一种采用流切割实现报文保序的负载均衡算法

卜佑军 王 超 汪斌强
(国家数字交换系统工程技术研究中心 郑州 450002)

摘 要 网络链路过载或链路失效时,使用负载均衡技术可以避免网络发生拥塞。负载的分派粒度决定了负载均衡系统的均衡性能。分派粒度越细,均衡效果越理想。基于包水平粒度的负载分派可以实现理想的均衡性能,但是会造成同一TCP业务流中报文乱序;基于流水平的分派可以保证报文不乱序,但均衡效果不理想。提出了按照报文段粒度分派负载的FSLB算法。仿真实验表明,该算法可避免报文乱序并能达到较理想的均衡效果。

关键词 负载均衡,报文段,流切割
中图法分类号 TP393 **文献标识码** A

Load Balancing Algorithm Using Flow Splitting to Avoid Packet Reordering

BU You-jun WANG Chao WANG Bin-qiang
(National Digital Switching System Engineering & Technology Researching Center, Zhengzhou 450002, China)

Abstract Load balancing is a effective technique that keeps network from congestion when link over loading or link failures. The performance of balancing system was determined by granularity of load splitting. The more fine granularity the more better performance. But splitting schemes must make a tradeoff between slicing granularity and packet reordering. Splitting traffic at the granularity of packet, each path of the balancing system can obtain accurately load assignment, but can make a lot of reordering packets in the same TCP flow. Splitting traffic at the granularity of flow, the packet of the same flow arrives at the destination along the same path, which will not cause packet reordering, but each path gets the inaccurately load assignment compared with it's desired load sharing. This paper showed that one could split a flow into multi path without causing packet reordering, which using the time slot between consecutive packet of the same flow to chop the flow into several segment. FSLB algorithm splited the traffic at the granularity of the segment, the results show that FSLB algorithm gets the fine performance by simulation.

Keywords Load balancing, Segment, Flow chopping

1 前言

基于最短路径算法的IGP是造成自治系统(AS)发生拥塞的主要原因。最短路径算法是简单的基于度量值的优化算法,都是拓扑驱动的,不考虑网络可用的带宽和流量的特征。当多个业务流的最短路径汇聚到一条特定的链路或路由器接口上,或者某条业务流的最短路径通过某条带宽不足以支持该业务的链路或接口时,就会发生拥塞。在这两种情况下,即使存在拥有充足带宽的其他路径,基于最短路径算法的路由协议都不会选择这些路径来分担负载。

当网络中间节点到目的节点拥有多条可达路径时,负载均衡系统成了避免网络拥塞的关键部件。负载均衡系统需要把汇聚到节点的数据量在多条路径中进行分派。分派粒度的不同决定了负载均衡系统的性能优劣。按数据包粒度在多径中分派流量可以实现很理想的均衡效果。但是,当各路径中的时延值不同时,容易造成大量的乱序报文。报文乱序对TCP协议的业务流影响很大。TCP会把这种乱序看作是网络发生拥塞的指示,它将减小自己的发送窗口,使TCP的性能严重下降[1,2]。即使是一些UDP的业务流,如VOD,VOIP等,对报文乱序也很敏感。按流粒度分派数据量,也就是把业务流都指定到某个固定的路径上,可使同一个业务流沿着相同的路径到达接收端而不会出现报文乱序。但是,网络中每个流的大小、速率都存在很大的差别,一旦被指定到固定的路径上,在整个传输过程中都不会被改变,这会造成某路径被数据量大的流占用而导致拥塞,其它路径则处于相对空闲的状态。因此,按流粒度分派不能实现很好的均衡效果。

本文介绍了一种可以得到很好均衡效果并保证报文保序的负载均衡算法FSLB(flow splitting load balancing algorithm)。它根据业务流中报文之间的时间间隔对业务流进行切割,一个业务流切割成多个报文段,按报文段粒度分派负载。

