等价多路径间基于 LRU Cache 和计数统计的流量分配调度算法

林 伟[1,2], 刘 斌[3], 唐 毅[3]
(1. 清华大学深圳研究生院, 广东深圳 518055; 2 香港城市大学, 香港; 3. 清华大学计算机科学与技术系, 北京 100084)

摘 要: 为了减少网络拥塞并充分利用链路带宽, 当在转发节点与目的子网间存在有多条等价路径(ECMPs)时, 流量负载应该在 ECMPs 间均衡分配, 并且属于同一个 TCP 流的 IP 分组应该按照相同顺序到达目的主机. 本文提出了一种基于 LRU(Least Recently Used Algorithm) Cache 和计数统计的算法. 该算法通过为每条 ECMP 分配一个计数器, 利用计数统计从而考虑到了 IP 分组的长度差异. 使用相对计数以及对某些情况增加约束条件解决了计数器溢出问题. UDP 分组只需要作为调节负载均衡的流量. 更进一步, 对于去往同一目的子网的不同主机的 TCP 流的时延差异被转化为 cache 中的表项失效的时间长度差. 仿真实验表明, 当 ECMPs 间的时延差不显著的情况下, 只需要很小的存储空间, 且每次 cache 查找只需要一个时钟周期, 负载均衡接近最优, 此时只有 2% 的分组出现乱序.

关键词: 流量分配; 等价多路径; LRU; Cache; 计数统计
中图分类号: TP393.05 文献标识码: A 文章编号: 0372-2112 (2008) 01-0032-07

Achieving Optimized Traffic Sharing over Equal-Cost-Multi-Paths Using LRU-based Caching with Counting Scheme

LIN Wei[1,2], LIU Bin[3], TANG Yi[3]
(1. *Graduate School at Shenzhen, Tsinghua University, Shenzhen, Guangdong* 518055, *China*; 2. *City University of Hong Kong, Hong Kong, China*; 3. *Department of Computer Science and Technology, Tsinghua University, Beijing* 100084, *China*)

Abstract: In order to reduce network congestion and fully use link bandwidth, when there are Equal-cost-multi-paths (ECMPs) between a forwarding node and a destination subnet, traffic load should be balanced among ECMPs and packets of the same TCP flow should reach destination host in the same order. An algorithm called LRU-based caching with counting (LCC) is proposed. Packet length differentiation is considered to achieve load balance by adapting a counter for each ECMP, and counter overflow is solved by relative counting and restrictions. UDP packets only need to be concerned to achieve load balance. Furthermore, flow delay differentiation forwarding to different hosts of the same destination subnet is transformed to entries in cache invalided time period difference. Simulation shows that when delay differentiation among ECMPs is not significant, storage requirement is small, only one cycle is needed for each cache lookup, load balance is near optimal, and only 2% of packets are out of order.

Key words: traffic splitting; ECMPs; LRU; cache; counting

1 引言

随着 Internet 网络规模的的迅速扩大, 其拓扑结构日趋复杂. 从流量工程、可靠性和网络安全等因素考虑, 网络中的两个节点间部署了多条路径. 目前, 绝大部分单一路径路由协议都是基于 dijkstra 的最短路径算法, 只有开销最小的路径被选为路由路径. 按照这种算法, 网络所提供的带宽没有被充分利用, 有可能出现多条路径中的一条路径负载过重, 而其他路径却处于空闲状态, 如此将造成网络的局部拥塞. 当 IP 分组经过多个节点之后, 负载不均衡的情况可能会进一步加剧. 如果存在多于一条的路径的开销近似相等, 单一路径路由协议的计算结果会在这几条路径间变化, 造成频繁的路由表更新, 网络变得不稳定, 性能也受到影响[1].

收稿日期: 2005-12-30; 修回日期: 2007-12-20
基金项目: 国家自然科学基金 (No. 60373007, 60573121); 中国-爱尔兰科学技术合作研究基金 (No. CI-2003-02); 高等学校博士点基金 (No. 20040003048); 清华大学 985 基金 (No. JCpy2005054)

与单一路径路由协议不同, 等价多路径路由协议被提出用以解决上述的问题[2,3]. 对于那些去往同一目的子网的 IP 分组, ECMPs 被提供作为下一跳路由选择的候选路径, 此处等价是指: 从转发节点到目的子网间的这些路径的最短路径开销(例如延时或者经过的跳数) 计算结果, 在一定的精度下近似相等.

