* 1

一种改善与 TCP Reno 兼容性的 TCP Vegas 改进算法

王建峰，　康巧燕，　张　辉
(空军工程大学　电讯工程学院，陕西　西安　710077)

摘　要：在分析 TCP Vegas 及其相关改进算法优缺点的基础上，针对 TCP Vegas 在与 TCP Reno 共享带宽时存在不兼容的问题，基于 TCP Vegas－A 算法，引入相对队列时延的拥塞状态判断方法，提出了一种 Vegas 改进算法 TCP Vegas－A＋。新算法将路由器缓存占用量和相对队列时延相结合，把网络状态进一步细分成拥塞增加和拥塞减轻状态，以更准确地判断网络拥塞情况、适时合理地调整拥塞窗口。分阶段对各算法的拥塞窗口大小、所传输的分组数进行数学计算，分析 Vegas－A＋连接与 Reno 连接的兼容性，并与 Vegas＋连接与 Reno 连接的兼容性进行比较，同时利用仿真实验进行验证。数学分析和仿真结果表明，Vegas－A＋算法能更准确判断网络状态，改善了与 TCP Reno 的兼容性，能和 TCP Reno 较公平地竞争带宽。

关键词：拥塞控制；TCP Vegas 算法；兼容性
DOI：10.3969/j.issn.1009－3516.2010.01.015
中图分类号：TN929.5　　**文献标识码**：A　　**文章编号**：1009－3516(2010)01－0064－06

TCP Vegas 算法[1]通过检测预期吞吐量与实际吞吐量的差值，预测拥塞的发生。相对于 TCP Reno，TCP Vegas 能多获得 37%－71%的吞吐量，减少 20%－50%的丢包率[2]。尽管 TCP Vegas 具有很多优点，但是要在实际网络中广泛应用还有差距。文献[3]指出由于 TCP Vegas 探测带宽的灵敏性，使得它较早地对网络的状况做出响应，从而过早结束慢启动，进入拥塞避免阶段，这大大降低了 Vegas 的性能。文献[4－5]指出，当 Vegas 与 Reno 竞争带宽时存在兼容性问题，且原有的连接相对于新连接受到不公平对待，从而可能导致持续拥塞问题。

针对 Vegas 存在的上述问题，研究者提出了一系列的改进算法[3－9]。文献[3]针对 Vegas 在高时延链路中应用时存在的长传播时延、延时应答的影响及慢启动过早结束等问题在发送端进行改进，提出了 TCP New Vegas 算法。文献[4]针对 Vegas 算法中阈值 α 和 β 固定不变，拥塞控制效果受限，提出了Vegas－A算法，使 α 和 β 的值可以自动调整，较好地适应网络状况的变化。文献[8]针对 Vegas 和 Vegas－A 与 Reno 竞争带宽时存在的兼容性问题，提出了一种能较好地与 Reno 竞争带宽的改进算法 Vegas＋。但是，在上述的 Vegas 改进算法中，没有考虑到拥塞的变化趋势问题，在与 Reno 竞争带宽时性能仍不够理想，为此，本文在 TCP Vegas－A 算法的基础上，引入 TCP New Veno 算法[10]中的基于相对队列时延的网络状态判断算法，对 Vegas－A 的拥塞避免机制进行改进，数学分析与仿真结果表明，改进算法在与 Reno 共享带宽时具有较好的兼容性。

1　Vegas－A＋算法

在 Vegas－A 算法的基础上，结合基于相对队列时延的拥塞状态区分方法，针对拥塞避免阶段提出一种能较好地与 TCP Reno 竞争带宽的 Vegas 改进算法，称为 TCP Vegas－A＋。算法的基本思想是：在 Vegas－A 算法的基础上，对网络的通信状况进一步细分。首先比较网络当前的吞吐量与上一个 RTT 内的吞吐量，若

* **收稿日期**：2009－07－02
基金项目：陕西省自然科学基金资助项目(2009JQ8008)
作者简介：王建峰(1978－)，男，陕西武功人，讲师，主要从事卫星通信、网络拥塞控制研究.
E－mail：wjf4206@126.com

