TCP Vegas 重选路问题及其解决方法*)

岳 鹏 张 冰 刘增基 曾伟军
(西安电子科技大学综合业务网国家重点实验室 西安 710071)

摘 要 为克服传统 Vegas 机制在网络层重选路后可能出现的吞吐量劣化问题,提出了一种称为"主动激励"的新机制。该机制的基本思想是:当 TCP 拥塞窗口(cwnd)稳定在某个平衡点上时,源端主动地增加基准往返时延,以打破这种平衡,激励 Vegas 进行窗口调整,通过 Vegas 自身的窗口调整机制使 cwnd 达到一个新的平衡,进而对 Vegas 连接的吞吐量进行有效的恢复。"主动激励"机制并不修改 Vegas 算法且开销很小,可作为一个独立模块内嵌到 Vegas 或其增强算法中,从而可以容易地对这些算法进行扩充。

关键词 TCP Vegas,拥塞控制,重选路,主动激励

A Rerouting Issue with TCP Vegas and its Solution

YUE Peng ZHANG Bing LIU Zeng-Ji ZENG Wei-Jun
(State Key of Integrated Service Networks, Xidian University, Xi'an 710071)

Abstract To overcome the problem of original Vegas scheme which probably brings forth the throughput degradation after rerouting operations, we propose a novel mechanism called "active spurring", whose basic idea is that the source actively increases the BaseRTT to break down the equilibrium when the congestion window(cwnd) is stabilized to some equilibrium, and then Vegas is spurred to adjust the cwnd to be stabilized to a new equilibrium by the basic window adjustment mechanism in Vegas, effectively restoring the throughput of Vegas connections accordingly. "Active spurring"does not modify Vegas and can be simply embedded into the TCP Vegas or enhanced TCP Vegas as an independent module with a low overhead and hence be extended easily.

Keywords TCP Vegas, Congestion control, Rerouting, Active spurring

1 引言

随着因特网规模的不断膨胀和业务的快速增长,网络拥塞随时都有可能发生。拥塞控制机制对保障网络的正常运行和服务质量(QoS)起着非常关键的作用,如何更有效地使用网络资源(如链路带宽、缓存)成为设计拥塞控制算法的一大挑战。传统的 TCP 拥塞控制算法(如 TCP Reno)主要是根据分组丢失信息探测可用带宽,是一种被动算法。TCP Vegas[1]与之不同,它基于往返时延(Round Trip Time,RTT)估计通路的可用带宽,并根据测量结果提前调整源端的发送速率,是一种主动算法。试验和仿真[2,3]表明,通常情况下,TCP Vegas 较之 TCP Reno 能够获得更高的吞吐量和更低的分组丢失,但是 Vegas 却存在诸如 TCP 公平[4,5]、非对称网络[6]以及持续拥塞[7]等问题,这些问题将导致 Vegas 连接出现不必要的吞吐量劣化。因此,许多学者从不同角度对 Vegas 进行研究,针对其存在的问题对 Vegas 算法进行改进。本文分析讨论另一个可能引起 Vegas 吞吐量劣化的问题,即重选路问题。与文[4~7]中 Vegas 连接吞吐量暂时劣化的问题不同,重选路问题对 Vegas 连接的吞吐量影响将是长期的。研究发现,当重选路后连接经历更长的往返通路时延(Round Trip Path Time,与 RTT 不同,RTPT 不包含中间结点的排队时延)时,即使瓶颈链路有大量的可用带宽,Vegas 连接吞吐量仍旧会保持在较低的水平而无法根据链路的实际状况正确地调整连接的发送速率,导致 Vegas 窗口调整机制失效,严重影响 Vegas 在因特网中的应用。针对 Vegas 重选路问题,笔者提出一种称为"主动激励"的新机制,该机制主动增加基准往返时延(BaseRTT),对通路的状态进行估计,克服了传统 Vegas 机制不能正确判断往返时延增加是因为拥塞还是网络层重选路造成的缺陷。另外,"主动激励"机制可作为一个独立模块内嵌到 Vegas 或其增强算法中,从而可以容易对当前的 Vegas 或其增强算法进行扩充。

2 TCP Vegas 重选路问题

TCP 是面向连接的传送层协议,而重选路发生在网络层,是 IP 路由需要解决的问题。表面看来两者似乎没有什么关系,可是对于 TCP Vegas 而言,由于其利用测量的 RTT 调整拥塞窗口,如果 Vegas 不能正确地估计因重选路而引起的时延变化,则将直接影响 Vegas 连接的吞吐量,然而 Vegas 算法本身没有任何机制处理连接的重选路所引起的时延变化。为了进一步说明重选路对 Vegas 吞吐量的影响,首先对 Vegas 窗口调整机制进行简单的分析。文[1]中给出的拥塞避免状态下 Vegas 窗口调整机制可用下式表示:

$$cwnd_{next}=\begin{cases} cwnd_{cur}+1 & Diff\times BaseRTT<\alpha \\ cwnd_{cur}-1 & Diff\times BaseRTT>\beta \\ cwnd_{cur} & \alpha<Diff\times BaseRTT<\beta \end{cases} \quad (1)$$