当路由协议提供 ECMPs 时, 为了充分利用可用带宽并降低路由的不稳定性, 流量应该在 ECMPs 间均衡分配. 在本文中, 为了便于讨论, 假设每条路径的带宽都相等; 不相等的情况下, 给每条路径附加一个权重参数来表示带宽的差异, 流量应该根据权重参数在 ECMPs 间进行分配.

对于TCP 分组, 当属于同一个 TCP 连接流* 的分组从不同路径转发时, 就可能会出现乱序的问题. 这是因为每条路径的时延并非精确相等, 其时延可能会随着流量负载压力的变化而变化, 因而有可能出现后转发的分组经过另一更短时延路径先于先转发的分组到达目的主机的情况. 如果连续到达的乱序分组数小于 TCP 滑动窗口长度, TCP 滑动窗口可以处理乱序的问题, 不会造成网络性能的下降; 否则, 目的主机会发出重传请求, 网络以及相关应用程序的性能会受到明显影响.

因此, 负载均衡问题以及分组保序问题应该被一同优化. 如果分组按照轮转的方式转发, 而不考虑分组长度的差异, 负载均衡将达到最优, 但是分组的乱序很严重. 另一个极端, 如果属于同一个流的分组都必须从同一个路径转发, 虽然分组可以被严格保序, 但是负载均衡不能达到最优, 其性能取决于流量分布. 当分组经过若干跳之后, 负载分配差异会愈加明显, 这在 Internet 上是很常见的情况.

本文首先提出了基于 LRU Cache 和计数统计的算法(LCC) , 指出不需要严格约束属于同一个流的分组都从同一条路径转发. 其次研究了由于流重定向造成的乱序的概率以及同一个流相邻两个分组的时间间隔之间的关系. 在负载平衡时考虑到了分组长度的差别. 由于UDP 分组不需要保序, 因此只需要将其在 ECMPs 间分配以满足负载均衡.

本文结构如下, 第 2 部分介绍了在 ECMPs 间流量分配的相关工作; 第 3 部分阐述 LCC 算法的基本思想和实现结构; 第 4 部分给出了 LCC 算法和其他算法的仿真实验结果, 并在负载均衡以及分组保序两方面进行了性能比较, 对分组乱序进行了初步的理论分析; 最后, 第 5 部分对全文作了总结.

2　相关工作调研

在 ECMPs 间分配流量的原则是:

• 均衡各路径的负载;

• 保证属于同一个流的 TCP 分组按照顺序到达目的主机;

这两个原则相互冲突, 需要进行相应的权衡折中以满足上述目标. 如下一些算法在相关文献中被提出来解决该问题.

2.1　逐个分组轮转算法(PBP)

假设分组长度固定, PBP[4] 可以达到最优的负载均衡, 但是分组乱序很严重. 如果分组的长度不固定, PBP 既不能达到负载均衡也不能使分组保序.

2.2　贪婪算法(Greedy Method)

当一个新的分组到达且等待转发时, 每条路径的负载计数器值被比较, 分组被分配给目前负载最轻的路径. 此方法[5] 考虑到了每个分组的长度的差别, 但是分组乱序仍然存在, 此外还存在计数器溢出的问题.

2.3　直接哈希算法(DH)

使用五元组(SIP, DIP, SP, DP, Protocol Type) 的组合作为哈希函数的输入. 哈希值被定义为流号, 流号作为区分不同流的标志. 如果 ECMPs 的数目为 k, 每条路径被赋以一个取值范围为 0 到 $k-1$ 的整数编号. 然后流号对 k 取模, 余数直接映射到具有相同编号的路径上. DH 算法[6] 如图 1 所示. 同一个流的分组严格保序, 但是流量负载没有很好地被均衡.

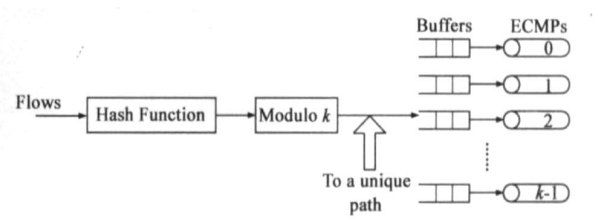

图 1　直接哈希算法示意图

2.4　基于表的哈希算法(TH)

TH 算法[6] 相对于 DH 算法的改进之处在于流与 ECMPs 之间的对应关系可以预先人为配置. 尽管改进了负载均衡并且没有分组乱序, 但是预先配置的流量分配是静态的, 而分布变化的流量负载不能很好地在 ECMPs 间得到均衡. TH 算法如图 2 所示, 在哈希函数输出端与备选路径间增加了一级映射表, 用于配置流与备选路径的对应关系.

