能有效降低数据迟到的比例,提高流传输质量;(2)将数据分发给相对带宽最大的结点,可加速数据的分发速率.在此基础上,设计了结点的服务调度算法 DSF,并通过实验证明了 DSF 在满足时序约束、快速数据分发和负载均衡等方面都有良好的性能.

参考文献

[1] BitTorrent[OL].http://www.bittorrent.com,2001.

[2] Emule[OL].http://www.emule-project.net,2002.

[3] C Dale,J Liu.Apt-p2p:A peer-to-peer distribution system for software package releases and updates[A].IEEE INFOCOM'09[C].Rio de Janeiro:IEEE,2009.864 – 872.

[4] F Wang,Y Xiong,J Liu.mTreebone:A hybrid tree/mesh overlay for application-layer live video multicast[A].ICDCS'07:Proceedings of the 27th International Conference on Distributed Computing Systems[C].Toronto:IEEE Computer Society,2007.49 – 56.

[5] D Ren,Y-T Hillman Li ,S-H.Gary Chan.On reducing mesh delay for peer-to-peer live streaming[A].Proc of IEEE INFOCOM'2008[C].Phoenix,Arizona:IEEE Computer Society,2008.1058 – 1066.

[6] Y Huang,T Z J Fu,D-M Chiu,J C S Lui C Huang.Challenges,design and analysis of a large-scale p2p-vod system[A].ACM SIGCOMM'08[C].Seattle:ACM,2008.375 – 388.

[7] V Venkataraman,K Yoshida,P Francis.Chunkyspread:heterogeneous unstructured end system multicast[A].Proceedings of IEEE ICNP'06[C].Santa Barbara,CA:IEEE,2006.2 – 11.

[8] 杨戈,廖建新,朱晓民,樊秀梅.流媒体分发系统关键技术综述[J].电子学报,2009,37(1):137 – 145.
Yang Ge,Liao Jian-min,Zhu Xiao-min,Fan Xiu-mei.Survey of key technologies of the distribution system for streaming media[J].Acta Electronica Sinica,2009,37(1):137 – 145.(in Chinese)

[9] N Magharei,R Rejaie ,Y Guo.Mesh or multiple-tree:A comparative study of live p2p streaming approaches[A].INFOCOM'07[C].Alaska,USA:IEEE,2007.1424 – 1432.

[10] Chen Feng,Baochun Li,Bo Li.Understanding the performance gap between pull-based mesh streaming protocols and fundamental limits[A].INFOCOM'09[C].Rio de Janeiro:IEEE,2009.891 – 899.

[11] C Liang,Y Guo,Y Liu.Is random scheduling sufficient in p2p video streaming? [A].Proceedings of the 2008 The 28th International Conference on Distributed Computing Systems[C].Washington,DC,USA:IEEE Computer Society,2008.5360.

[12] A Vlavianos,M Iliofotou,M Faloutsos.Bitos:enhancing bittorrent for supporting streaming applications[A].INFOCOM'06[C].Barcelona:IEEE,2006.1 – 6.

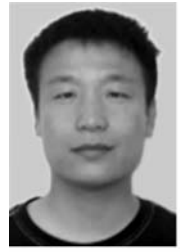
[13] Y Zhou,D-M.Chiu,J C S Lui.A simple model for analyzing p2p streaming protocols[A].Proc of IEEE ICNP'07[C].Beijing:IEEE,2007.226235.

[14] D Xu,M Hefeeda,S Hambrusch ,B Bhargava.On peer-to-peer media streaming[A].ICDCS'02[C].Vienna,Austria:IEEE,2002.363 – 371.

[15] V Agarwal ,R Rejaie.Adaptive multi-source streaming in heterogenous peer-to-peer networks[A].MultiMedia Computing and Networking 2005 (MMCN)[C].San Jose,CA,USA:SPIE/ACM,2005.13 – 25.

[16] Python-based p2p streaming simulator[OL].http://code.google.com/p/pyp2pstrmsim.

作者简介



吴吉庆 男,1980 年生,博士研究生,主要研究方向为 P2P 流媒体和 P2P 计算 .
E-mail:jiqingwu@gmail.com

彭宇行(通信作者) 男,1963 年生,研究员,博士生导师,主要研究方向为分布式计算.
E-mail:pengyuxing1963@yahoo.com.cn