式中 $cwnd_{cur}$ 和 $cwnd_{next}$ 分别表示当前和下一 RTT 的拥塞窗

*)国家自然科学基金重大研究计划项目(No. 90104012)和 863 计划(No. 2002AA121061)资助。岳 鹏 博士研究生。

口($cwnd$)的大小；$BaseRTT$ 表示所有已测量 RTT 的最小值；$0<\alpha<\beta$ 是两个门限，经验值分别为 1 和 3；$Diff=UExpected-UActual$，即期望速率与实际速率的差值；而期望速率和实际速率分别由下式确定：

$$UExpected=\frac{cwnd_{cur}}{BaseRTT},UActual=\frac{cwnd_{cur}}{RTT} \quad (2)$$

Vegas 的主要思想是：为了获得高的吞吐量和避免分组丢失，通过(1)式保持 Vegas 连接期望的积压分组的个数介于 α 和 β 之间。将(1)式中的条件展开，并结合(2)式，有：

$$Diff\times BaseRTT=cwnd_{cur}=-\frac{cwnd_{cur}}{RTT}\times BaseRTT \quad (3)$$

事实上，(3)式右边第一项为当前 RTT 内 cwnd 大小，第二项中$\frac{cwnd_{cur}}{RTT}$为当前 RTT 内 Vegas 连接获得的可用带宽的估计值，而 $BaseRTT$ 为 RTPT 的估计值，所以第二项可近似地看成 Vegas 连接的带宽时延积，反映通路容纳分组的能力，故(3)式左边近似为 Vegas 连接期望的积压分组数。为了保持 Vegas 连接期望的积压分组数介于 α 和 β 之间，当带宽时延积变化导致 $Diff\times BaseRTT>\beta$ 或 $<\alpha$ 时，cwnd 就必须做出相应的调整。在 $BaseRTT$ 不变的前提下，影响带宽时延积的因子只有 RTT。Vegas 通过对 RTT 的测量，根据 RTT 所反映的网络状况就能正确地对拥塞窗口进行调整。但是在因特网中，采用 IP 承载 TCP 业务。IP 的最大特点就是可以根据网络的状况动态调整路由，所以 IP 分组重选路的情况时有发生。Vegas 连接的 RTPT 经常因 IP 层重选路而发生改变，因此带宽时延积的变化很有可能是 IP 重选路引起的。此时 Vegas 会将 RTT 的变化误解释为瓶颈链路的拥塞状况发生变化，从而对拥塞窗口进行错误的调整，导致吞吐量劣化。下面分两种情况讨论。

2.1 重选路后 Vegas 连接的分组经历更短的 RTPT

在这种情况下，一旦测得分组的 RTT 小于 $BaseRTT$，则 Vegas 将会对 $BaseRTT$ 进行更新，即用小的 RTT 代替 $BaseRTT$。文[8]指出，Vegas 连接的发送速率在窗口不变和没有重传超时(RTO)情况下近似地与 RTT 成反比，因而在 $BaseRTT$ 更新前后，Vegas 连接的发送速率将会出现跃升。其跃升程度与更新前后 $BaseRTT$ 减少程度成正比，即更新后的 $BaseRTT$ 减小得越多，Vegas 连接的发送速率跃升程度越大。由(3)式，为了保持 Vegas 连接期望的积压分组数介于 α 和 β 之间，cwnd 必然增加。这将进一步加剧发送速率跃升的程度。发送速率跃升将引起数据流量增大，引起不必要的链路拥塞，严重的时候将导致分组丢失。尽管这种速率跃升所引起的链路拥塞或者可能的分组丢失将会触发 Vegas 的拥塞避免和快速恢复机制，从而减少 cwnd，抑制连接的发送速率，但是由速率跃升引起的瓶颈链路业务负载短暂增加将会影响瓶颈链路中其他 TCP 连接的吞吐量。因此，为了减少重选路后连接速率的大幅波动，笔者建议当最新测得的 RTT 小于 $BaseRTT$ 时，用下式代替(1)式中的第一项对 cwnd 进行调整，即