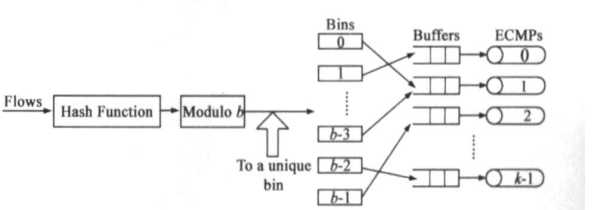

图 2　基于表的哈希算法示意图

* 同一个流的分组是指其源 IP 地址(SIP) 、目的 IP 地址(DIP) , 源端口号(SP) , 目的端口号(DP) 和协议类型都相等.

2.5 动态重新配置的基于表的哈希算法(TRH)

考虑到TCP滑动窗口的存在,轻度的乱序分组可以被处理而不会影响性能. 动态重新配置被引入用以进一步均衡ECMPs间的流量负载. 每隔一个固定时间片,当前时间片内负载最重的路径上的某一个流将会被重定向到当前时间片内负载最轻的路径上[7]. 为了避免计数器溢出,在本时间片开始时,上个时间片的流量负载统计信息会因清零而丢失. 时间片的长度以及被重新定向的流需要精心选择,以在均衡负载和分组保序间折中. THR算法如图3所示,映射表额外记录流经该路径的流量大小信息,每个周期结束后根据此信息更新流与备选路径对应关系,从而使负载动态均衡.

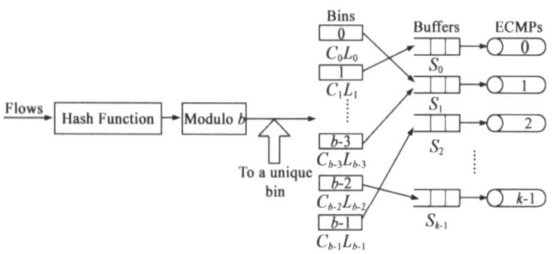

图3 动态重新配置的基于表的哈希算法示意图

2.6 快速交换算法(FS)

使用cache处理在ECMPs间流量分配是由CISCO首先提出的[8]. FS没有考虑分组长度的差别. 当流不能在cache中被找到时,FS使用轮转法选择一条路径,这种方法不能提供均衡的负载. TCP和UDP协议被处理的需求差异也没有被考虑.

为了解决上述算法存在的问题,本文提出了基于LRU Cache和计数统计的流量分配调度算法(LCC),用以在存在小概率的分组乱序的情况下进一步改进负载均衡,小概率的分组乱序可以被TCP滑动窗口处理而不会影响性能. LCC基于按照LRU替换更新算法维护的cache;考虑到了分组长度的差别,通过增加计数统计的功能,为每条路径增加一个按照字节计数的计数器加以实现;通过使用相对计数和特定约束条件,避免了计数器溢出的问题;TCP分组与UDP分组由于需求不同而被区别对待;更进一步,到达同一目的子网的不同主机的流的时延差别被转化为cache中的流表项的生存时间的差别,cache的存储空间可以得到更有效的利用. 负载均衡被进一步改进,同时仍然保持分组乱序的概率很低.

在下一节中,LCC算法将会被更详细的阐述.

3 LCC调度算法与结构

LCC是基于cache的流量分配调度算法,利用了如下通过观察得到的结论,如果同一个流的相邻两个分组的转发间隔时间足够长的话,例如间隔时间超过不同路径的时延差异的最大值,后一个分组可以从ECMPs中的任意的一条路径转发,而不会造成分组乱序. 因为等价多路径间的时延差异很小,所以只需要很短的间隔时间就可以避免分组乱序.

回想起先前所述的TCP滑动窗口在乱序的分组数不超过窗口长度的情况下可以处理分组乱序问题,因此,属于同一个流的分组可以从不同的路径转发,只存在很小的分组乱序概率. 这与使用LRU更新替换算法的cache正相似,只有最近的转发信息存储在cache中,同时cache中还存有流与路径的对应关系. 因为需要记录的时间间隔很短,只需要使用很小的cache.