2 相关研究工作

目前常见的报文分派方法主要分为两类。一类是基于包水平的负载分派方法,主要有轮循算法、加权轮循算法、最小

到稿日期:2010-01-25 返修日期:2010-04-15 本文受863国家重点基金项目(2008AA01Z214),863国家重点基金项目(2007AA01Z2a1),863国家重点基金项目(2007AA01Z212)资助。

卜佑军(1978—),男,博士生,讲师,主要研究方向为宽带信息网络、路由与交换技术,E-mail:buyoujun@gmail.com;王 超(1984—),男,硕士生,主要研究方向为高速转发技术;汪斌强(1963—),男,博士,教授,主要研究方向为宽带信息网络、高速路由器核心技术。

负载分派算法等，这些算法简单且易于实现，但会造成大量的乱序报文。OSPF 和 IS-IS 路由协议中都支持 ECMP(Equal-Cost Multipath)。文献[3,4]提出了 3 种解决等代价多路径流量分配的方法。另一类是基于流水平的负载分派方法，即属于相同流的报文被映射到同一个路径，这样不会出现报文乱序，其一般都是基于哈希算法实现的。文献[5]总结了几种哈希算法，并对各种哈希算法的性能进行了分析。Rost 和 Balakrishnan[6]评估了不同负载分派策略的性能，包括自适应感知的分派方法。文献[7]提出了一种基于 TCP 突发特性进行多径负载均衡的方法。

负载均衡需要把流量分派到不同的路径中，新近提出的几种协议，例如 TeXCP[8]、COPE[9]和 MATE[10]，都关注用动态的方法来实现期望的负载分派效果。为了克服 TCP 报文乱序，文献[11]提出通过对 TCP 协议进行局部修正的方法，来降低报文乱序的程度。

3 数据流的切割

3.1 数据流切割机制

网络中的数据流都具有突发性，每个流的大小、持续时间、速率不尽相同，网络中各路由器对业务流的处理时间可能也不同。相同业务流的数据包到达某个网络节点的时间间隔不可能是一样的。两个相邻的报文，其时间间隔可能会比较大，也可能会比较小。我们就是利用时间间隔的不同对同一数据流进行切割，切割后一个业务流由多个报文段串行组成，每个报文段中的报文个数是不固定的。

设用 β 表示切割时间门限，当同一业务流中相邻报文到达节点的时间间隔 t' 大于 β 时，对业务流进行切割，小于 β 时不切割。报文段内部相邻报文之间的时间间隔 $t1$ 肯定小于 β，前一报文段中最后一个报文与相邻的后一报文段中第一个报文的时间间隔 $t2$ 大于 β，如图 1 所示。β 为报文段内相邻报文时间间隔的最大值、相邻报文段之间时间间隔的最小值。

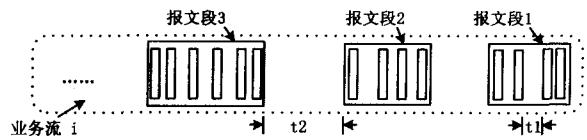

图 1 业务流切割

3.2 报文段粒度的保序

基于包水平粒度的流量分派，由于数据包的个数比流的个数要大很多，采用轮循算法(RR)等分派算法可以实现很好的均衡效果，但是会造成报文乱序，降低网络的性能。基于流水平粒度的流量分派，相同流的报文沿着相同的路径到达接收端，不会造成报文乱序，但是由于流的大小、速率的不同易造成负载分派的不均衡。基于报文段粒度的分派可以避免报文段沿着不同的路径到达接收端造成的乱序问题，还可以实现接近于包水平粒度分派的均衡效果。我们先介绍基于报文段粒度的报文分派怎样避免乱序问题。

考虑图 2 所示的多路径拓扑中，存在多条不相交的路径，每条路径中都包含多个网络节点。数据汇聚到一个发散节点，沿着不同的路径转发后再汇聚到聚合节点。

结论：同一个业务流中两个连续的数据报文，当其时间间隔大于所有路径时延差值的最大值，两个报文沿着不同路径到达聚合节点时不会造成报文乱序。即

$$t_{interal} > \max\{t_{pathi} - t_{pathj}\}; i,j \in [1,K] \quad (1)$$

式中，$t_{interal}$ 为同一业务流中两个连续数据报文的时间间隔，K 为不相交路径的路径数，$(t_{pathi} - t_{pathj})$ 为其中任意两个路径时延的差值。