当前吞吐量大于前一个吞吐量时,表明网络并未饱和;反之,说明网络趋近于饱和状态,这时通过相对队列时延对网络状态进一步进行判断,根据判断的结果采取相应的措施。

假设发送端分别在时刻 S_n 和 S_{n+1} 发送 2 个连续的分组 packet_n 和 packet_{n+1},在时刻 R_n 和 R_{n+1} 分别接收到 packet_n 和 packet_{n+1} 的 ACKs。定义相对队列时延 D_q 为连续 2 个分组的 ACK 接收时间间隔与这 2 个分组的发送时间间隔之差,则可表示如下:

$$D_q = (R_{n+1} - R_n) - (S_{n+1} - S_n) \tag{1}$$

通过 D_q 可判断网络的拥塞状态,若 $D_q \leqslant 0$,即连续 2 个分组的 ACK 接收时间间隔小于等于发送时间间隔,说明分组所经过的中间节点的队列时延在减少,网络处于拥塞减轻状态;若 $D_q > 0$,即连续 2 个分组的 ACK 接收时间间隔大于发送时间间隔,说明分组由于中间节点的队列时延增加而导致网络拥塞程度加剧,网络处于拥塞增加状态。图 1 为 Vegas – A + 算法的详细流程图。

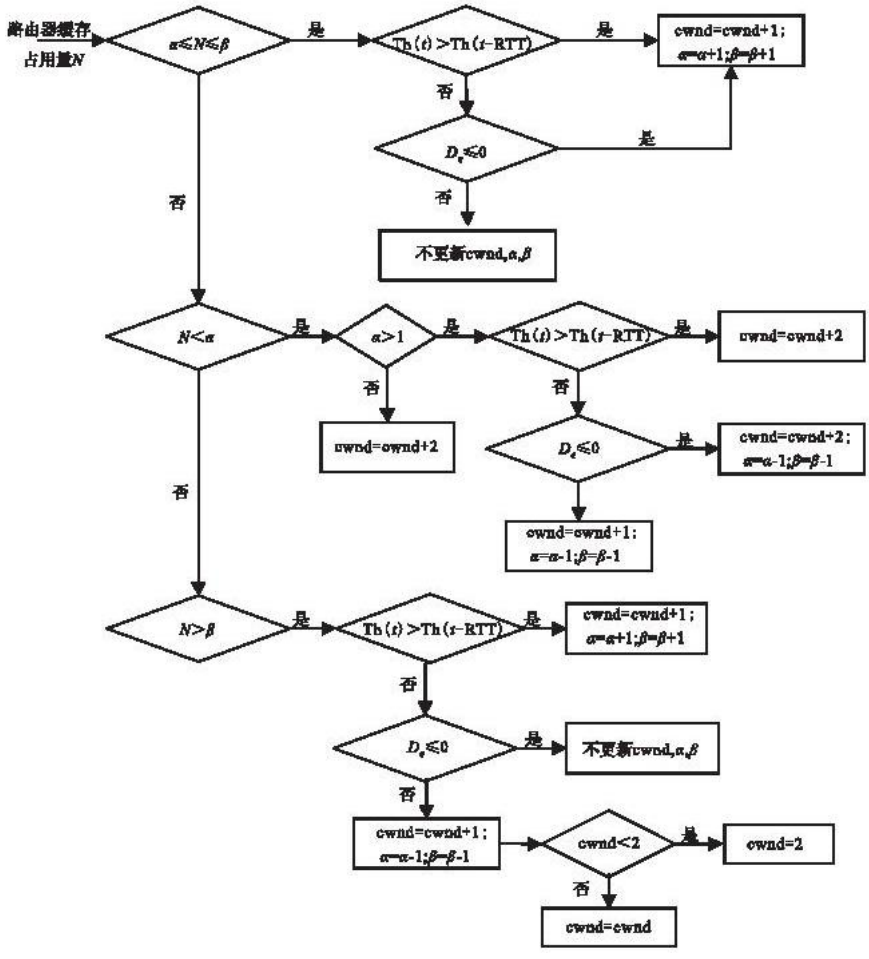

图 1　TCP Vegas – A + 算法流程图

Fig. 1　TCP Vegas – A + algorithm flow chart

图中:N 为路由器缓存占用量;cwnd 表示当前拥塞窗口的大小;RTT 表示往返时间;Th(t)表示当前的吞吐量;Th(t – RTT)表示上一个 RTT 的吞吐量;阈值 α 和 β 的初值分别取为 x 和 z。