分组长度是有差别的,以太网帧最长为1518字节,最短为64字节. 因此在需要在ECMPs间均衡负载时,应该考虑分组长度的差异. PBP算法与FS算法只使用轮转法分配调度分组,没有考虑长度的差别,因此这两种算法不能总取得很好的负载均衡结果.

LCC通过为每条ECMP设置一个计数器,用以监视流量负载. 当属于某个没有在cache中被记录的流的分组到达,需要被转发时,这表明该流可能是一个原来没有到达过的新的流,或者是属于该流的前一个分组与当前的分组间隔时间很长,以至于在cache中的流记录被最近到达的其它流替换掉了. 本文将这两种情况的流都称为新流. LCC将会根据目前计数器的统计信息,选择负载最轻的路径作为该流的转发路径,并且按照LRU替换算法更新cache,同时更新相应的计数器.

分组的长度将会累加到所选路径对应的计数器上,然而,这有可能会造成计数器溢出. 相对计数被用来解决此问题. 因为当为一个新流选择一条负载最轻的ECMP路径时,只有计数器间的相对差值而不是计数器的绝对值才有意义,每次所有的计数器都被更新:计数器的新值等于旧值减去当前所有计数器中的最小值. 然而属于同一个流的分组连续到达仍然会造成计数器溢出,因此增加约束条件:如果计数器的旧值加上分组长度超出计数器的表示范围而溢出,那么就把新值设为计数器所能表示的最大值. 因为ECMPs的数目通常都不多于4条[9],所以更新计数器的复杂度很低.

因为UDP分组不需要保序,所以直接将其分配给目前负载最轻的ECMP. 当UDP分组占总流量的比例达到10%以上并且流的数目远多于cache中的记录表项数时,负载均衡可以达到近似最优.

为了更有效的利用cache的存储空间,LCC考虑到了从转发节点到目的主机的流的时延差异. 如图4所示,尽管从转发节点到目的子网的入口节点的时延开销是等价的,从目的子网的入口节点到目的主机的时延差异可能很显著. LCC将时延的差异转化为在cache中记录的流的有效生存期的差异. 最初,所有记录的流

在 cache 中的生存期都相等. 具有较短的 cache 生存期的流记录可以在 cache 满之前就被替换掉. 在 cache 大小不变的情况下, 流量负载均衡可以进一步被改善, 而分组乱序的概率依然很低. 一种可能的探测流的时延的方案是使用 ICMP 分组(例如“ ping”) 来探测从转发节点到目的主机的时延差异.

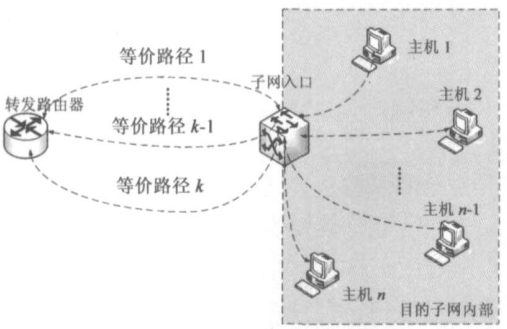

图 4　网络中等价多路径拓扑示意图

多下一跳处理模块在查找引擎中的位置如图 5 所示. 到达的分组首先被解析模块处理; 解析模块提取 DIP 字段送给路由查找模块, 提取一些字段(DIP、SIP、DP、SP、Protocol、Length) 送给 ECMPs 选择模块. IP 分组在分组缓冲区中缓存, 等待路由查找结果并将其附加在分组头部. 接下来, 如果路由查找模块的返回结果指示

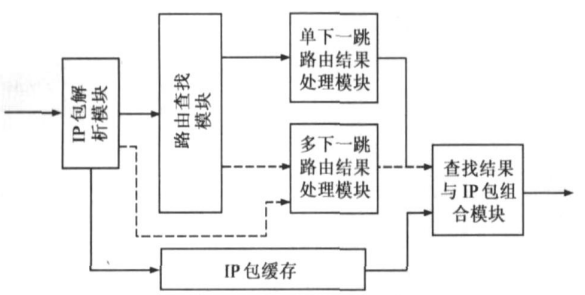

图 5　多下一跳处理模块在查找引擎中的位置

只有单一的下一跳, 单下一跳处理模块就负责给出下一跳 IP 地址以及对应的端口号查找结果. 而多下一跳处理模块直接忽略该分组所对应的输入字段信息; 否则, 如果路由查找结果指示存在多个等价路径供选择的话, 相应的多下一跳处理逻辑就负责使用 LCC 算法从中选取一个下一跳的 IP 地址和对应的端口号. 最后,