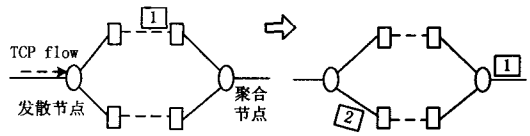

图 2 多路径拓扑

证明：假设第一个报文在时刻 $time1$ 到达发散节点，第二个报文在时刻 $time2$ 到达发散节点；第一个数据报文沿着路径 $pathi$ 到达聚合节点，$pathi$ 的路径时延为 t_{pathi}，第二个数据报文沿着 $pathj$ 到达聚合节点，$pathj$ 的路径时延为 t_{pathj}。从 $time1$ 时刻开始，第一个报文到达聚合节点所需的时间为 t_{pathi}，第二个报文到达聚合节点所需的时间为 $time2-time1+t_{pathj}$。若要保证报文传输不出现乱序，只要第一个数据报文先于第二个数据报文到达聚合节点即可。即

$$time2 - time1 + t_{pathj} > t_{pathi}$$

$$time2 - time1 > t_{pathi} - t_{pathj}$$

$$t_{interval} > t_{pathi} - t_{pathj}$$

如果 t_{pathi} 小于 t_{pathj}，不论两个报文的时间间隔有多小，都不会造成报文乱序，只有 t_{pathi} 大于 t_{pathj} 时才有可能会造成报文乱序。只要两个报文的时间间隔大于所有路径时延差值的最大值，不论两个报文沿着哪条路径到达聚合节点，都不会造成报文乱序，故式(1)成立。

所以，只要切割时间门限 β 大于所有并行路径时延差值的最大值 MDDT(Max differ delay time)，业务流中同一报文段的报文沿着同一路径到达聚合节点，不同报文段可独立地沿着不同路径到达，同一业务流的报文在聚合节点就不会产生乱序。

4 FSLB 均衡算法

FSLB 算法是一种数据包调度策略，适用于存在多径的路由系统中。当网络节点收到一个数据报文时，FSLB 为报文选取最合适的路径来转发报文，以实现各路径上按期望负载分配比来承担负载。FSLB 算法由 3 部分构成，算法的结构如图 3 所示。

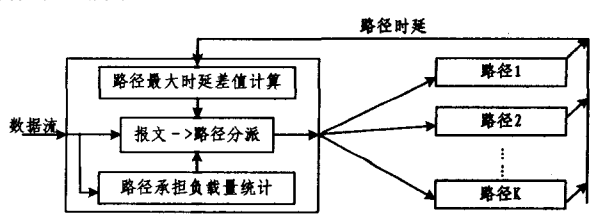

图 3 FSLB 算法的结构图

4.1 路径最大时延差值的计算

FSLB 算法使用周期探寻的方式来计算各路径最大时延差值 MDDT。周期性地向各路径发送时延探寻报文，得到各路径的时延值之后，计算路径最大时延差值。由于探寻方式得到的时延方差很大，需对得到的 MDDT 进行平滑处理，采用下式确定 MDDT：

MDDT=∂×(旧的 MDDT)+(1−∂)×新计算出来的

MDDT 样本

式中，$0 \leqslant \partial < 1$。∂接近于 1 时，表示新计算出的 MDDT 和旧的 MDDT 差别不大，新计算出来的 MDDT 受样本的影响不大；若∂接近于 0，则表示平均 MDDT 受新的 MDDT 样本影响很大。参照计算 RTT 的算法，最佳∂值为 7/8。