2　Vegas – A + 与 Reno 兼容性分析

这里,把拥塞避免阶段分成 4 个小阶段,对 Vegas – A + 连接与 Reno 连接的兼容性(竞争带宽能力)进行数学分析,并与 Vegas + 算法与 Reno 的兼容性进行比较。当发送端接收到接收端反馈的 ACK 后,根据算法来调整发送窗口的大小,这表明新一轮的开始,直达发送端接收到新的 ACK 分组,这一轮才结束,同时开始下一个新的轮。假设网络模型左右对称,分别考虑 m 条 Reno 连接与 n 条 Vegas – A + 连接共享瓶颈链路和 m 条 Reno 连接与 n 条 Vegas + 连接共享瓶颈链路时,各个连接的带宽占有情况,模型中涉及的参数见表 1。

表 1 模型中涉及的参数及其描述

Fig. 1 Parameter of model and its description

参数	描述	参数	描述
B	瓶颈链路上的带宽大小	minRTT	最小往返时间
T_i	第 i 个阶段所经历的时间	$b_{(j)}$	第 j 轮结束时缓存队列的长度
$\bar{b}_{(j)}$	第 j 轮的平均队长	$\mathrm{RTT}_{(j)}$	拥塞避免阶段第 j 轮的往返时间
$P'_{V(i)}$	第 i 个阶段 Vegas + 发送的分组数	$P'_{R(i)}$	第 i 个阶段 Reno 发送的分组数(与 Vegas + 共享瓶颈链路)
$P^{\mathrm{new}}_{V(i)}$	第 i 个阶段 Vegas - A + 发送的分组数	$P^{\mathrm{new}}_{R(i)}$	第 i 个阶段 Reno 发送的分组数(与 Vegas - A + 共享瓶颈链路)
$W'_{V(j)}$	第 j 轮 Vegas + 的拥塞窗口大小	$W'_{R(j)}$	第 j 轮 Reno 的拥塞窗口大小(与 Vegas + 共享瓶颈链路)
$W^{\mathrm{new}}_{V(j)}$	第 j 轮 Vegas - A + 的拥塞窗口大小	$W^{\mathrm{new}}_{R(j)}$	第 j 轮 Reno 的拥塞窗口大小(与 Vegas - A + 共享瓶颈链路)

1)第 1 阶段:从慢启动结束到链路带宽被完全占用。

从慢启动结束到满足 $n\dfrac{W^{\mathrm{new}}_{V(j)}}{\mathrm{RTT}}+m\dfrac{W^{\mathrm{new}}_{R(j)}}{\mathrm{RTT}}\leqslant B$ 的 t 被划分为第 1 阶段,此阶段瓶颈链路上的路由器缓存占用量为 0,继而其每轮的往返时间 RTT = minRTT。该阶段的轮数为从 1 到 l_1,Reno 连接每轮窗口只增加 1,Vegas - A + 连接每轮窗口增加 2,所以有:

$$W^{\mathrm{new}}_{R(i)}=W^{\mathrm{new}}_{R(1)}+(i-1)\qquad(1\leqslant i\leqslant l_1)\tag{2}$$

$$W^{\mathrm{new}}_{V(i)}=W^{\mathrm{new}}_{V(1)}+2(i-1)\qquad(1\leqslant i\leqslant l_1)\tag{3}$$