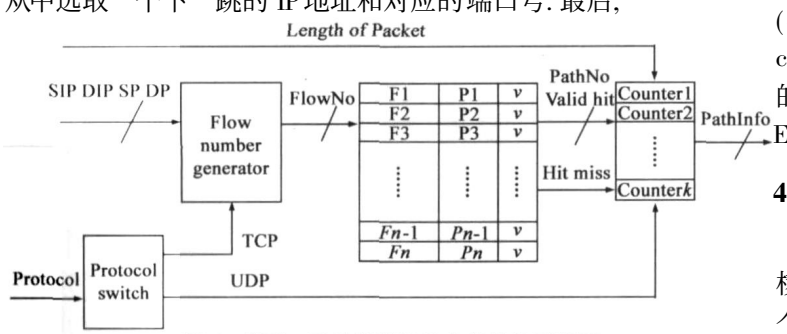

图 6　多下一跳处理模块的内部结构示意图

组合模块将路由查找结果附加在分组的前面, 输出用以转发.

从转发节点到某个目的子网间的多下一跳处理模块的内部结构如图 6 所示. 该模块采用流水线处理方式. 对于那些需要由多下一跳处理模块处理的分组, 其协议字段首先被用来判断该分组是 TCP 分组还是 UDP 分组. TCP 分组和 UDP 分组的处理方式是不同的, 因为 UDP 分组不需要保序.

对于 TCP 分组, 通过使用 SIP、DIP、SP、DP 作为输入流号产生器的输入, 产生一个流号. 流号产生器可以使用 CRC32 或者其它的哈希算法产生流号(FlowNo) .

为了能使查找 cache 在一个周期内完成, 并行比较被引入用以保证查找性能. 当 cache 长度很小时, 可以使用寄存器或者片上的 B-CAM 实现并行比较.

通过将流号与 cache 中所有 V 字段为 1(表明该纪录有效) 的记录的 F 字段(F 表示流号) 相比较, 得到一个结果位图. 对该位图的所有位进行或操作, 如果返回“ 1” 表明 cache 查找命中(对应的流在 cache 中) , 否则, 表明 cache 查找不命中(对应的流不在 cache 中) .

当“ cache 命中” 时, “ 命中” 记录的地址可以从结果位图的第一个 1 所在的位置计算得到. 相应的分组将根据“ 命中” 记录中的 P 字段(记录路径编号) 所示的路径编号从相应的路径转发, cache 使用 LRU 算法进行更新(记录的索引信息可以按照链表结构进行组织, 当 cache 更新时只需要移动一个记录的索引) , 计数器使用相对计数的方法进行更新.

当“ cache 不命中” 时, 新的流将从当前计数器值最小的路径转发. 流号与路径编号的映射关系使用 LRU 替换算法保存在 cache 中, 如果 cache 在更新之前已满, 最近最久没有被引用的记录将会从 cache 中被替换掉.

对于 UDP 分组, 由于其不需要保序, 只需要根据当前计数器的最小值选择路径. 所有计数器按照相对技术的方法进行更新. 通过 UDP 分组的辅助, LCC 可以比其它算法达到更优的 ECMPs 间负载均衡的能力.

因为 ECMPs 间的时延差异很小, 只需要带有很少的记录数的 cache 就可以满足低分组乱序概率的要求. (在后面的实验中, 当使用具有 16 条记录的 cache 时, 每 1 百万个分组中只有 2% 乱序) . 小的存储空间要求可以允许片上支持更多的 ECMPs 处理模块存在.