4.2 报文分派机制

FSLB 算法利用哈希表实现报文段与路径的映射。每条哈希表项中包含两个域：前一报文的到达时间（pre-packet time）、映射的路径（path-id）。当收到一个数据报文的时候，首先提取报文头部信息（源地址、目的地址、源端口号、目的端口号），将提取出的报文头部信息作为哈希函数的输入，得到的哈希值作为标识业务流的 ID，并作为关键字进行哈希查表。如果当前报文的到达时间小于 pre-packet time$+\beta$，说明当前报文与先前到达的报文属于同一报文段，报文送往表项中 path-id 指明的路径，pre-packet time 更新为当前报文的到达时间。否则，说明当前报文属于一个新的报文段并且是该报文段的第一个数据报文，可以被发送到其他的路径。FSLB 算法对每个路径都维护有一个计数器，这个计数器可以表明各路径的负载程度。FSLB 算法把新的报文段映射到计数器值最大的路径上，并把该路径的 path-id 设置为表项的 path-id，更新表项的 pre-packet time。报文分派机制算法的流程图如图 4 所示。

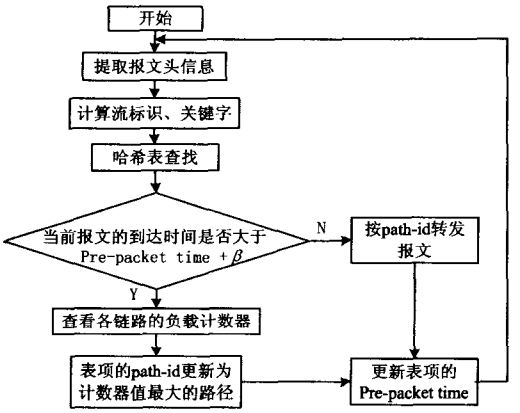

图 4 FSLB 算法中报文分派流程图

4.3 路径负载量计数算法

采用计数器来统计各路径的负载量，负载量计数算法表明了各路径承担负载量的差异程度。FSLB 算法给路径 i 分派一个计数器 C_i，当大小为 S 字节的数据报文到达时，每个路径的负载计数器按照下式更新：

$$C_i = C_i + F_i \times S$$

式中，F_i 为路径 i 的期望负载分配比，且有 $\sum_{i=1}^{K} F_i = 1$。报文按照报文分派机制中的算法分派到路径上传输。一旦报文被分派到具体的路径 j 上后，路径 j 的负载计数器相应地减小 S，即：

$$C_j = C_j - F_j \times S$$

其它路径的负载计数器不更新。

负载计数器可以简洁地指示出过去分派决定的累积效果。与期望分得的负载量相比，路径实际承担的负载越少，其负载计数器的值越大。FSLB 每当判断出报文属于新的报文段时，都会把该报文对应的流在哈希表中的 path-id 映射为负载计数器值最大的路径。这样就可以较理想地实现期望的负载均衡比。设定的计数器都是有计数上限的，若到达发散节点的数据报文其报文时间间隔一直都不满足业务流切割的条件，而是按照哈希表中指定的路径一直转发，直到某路径的负载计数器超出上限又重新开始计数后才出现新的报文段，则采用报文分派机制并不能实现很好的均衡效果。FSLB 算法采用设定测量间隔的方法（默认设定为 1s），避免了由于计数器溢出带来的相关问题。每当测量间隔，负载计数器都会复位并开始新的计数过程。

5 分析和仿真

5.1 报文段的分析

基于报文段粒度的负载分派算法把判定出的新的报文段分派到负载量最小的路径上，以实现期望的负载分配比，实际上是把一个业务流分成了若干个报文段。网络中比较小的业务流对均衡性能的影响是非常小的，均衡性能主要受大业务流的影响。那么，网络中大的业务流是否能够被切割成不同的报文段呢？通过分析采集到的 trace 文件中同一流报文时间间隔来说明这个问题。

表 1 中的数据源是通过采集校园网 Internet 接入点得到的。为了充分说明问题，我们对 2007 年 7 月 12 日 3 个不同时段的 trace 文件进行分析，各 trace 文件的信息如表 1 所列。

表 1 trace 文件信息

数据源	采集时段	大小(kB)	流个数	报文个数
Trace1	16:35—16:45	826,972	50135	1,920,459
Trace2	21:00—21:14	517,499	25537	1,221,142
Trace3	22:12—22:20	103,199	37355	997,116