由于第 l_1 轮刚好是这 $m+n$ 个连接占用完所有带宽的一轮,于是有:

$$n\frac{W^{\mathrm{new}}_{V(l_1)}}{\mathrm{minRTT}}+m\frac{W^{\mathrm{new}}_{R(l_1)}}{\mathrm{minRTT}}=B\tag{4}$$

联立式(2)-(4)可解得:

$$W^{\mathrm{new}}_{V(l_1)}=\frac{2B\mathrm{minRTT}+mW^{\mathrm{new}}_{V(1)}-2mW^{\mathrm{new}}_{R(1)}}{m+2n}\tag{5}$$

$$W^{\mathrm{new}}_{R(l_1)}=\frac{B\mathrm{minRTT}+2nW^{\mathrm{new}}_{R(1)}-nW^{\mathrm{new}}_{V(1)}}{m+2n}\tag{6}$$

当 Vegas + 与 Reno 共享瓶颈链路时,在第 n_1 轮第 1 阶段结束时[8],有:

$$W'_{V(n_1)}=\frac{B\mathrm{minRTT}+mW'_{V(1)}-mW'_{R(1)}}{m+n}\tag{7}$$

以上式中的 $W^{\mathrm{new}}_{V(1)}=W'_{V(1)}$,$W^{\mathrm{new}}_{R(1)}=W'_{R(1)}$ 为慢启动阶段结束时留下来的值,且有 $W'_{V(1)}<W'_{R(1)}$[8],从而在该阶段结束时,Vegas - A + 与 Vegas + 的窗口大小之差为:

$$\Delta W_{(1)}=W^{\mathrm{new}}_{V(l_1)}-W'_{V(n_1)}=\frac{m\mathrm{minRTT}}{(m+2n)(m+n)}\left(B-\frac{mW'_{R(1)}}{\mathrm{minRTT}}-\frac{nW'_{V(1)}}{\mathrm{minRTT}}\right)\geqslant 0\tag{8}$$

每个 TCP Reno 连接和 TCP Vegas - A + 连接在第 1 阶段传输的分组数分别为:

$$P^{\mathrm{new}}_{R(1)}=\sum_{i=1}^{l_1}W^{\mathrm{new}}_{R(i)}=l_1W^{\mathrm{new}}_{R(1)}+\frac{l_1(l_1-1)}{2}\tag{9}$$

$$P^{\mathrm{new}}_{V(1)}=\sum_{i=1}^{l_1}W^{\mathrm{new}}_{V(i)}=l_1W^{\mathrm{new}}_{V(1)}+l_1(l_1-1)\tag{10}$$

从而在第 1 阶段,每个 Vegas - A + 与 Reno 连接传输的分组数之差为:

$$\Delta P^{\mathrm{new}}_{(1)}=P^{\mathrm{new}}_{V(1)}-P^{\mathrm{new}}_{R(1)}=l_1(W^{\mathrm{new}}_{V(1)}-W^{\mathrm{new}}_{R(1)})+\frac{l_1(l_1-1)}{2}\tag{11}$$

而由文献[8]可知,在这个阶段,每个 Vegas + 与 Reno 连接传输的分组数差为:

$$\Delta P'_{(1)}=n_1(W'_{V(1)}-W'_{R(1)})\tag{12}$$

式中 $n_1=l_1(m+2n)/(m+n)$,比较式(11)和式(12),明显有 $\Delta P^{\mathrm{new}}_{(1)}>\Delta P'_{(1)}$,这说明,Vegas - A + 算法在第 1 阶段中与 Reno 竞争带宽的能力优于 Vegas + 算法,且由式(8)可知,该阶段结束时 Vegas - A + 算法的拥塞窗口大小将不小于 Vegas + 的拥塞窗口。