4　仿真实验与性能理论分析

根据统计结果显示, 2003 年在 CERNET 的核心路由器上同时并发的流的数量接近 3 亿个[10]. 表 1 表明 Internet 的绝大多数流量都是 TCP 分组和 UDP 分组[11]. 尽管 TCP 分组占了绝

大多数, UDP 分组所占比例也不容忽视. 在 LCC 算法中, 考虑到UDP 分组不需要保序, 因此用 UDP 分组平衡 ECMPs 间的负载差异. 而其它的算法没有对 UDP 分组进行特别对待和处理, UDP 分组和 TCP 分组按照同样的方式进行处理. 表2 显示骨干网中的不同应用程序所占的比例[11].

表 1 骨干网中不同协议所占的比例

协议	字节	分组	流
TCP	95%	90%	80%
UDP	5%	10%	20%
其它	ICMP 占绝大多数		

表 2 骨干网中不同应用程序所占的比例

应用程序	字节	分组	流
Web	70%	75%	75%
DNS	1%	3%	18%
SMTP	5%	5%	2%
FTP	5%	3%	1%
NNTP	2%	1%	1%
Telnet	1%	1%	1%
Others	web 相关应用程序		

本仿真实验基于 *NS*-2[12]. 仿真中的流的数目为 10000 个. 仿真时间为30 秒. 流量的分布如表 1、表2 所示.

仿真实验所采用的网络拓扑结构如图7 所示. 节点 *B* 代表子网 *A* 的边界路由器, 负责将分组转发到子网 *D*. 节点 *C* 是子网 *D* 的边界路由器. 节点 *B* 与节点 *C* 之间有 8 条等价路径 ECMPs, 连线上面的数字代表链路传输时延, 以毫秒为单位.

基于以上仿真环境, LCC 算法与其它算法(例如 PBP、DH、TH、THR、FS) 的性能加以比较.

4.1 负载均衡性能分析

对于图 7, 我们关注去往节点 *C* 的流, 可以将这些流分配到 8 条 ECMPs 上. 假设 8 条 ECMPs 上的理想的流量比是 1∶1∶1∶1∶1∶1∶1∶1. 不同算法的负载平衡性能如表 3 所示.

表 3 各算法负载平衡的比较

%	LCC	PBP	DH	TH	THR	FS
Port 1	12.49	10.37	12.36	12.39	11.57	10.43
Port 2	12.51	12.07	11.73	11.68	12.38	12.21
Port 3	12.51	11.77	18.04	13.47	11.62	14.45
Port 4	12.49	12.32	5.86	12.35	12.40	12.42
Port 5	12.52	11.63	15.50	10.67	13.36	12.17
Port 6	12.50	14.68	13.42	17.49	13.56	13.23
Port 7	12.49	13.50	6.99	13.15	12.53	11.97
Port 8	12.49	13.62	16.10	8.79	12.54	13.08
方差	0.0001	1.8644	18.365	6.2976	0.5044	1.3481

根据表 3 所示, LCC, FS, PBP 算法都基本能做到负载平衡, 其中LCC 方法的负载平衡效果最好, 可以达到 10^{-4}, 因为它可以做到实时的调节每条路径的流量. FS

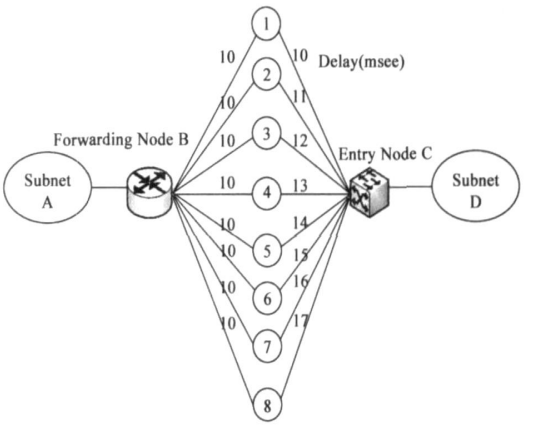

图 7 仿真实验网络拓扑图

和 PBP 两个算法是基于轮转方式对端口流量进行改变. 不同的是 FS 只有在 cache 不命中时才采用轮转法调度. DH 和 TH 两个算法的负载平衡情况比较差. THR 算法通过选择合适的参数也可以获得较好的负载平衡, 但是负载平衡调节缺乏实时性, 在较短的时间段内各路径的流量偏差较大, 并且在每一周期过后由于计数器全部清零, 导致负载平衡调节缺乏连贯性.

根据图 8 所示, 我们对这几个算法的负载平衡的实时性加以比较. *X* 坐标轴代表分组数目, *Y* 坐标轴代表等价路径间的最大负载差异. LCC 达到最好的负载均衡性能, 并且等价路径间流量抖动很小.