选取不同的 β，得到的各 trace 中报文段数目的结果如表 2 所列。

表 2 β 不同时各 trace 中报文段的数目

β(ms)	Trace1	Trace2	Trace3
50	885182	540283	412503
100	763526	458643	316459
200	600027	383286	195239
500	432243	199627	148520
1000	321778	146594	93124
2000	200735	90814	78453
5000	121249	52029	52143
10000	77481	43137	46297

从表 2 的统计结果可以看出，β 取值较小时，切割后报文段的数目比较多，接近报文个数的一半。β 取值比较大时，报文段的数目接近于 trace 文件中流的个数。在满足报文切割条件下，β 决定了报文段的个数。

取 β=50ms，图 5 给出了每个流中平均的报文段个数。结果显示，包含字节数大的业务流，可以被分割成大量的尺寸相对较小的报文段。

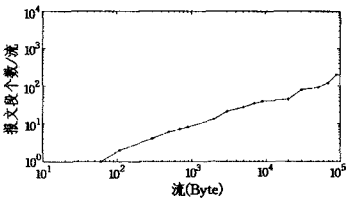

图 5 报文段个数随流大小增长而增加

（下转第 80 页）

Credit Card Risk Detection Based on Chaos Theory and Neural Network[C]//CESCE. Wuhan, China, 2010: 29-31

[5] 郭涛,李贵洋,袁丁. 基于置信度和神经网络的信用卡异常检测[J]. 计算机工程,2008,34(15):205-207

[6] Fawcett T. An introduction to ROC analysis[J]. Pattern Recogn Lett., 2006, 27: 861-874

[7] 葛唯益,程龚,瞿裕忠. 语义 Web 链接结构分析之综述[J]. 计算机科学,2010,37(3):17-21

[8] Getoor L, Diehl C P. Link mining: a survey[J]. ACM SIGKDD Explorations Newsletter, 2005, 7(1): 3-12

(上接第 69 页)

实际上,能把数据流切割成报文段的主要原因是 TCP 在 RTT 或稍小于 RTT 的时间段内的突发特性。我们知道,TCP 发送端通过一次或几次的突发,发送拥塞窗口大小的报文,而在每 RTT 或稍小于 RTT 的剩余时间内都在空闲等待。这主要是 TCP 机制中的 ACK 压缩、慢启动和其它因素造成的。TCP 的这种突发性使得一个长的 TCP 流可以看成是被空闲时间分隔的一些小的报文段的串联。

5.2 FSLB 算法的性能仿真

我们对 FSLB 算法采用基于 trace 驱动的仿真方式,利用表 1 中列出的 trace 作为分派数据,通过与包粒度分派的轮循算法(RR)、流粒度的静态哈希表算法(S-Hash)的比较,说明 FSLB 算法具有良好的均衡性能。

5.2.1 性能参数

通过以下两个参数衡量 FSLB 算法的性能:

· 负载分派误差 LI(Load inaccurate)。均衡系统中 F_i 为路径 i 的期望负载分配比,且有 $\sum_{i=1}^{K} F_i = 1$,K 为路径数。一个最佳的负载均衡系统,在任何时间,其负载分派算法都应该保证路径 i 的负载分配比为 F_i。但是实际中是不可能达到这种理想效果的。设路径 i 的实际负载分配比为 F_i',我们定义负载分派误差为:

$$LI=\frac{1}{K}\sum_{i=1}^{K}|F_i-F_i'|$$

· 报文乱序程度。TCP 机制中当报文超时、重传计时器超时或收到 3 个相同的 ACK 时,发送端重传报文,并认为此时网络发生拥塞,启用慢开始算法或快恢复算法,降低自己的拥塞窗口 cwnd。我们用重传的报文个数来衡量数据报文的乱序程度。

5.2.2 仿真结果及分析

均衡系统从发散节点到聚合节点有 3 条路径,它们期望的负载分配比为 $\vec{F}=(0.3, 0.3, 0.4)$。基于 trace 驱动的仿真得到的各负载分配方法的平均负载分派误差 LI 如图 6 所示。