2)第 2 阶段:路由器中缓存占用量不大于阈值 α。

满足 $0<\left(\frac{W^{new}_{V(t)}}{\text{minRTT}}-\frac{W^{new}_{V(t)}}{\text{RTT}_{(t)}}\right)\text{minRTT}\leqslant\alpha$ 的 t 被划分为第 2 阶段,此阶段带宽被各连接占用完,但路由器中缓存占用量还未达到阈值 α,它的往返时间由于排队而被延长了,即:

$$\text{RTT}_{(t)}=\text{minRTT}+\frac{\bar{b}_{(t)}}{B} \tag{13}$$

第 2 阶段从 l_1 轮到 l_2 轮,把 $t=l_2$ 代入 $\left(\frac{W^{new}_{V(t)}}{\text{minRTT}}-\frac{W^{new}_{V(t)}}{\text{RTT}_{(t)}}\right)\text{minRTT}=\alpha$,变形可得:

$$W^{new}_{V(l_2)}=\alpha\frac{\text{RTT}_{(l_2)}}{\text{RTT}_{(l_2)}-\text{minRTT}}=\alpha\frac{\text{RTT}_{(l_2)}}{b_{(l_2)}/B}=\frac{\alpha B\text{RTT}_{(l_2)}}{b_{(l_2)}} \tag{14}$$

由式(13)和式(14)可得:

$$W^{new}_{V(l_2)}b_{(l_2)}/\alpha B=\text{minRTT}+\bar{b}_{(l_2)}/B \tag{15}$$

当 $\alpha=x$ 时,Vegas - A + 与 Reno 的窗口增长方式与在第 1 阶段中的增长方式相同,采用类似的分析方法,可得 2 种连接在该过程发送的分组数分别为:

$$P^{new}_{R(2)}=\sum_{i=l_1+1}^{l_2}W^{new}_{R(i)}=(l_2-l_1)W^{new}_{R(l_1)}+(l_2-l_1-1)(l_2-l_1)/2 \tag{16}$$

$$P^{new}_{V(2)}=\sum_{i=l_1+1}^{l_2}W^{new}_{V(i)}=(l_2-l_1)W^{new}_{V(l_1)}+(l_2-l_1-1)(l_2-l_1) \tag{17}$$

从而有:

$$\Delta P^{new}_{(2)}=P^{new}_{V(2)}-P^{new}_{R(2)}=(l_2-l_1)(W^{new}_{V(l_1)}-W^{new}_{R(l_1)})+(l_2-l_1-1)(l_2-l_1)/2 \tag{18}$$

由文献[8]有:

$$\Delta P'_{(2)}=(n_2-n_1)(W_{V(n_1)}-W_{R(n_1)}) \tag{19}$$

明显有 $\Delta P^{new}_{(2)}>\Delta P'_{(2)}$,这说明在第 2 阶段中,当 $\alpha=x$ 时,改进算法 Vegas - A + 较 Vegas/Vegas + 具有较好的竞争带宽能力。

由于 Vegas - A + 算法在第 3 阶段中,当吞吐量仍然增加时,增大拥塞窗口大小的同时,增加了 α 和 β 的值,随着吞吐量的继续增加,这将导致算法又回到第 2 阶段,并且 $\alpha>x$[8]。

当 $\alpha>x$ 时,若 $\text{Th}(t)$ 持续增加,Vegas - A + 算法保持阈值不变,而将窗口大小 $W^{new}_{V(t)}$ 加 2。此过程中,改进算法的拥塞控制策略与该阶段 $\alpha=x$ 时相同,因此 Vegas - A + 算法与 Reno 竞争的性能将优于 Vegas + ;若 $\text{Th}(t)$ 减小,Vegas - A + 算法通过相对队列时延 D_q 进一步区分网络的拥塞状态,若 $D_q>0$,拥塞窗口增加 1,若 $D_q\leqslant 0$,拥塞窗口增加 2,阈值 α 和 β 均减小,直至 α 减小到 x。此时,当 $D_q>0$ 时,Vegas - A + 算法的拥塞控制策略与 Vegas + 相同,在第 1 阶段结束时,Vegas - A + 算法具有不小于 Vegas + 的拥塞窗口,性能将至少与 Vegas + 相同;当 $D_q\leqslant 0$ 时,由于拥塞窗口以更大的速度增加,Vegas - A + 算法的性能优于 Vegas + 。由此可见,Vegas - A + 算法在这一过程中与 Reno 竞争带宽的性能将优于 Vegas + 。