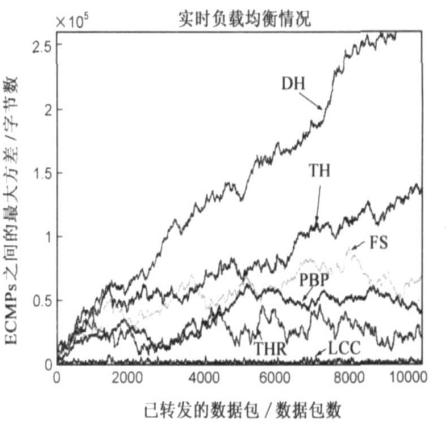

图 8 等价路径间的实时流量差异

4.2 分组保序性能分析

本文的乱序统计, 只统计同一个流之间的乱序情况. 对于同一个流内的乱序统计, 我们规定, 如果流内第 i 个发出的包, 不是在流内第 i 个被接受, 我们就可以认定他为乱序. 最后我们通过统计乱序包数在总共发出的包数中的比例来衡量每种算法的乱序的情况.

与乱序测试主要相关的参数有: 链路带宽(α), 各条等价路径的延迟时间($delay[i]$), cache 中表项最短替换时间(c). 假设当前数据包大小(γ_k), ECMPs 的数目(p), 当前流相邻两分组间隔时间($k+1$), 则如果发生

乱序, 要满足条件不等式组(1).

$$\exists i,j(i,j \in N, 0 \leqslant i \leqslant p, 0 \leqslant j \leqslant p)\begin{cases} |delay[i]-delay[j]| > \sum(\gamma_k/\alpha + t_k) \\ \sum(\gamma_k/\alpha + t_k) > c \end{cases} \quad (1)$$

满足不等式组(1)的 i,j 越多, 则发生乱序的可能性越大. 从不等式组(1)中我们可以看出, 当 $\sum(\gamma_k/\alpha + t_k)$ 较小时, 乱序发生的概率较低, 所以降低网络乱序比例的办法是使 $\sum(\gamma_k/\alpha + t_k)$ 尽可能小. 通过减小数据包大小(γ_k), 增大链路带宽(α), 增加当前流相邻两数据包间隔时间(t_k)三种办法来实现, 但这 3 种办法是很难实现的. 减小 p 也可以减少乱序发生的概率, 但是这就没有充分利用网络中可用的带宽, 违背了我们设计的初衷. 因此只可以通过调节 cache 中表项最短替换时间 c 来实现以上对乱序发生概率的调节, 即通过调节 cache 的大小来调节乱序发生的概率.

若 MAX($delay[i]-delay[j]$) 比较小, 也就是各路径的时延差异较小, 则可以使用较小的 cache 满足较低的乱序发生概率.

仿真系统中, 每条输出路径的带宽参数设为 44.736Mbps(DS3), cache 大小为 16, 等价输出路径 8 条, 乱序比率如表 4 所示.

表 4　各算法的乱序分组占总分组数的比率

算法	LCC	PBP	DH	TH	THR	FS
乱序比率	2.36%	32.83%	0	0	2.59%	2.56%

如表 4 所示, 乱序情况最多的是 PBP 算法, 因为它的算法是相邻的包都发送到不同的端口, 所以相当于 c 较小的情况, 发生乱序可能性大. DH 和 TH 这两种算法, 由于他们属于同一个流的数据包都发往同一个输出端口, 所以这两种算法是不存在乱序. FS 和基本的 LCC 算法对于乱序都采用相同保序策略, 所以他们的乱序比例都相当, 且都低于 3%. 但如果引入时延差异导致表项过期的时间不同, 则 LCC 可以更有效地利用有限的 cache 资源, 进一步降低乱序的概率. THR 算法在路径切换的过程中也会有乱序的可能.

5　总结

为了解决等价多路径间的负载均衡与分组保序的问题, 很多算法被提出. 然而这些算法不能同时很好的解决以上两个问题. 本文指出如果属于同一个流的连续两个分组的转发间隔时间足够长, 后一个分组可以从不同的路径转发而乱序的概率很低. 相应的基于 cache 的 LCC 算法被提出, LCC 采用 LRU 替换算法更新 cache, 考虑到了分组长度的差异以及 TCP 与 UDP 分组对保序的不同需求, 使用相对计数的方法避免计数器溢出, 利用从转发节点到属于同一子网的不同目的主机的时延差别进一步提高 cache 的空间利用率.