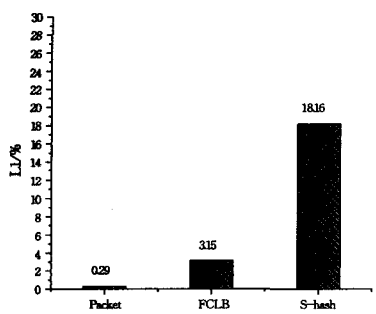

图 6 3 种负载分派算法的负载分派误差 LI

基于 trace 驱动的仿真可以体现出 Internet 网络中数据的特征,但是这种方法不能得到 FSLB 算法对 TCP 拥塞控制机制的响应。利用 ns2 对图 7 所示的网络拓扑进行 FSLB 算法仿真,通过改变链路时延 i,j,获得不同的最大路径时延差值,结果如图 8 所示。重传报文的个数与 β 和路径最大时延差值有关。当时延差值大于取定的 β 值时,重传的报文个数增加;反之,重传的报文非常少。这说明,只要 β 大于各路径时延差值的最大值,FSLB 算法就可以保证报文不出现乱序。

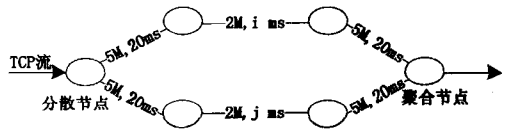

图 7 仿真拓扑

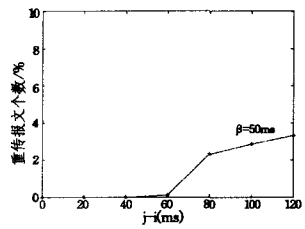

图 8 重传报文个数的百分比

结束语 本文提出采用基于相邻数据报文的时间间隔对同一业务流进行分割。分割后,一个业务流可以看成是由多个报文段串接而成。在有多条路径的均衡系统中,提出了 FSLB 负载分派算法,其使均衡系统在保证 TCP 流保序的同时,各路径能得到较理想的负载分配比。通过基于 trace 驱动的分析,FSLB 算法的均衡效果远远优于基于流水平的负载分派机制,稍逊于基于包水平的负载分派机制。

参 考 文 献

[1] Berkeley V J. TCP Evolution from 4. 3-Tahoe to 4. 3-Reno [C]// Proceedings of the 18th Internet Engineering Task Force. 1990: 365-374

[2] Brakmo L, O'malley S, Pererson L. TCP Vegas: New Techniques for Congestion Detection and Avoidance[C]// ACM SIGCOMM 1994. 1994: 24-35

[3] Thaler D, Hopps C. 2991 RFC[S]. November 2000

[4] Hopp C. 2992 RFC[S]. November 2000

[5] Cao Zhiruo, Wang Zheng, Zegura E. Performance of Hashing-Based Schemes for Internet Load Balancing[C]//Proceedings of IEEE INFOCOM. 2000: 332-341

[6] Rost S, Balakrishnan H. Rate-aware splitting of aggregate traffic [R]. MIT, 2003

[7] Sinha S, Kandul S, Katabi D. Harnessing TCP's Burstiness with Flowlet Switching[C]// 3rd ACM SIGCOMM Workshop on Hot Topics in Networks. San Diego, November 2004

[8] Kandul S, Katabi D, Davie B, et al. Walking the Tightrope: responsive yet stable traffic engineering[C]// SIGCOMM. 2005

[9] Wang H, Xie H, Qiu L, et al. Cope: Traffic engineering in dynamic networks[C]// ACM SIGCOMM. 2006

[10] Elwalid A, Jin C, Low S H, et al. MATE: MPLS Adaptive Traffic Engineering[C]// INFOCOM. 2001

[11] Lee Youngseok, Park Ilkyu, Choi Yanghee. Improving TCP Performance in Multipath Packet Forwarding Networks[J]. Communication and Networks, 2002, 4(2): 1-10