通过对该阶段3 种情况的分析可知,Vegas - A + 算法与 Reno 竞争带宽的性能优于 Vegas + 算法,这也说明,在该阶段结束时 Vegas - A + 算法的拥塞窗口大小将大于 Vegas + 的拥塞窗口。

3)第 3 阶段:路由器中缓存占用量在阈值 α 和 β 之间。

满足 $\alpha<\left(\frac{W^{new}_{V(t)}}{\text{minRTT}}-\frac{W^{new}_{V(t)}}{\text{RTT}_{(t)}}\right)\text{minRTT}\leqslant\beta$ 的 t 被划分为第 3 阶段,从第 l_2+1 轮到第 l_3 轮。

在这个阶段中,当吞吐量 $\text{Th}(t)$ 仍然增加时,Vegas - A + 算法与 Vegas + 算法的拥塞控制策略相同,即将窗口大小与阈值 α 和 β 分别加 x。但由于第 2 阶段结束时,Vegas - A + 算法具有大于 Vegas + 的拥塞窗口,因此性能将优于 Vegas + 。

当吞吐量 $\text{Th}(t)$ 减小时,若 $D_q>0$,Vegas - A + 采用与 Vegas + 相同的拥塞控制策略,即保持拥塞窗口与阈值 α 和 β 的大小不变,但由于第 2 阶段结束时,Vegas - A + 算法具有大于 Vegas + 的拥塞窗口,因此性能将优于 Vegas + 。若 $D_q\leqslant 0$,将窗口大小与阈值 α 和 β 分别加 x,而 Vegas + 保持拥塞窗口与阈值 α 和 β 的大小不变,很明显,这个过程 Vegas - A + 性能优于 Vegas + 。

通过对该阶段上述 2 种情况的分析可知,Vegas - A + 算法的竞争能力较 Vegas + 得到了有效地改善,相应的,该阶段结束时,Vegas - A + 连接的窗口大小大于 Vegas + 连接的窗口大小。

4)第 4 阶段:路由器中缓存占用量大于阈值 β。

满足 $\left(\frac{W^{new}_{V(t)}}{\text{minRTT}}-\frac{W^{new}_{V(t)}}{\text{RTT}_{(t)}}\right)\text{minRTT}\geqslant\beta$ 的 t 划分为第4阶段,当吞吐量 $\text{Th}(t)$ 仍然增加时,Vegas－A＋采用与 Vegas＋相同的拥塞控制策略,但由于 Vegas－A＋在上一阶段结束时具有较大的拥塞窗口,因此,这个过程中 Vegas－A＋与 Reno 竞争带宽的性能优于 Vegas＋。

当吞吐量 $\text{Th}(t)$ 开始减小时,Vegas－A＋通过 D_q 进一步区分网络的拥塞状态,若 $D_q\leqslant 0$,拥塞窗口和阈值均保持不变;若 $D_q>0$,Vegas－A＋通过减小拥塞窗口和阈值来对拥塞进行控制。此时,当 $D_q\leqslant 0$ 时,很明显,Vegas－A＋算法竞争带宽的性能优于 Vegas＋算法。当 $D_q>0$ 时,Vegas－A＋的拥塞控制策略与Vegas＋相同,但由于上一阶段结束时,Vegas－A＋具有较大的拥塞窗口,因而该过程中,Vegas－A＋算法竞争带宽的性能依然优于 Vegas＋算法。

综上可知,Vegas－A＋算法与 Reno 的兼容性优于 Vegas＋算法与 Reno 的兼容性。

3 性能评估

采用 NS2 仿真对 Vegas－A＋与 Reno 算法竞争带宽的性能进行验证。网络仿真拓扑及相关参数见图2。瓶颈链路采用 Drop－tail 策略,路由缓冲区大小为50个分组,分组大小为500 bytes,网络传输的数据类型为 FTP。连接 S_1-D_1 之间采用 TCP Vegas－A＋或者 Vegas＋算法,连接 S_2-D_2 之间采用 TCP Reno 算法。S_1 首先启动,10 s 后 S_2 启动,仿真运行 200 s。