更进一步, 为了达到高速的吞吐率, 本文提出了一种采用流水线的并行硬件处理结构. Cache 查找可以在一个时钟周期内完成. 只需要使用很小的 cache, 可以采用片上资源实现.

最后, 仿真结果表明 LCC 算法的相对于其他算法有明显的优点. LCC 可以实时的接近最优的均衡等价多路径间的负载流量, 同时仅仅有很小概率的分组乱序出现.

参考文献:

[1] Andersen, D G, A C Snoeren, H Balakrishnan. Best-path vs. multi-path overlay routing[A]. In Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC, pp. 91-100, 2003[C]. Miami Beach, FL, United States: Association for Computing Machinery, New York, NY 10036-5701, United States.

[2] C Hopps. RFC 2992: Analysis of an Equal-Cost Multi-Path Algorithm[OL]. www.ietf.org/rfc/rfc2992 November 2000.

[3] D Thaler. RFC 2991: Multipath Issues in Unicast and Multicast Next-Hop Selection [OL]. www.ietf.org/rfc/rfc2991, November 2000.

[4] E L Hahne. Round-robin scheduling for max-min fairness in data networks[J]. IEEE Journal on Selected Areas in Communications, 1991, 9(7): 1024-1039.

[5] Shi H, H Sethu. Greedy fair queueing: A goal-oriented strategy for fair real-time packet scheduling[A]. in Proceedings of Real-Time Systems Symposium[C]. Cancun, Mexico: Institute of Electrical and Electronics Engineers Inc, 2003. 345-356.

[6] Cao Z, Z Wang, E Zegura. Performance of hashing-based schemes for Internet load balancing[A]. in Proceedings of IEEE INFOCOM[C]. Tel Aviv, Isr: IEEE, Piscataway, NJ, USA, 2000. 1. 332-341.

[7] Chim T W, K L Yeung. Traffic distribution over equal-cost multi-paths[A]. in Proceedings of IEEE International Conference on Communications[C]. Paris, France: Institute of Electrical and Electronics Engineers Inc., Piscataway, United States:, 2004. 2. 1207-1211.

[8] A Zinin. Cisco IP Routing, Packet Forwarding and Intra-domain Routing Protocals[M]. Section 5.5.1: Addison Wesley, 2002.

[9] Z-Y Liang, K Xu, J-P Wu, M-W Xu. IP lookup scheme supporting routing compaction and multi next hops[J]. Ruan Jian Xue Bao/Journal of Software, 2004, 15(4): 550-560.

[10] G Cheng, J Gong, W Ding, J-L Xu. Hash algorithm for IP flow measurement[J]. Ruan Jian Xue Bao/Journal of Software, 2005, 16(5): 652-658.

[11] K Thompson, G J Miller, R Wilder. Wide-area internet traffic patterns and characteristics[J]. IEEE Network, 1997, 11(6): 10-23.

[12] http://www.isi.edu/nsnam/ns.[OL]

作者简介:

林 伟 男,1981 年 8 月生于辽宁沈阳,博士研究生,2004 年毕业于大连理工大学计算机科学与技术系,同年进入清华大学计算机科学与技术系计算机网络技术研究所直读博士.研究方向为核心路由器体系结构、路由查找、流分类查找、以及 IP 报文深度处理.

E-mail: linrw04@mails.tsinghua.edu.cn

刘 斌 男,1964 年 7 月生于山东临朐,清华大学计算机科学与技术系教授,博士生导师.1993 年毕业于西北工业大学计算机应用专业,获工学博士学位,1993 年 5 月至 1995 年 11 月在北京邮电大学通信与电子系统专业从事博士后研究工作,1995 年至今任教于清华大学.科研方向为现代交换技术理论与方法学、QoS 控制与流量工程、网络处理器及高速网络中的信息安全.

唐 毅 男,1983 年 3 月生于湖北武汉,博士研究生,2005 年毕业于西北工业大学计算机科学与技术系,同年进入清华大学计算机科学与技术系计算机网络技术研究所直读博士.研究方向为核心路由器体系结构、流分类技术、网络安全技术等.