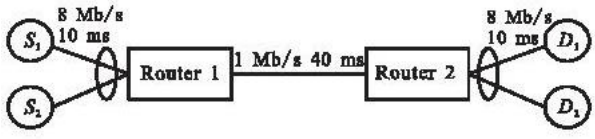

图2 网络仿真拓扑

Fig. 2 Network simulation topology

图3和图4分别给出了 TCP Vegas＋与 TCP Reno 共享瓶颈链路、TCP Vegas－A＋与 TCP Reno 共享瓶颈链路时的平均吞吐量。从图中可以看出,当 S_2 在 10 s 开始传输 Reno 流时,它开始与 Vegas＋/Vegas－A＋竞争带宽。比较图3和图4可知,在与 Reno 竞争带宽时,Vegas－A＋流能获得比 Vegas＋流更高的吞吐量,具有比 Vegas＋更好的竞争带宽的性能,这与上一节的数学分析相一致。因为 Vegas－A＋连接以比 Vegas＋连接更激进的方式增加拥塞窗口,能使拥塞窗口 cwnd 达到更大的值,并且 Vegas－A＋算法增加了拥塞状态判断机制,能够较好地判断网络所处的拥塞状况,避免了不必要的拥塞窗口减小以及慢启动的启动。

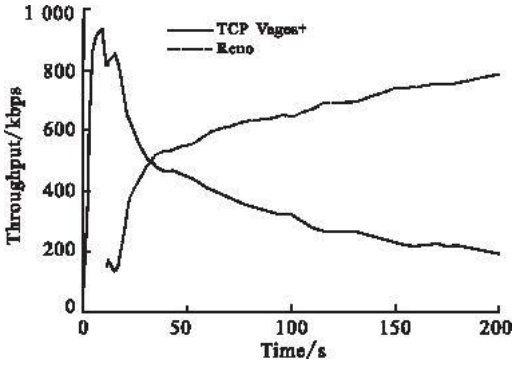

图3 Vegas＋与 Reno 连接竞争带宽性能比较

Fig. 3 Performance compare of TCP Vegas＋and Reno in bandwidth sharing

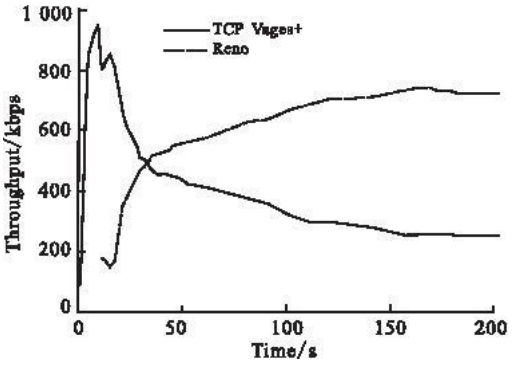

图4 Vegas－A＋与 Reno 连接竞争带宽性能比较

Fig. 4 Performance compare of TCP Vegas－A＋and Reno in bandwidth sharing

4 结束语

TCP Vegas 相对于目前广泛应用的 TCP Reno 能获得更高的吞吐量和更低的丢包率,但由于与 Reno 竞争带宽时存在较严重的不兼容问题,未能在实际网络中得以广泛应用。针对这一问题,文中结合 TCP Vegas－A 算法和 TCP New Veno 算法的优势,提出了改进算法 TCP Vegas－A＋。改进算法引入的基于相对队列时延的拥塞状态判断机制,可对网络状态进一步细划,增加了网络状态判断的准确性。通过将拥塞避免划分为4个小阶段的数学分析表明,改进算法能较准确判断网络状态,提高了与 Reno 竞争带宽的能力,改善了与

Reno 的兼容性,并通过仿真实验对改进算法的有效性进行验证。

参考文献:

[1] Brakmo Lawrence S, O' Mally Sean W, Peterson Larry L. TCP Vegas: New Techniques for Congestion Detection and Avoidance [C]//Proceedings of Conference on Communications Architectures Protocols and Application. New York: ACM, 1994: 1024 - 1035.

[2] Brakmo Lawrence S, L Larry. TCP Vegas: End to End Congestion Avoidance on A Global Internet[J]. IEEE Journal on Selected Areas in Communications, 1995, 13(8): 1465 - 1480.

[3] Sing Joel, Soh Ben. TCP New Vegas: Improving the Performance of TCP Vegas Over High Latency Links [C]//Proceedings of the 2005 4th IEEE International Symposium on Network Computing and Applications (NCA'05). Cambridge, MA: IEEE Press, 2005: 73 - 82.

[4] Srijith K N, Jacob Lillykutty, Ananda A L. TCP Vegas - A: Improving the Performance of TCP Vegas [J]. Computer Communications, 2005, 28(4): 429 - 440.

[5] 周铁军, 寇小文, 李阳. TCP Vegas 和 TCP Reno 的兼容性问题及其解决办法[J]. 湘潭大学自然科学学报, 2008, 30(4): 130 - 134.
ZHOU Tiejun, KOU Xiaowen, LI Yang. A Solution to Compatibility between TCP Vegas and TCP Reno [J]. Natural Science Journal of Xiangtan University, 2008, 30(4): 130 - 134. (in Chinese)

[6] Sing Joel, Soh Ben Soh. TCP New Vegas: Performance Evaluation and Validation [C]//Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC'06). Cagliari, Sardinia, Italy: IEEE Press, 2006: 541 - 546.

[7] Sing Joel, Soh Ben. TCP New Vegas Revisited [J]. IEEE 7th Malaysia International Conference on Communication. [S. l.]: IEEE Press, 2005: 6 - 15.

[8] 孟川杰. TCP Vegas 的拥塞控制算法研究[D]. 成都:四川大学, 2006.
MENG Chuanjie. The Research of TCP Vegas Algorithm [D]. Chengdu: Sichuan University, 2006. (in Chinese)

[9] 沈明玉, 刘平. MANET 中 TCP Vegas 拥塞机制的改进[J]. 合肥工业大学学报: 自然科学版, 2009, 32(4): 457 - 460.
SHEN Mingyu, LIU Ping. Improvement of TCP Vegas Congestion Control Mechanism in the Mobile Ad Hoc Network [J]. Journal of Hefei University of Technology: Natural Science Edition, 2009, 32(4): 457 - 460. (in Chinese)

[10] Cho Namho, Chung Kwangsue. TCP - New Veno: The Energy Efficient Congestion Control in Mobile Ad - Hoc Networks [C]//Conference on Emebdded and Ubiquious Computing. Berlin, Germany: Springer - Verlag, 2006: 254 - 263.

(编辑:徐楠楠)

An Improved Vegas Algorithm for Enhancing Compatibility with TCP Reno

WANG Jian - feng, KANG Qiao - yan, ZHANG Hui

(Telecommunication Engineering Institute, Air Force Engineering University, Xi'an 710077, China)

Abstract: TCP Vegas can provide a better performance compared to the traditional TCP Reno scheme. However, when TCP Vegas and TCP Reno connections coexist and share the same link in wired network, there exists a serious incompatibility problem of unfair bandwidth sharing in favor of TCP Reno. To solve this problem, by adding monitoring algorithm of the network state based on relative queuing delay to TCP Vegas - A, an enhanced Vegas algorithm is proposed, termed TCP Vegas - A +. TCP Vegas - A + can adjust the congestion window more properly and timely by classifying the network state into congestion increase and congestion decrease. We analyze the compatibility of Vegas - A + and Reno connections by computing and comparing the congestion window size and the number of transmitted packets in congestion avoidance phase, and compare it with the compatibility of Vegas + and Reno connections. Mathematical analysis and simulation results show that, TCP Vegas - A + algorithm improves the compatibility with TCP Reno and has a better capacity in competition with the Reno for bandwidth than other improved Vegas algorithms.

Key words: congestion control; TCP Vegas algorithm; compatibility