$$cwnd_{next}=cwnd_{cur}\times\frac{RTT}{BaseRTT}+1 \quad (4)$$

(4)式的作用是：通过调整拥塞窗口，保持 Vegas 连接的发送速率在更新 $BaseRTT$ 时基本不变，从而抑制发送速率跃升对网络的影响。

2.2 重选路后 Vegas 连接的分组经历更长的 RTPT

在这种情况下，由于 Vegas 连接不能区分 RTT 的增长是因网络拥塞还是由重选路引起，因此 Vegas 将 RTT 的增大一律解释成瓶颈链路即将发生拥塞。由(1)式第二项，Vegas 将不断地减小 cwnd，直到(1)式第三项满足为止。从而 Vegas 连接的吞吐量将会限制在一个较低的水平，即导致吞吐量劣化。然而事实上，RTPT 的增大将使带宽时延积增加，由(3)式，为了保持连接期望的积压分组的个数介于 α 和 β 之间，Vegas 应该增加 cwnd。得到这两个截然相反结果的根本原因是 Vegas 没有任何机制对重选路后的 $BaseRTT$ 进行更新。也就是说，Vegas 无法用重选路后大的 RTPT 更新 $BaseRTT$，仍然采用重选路之前的 $BaseRTT$ 作为基准对 cwnd 进行调整。

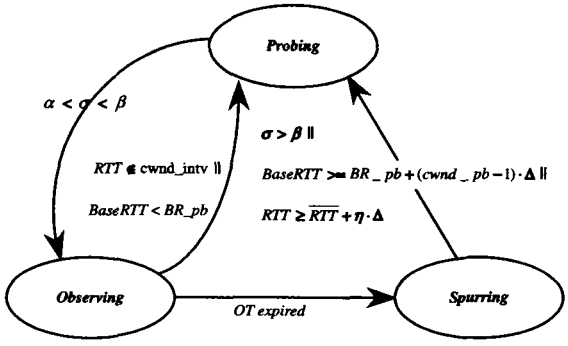
比较上述两种情况，尽管 2.1 会造成 Vegas 连接的发送速率发生跃升，引起瓶颈链路的业务负载激增而可能导致拥塞，但是 Vegas 算法能够对其发送速率进行有效的抑制，因而由 2.1 引起的瓶颈链路拥塞是暂时的，不会对 Vegas 连接的吞吐量造成长期的影响。然而 2.2 导致的 Vegas 连接吞吐量劣化却不能由 Vegas 窗口调整机制本身解决。如果 Vegas 连接重选路后一直经历较长的 RTPT，则该连接的吞吐量将一直维持在较低的水平，可见 2.2 导致的吞吐量劣化是长期的。因此 Vegas 重选路问题主要解决的是如何在 2.2 发生的情况下，对 Vegas 连接的吞吐量进行有效的恢复。为此，下一节我们将引入“主动激励”机制以期解决 2.2 情况下 Vegas 重选路问题。

3 主动激励(Active Spurring)机制

上一节着重讨论了 Vegas 重选路问题，指出其危害并分析了产生的根本原因。可见，如何使源具有区分重选路和通常链路拥塞，并能合理地调整拥塞窗口的能力，是解决 Vegas 重选路问题(如没有特别指明，以下 Vegas 重选路问题特指 2.2 情况。)的关键。重选路可以引起 RTT 突然增大、cwnd 持续减小以及 Vegas 连接吞吐量持续下降的现象。但是研究发现，当新的连接接入或者与其竞争带宽的其他连接的速率发生变化时，以上的情况同样会出现。因此，当采用 RTT，cwnd 或者吞吐量的变化作为判断重选路的依据时，必须设计相应的辅助技术(例如设置合适的判决规则)以去除其他因素的干扰。但是，这样的辅助技术往往需要改动基本的 Vegas 窗口调整机制，且参数配置起来非常复杂。

文[9]分析了 Vegas 的稳定性，指出当 Vegas 连接达到稳定时，其拥塞窗口将稳定在某个平衡点上下波动。一旦平衡被打破(比如 RTT 增大)，Vegas 会不断尝试调整 cwnd，使 cwnd 要么重新稳定到之前的平衡点(表示此时网络又恢复到以前的状态)，要么稳定到一个新的平衡点上下波动(表示此时网络处于一个新的状态)。主动激励机制的基本思想源于以上的结论，即当 cwnd 稳定在某个平衡点上时，源端主动地增加 $BaseRTT$，以打破这种平衡，激励 Vegas 进行窗口调整。通过 Vegas 自身的窗口调整机制使 cwnd 达到一个新的平衡，从而对重选路后连接的吞吐量进行恢复。下面对主动激励机制进行细化，笔者将主动激励机制划分成探测(Probing)、观察(Observing)和激励(Spurring)3 个状态。状态间转移和各状态中执行的算法伪码分别见图 1(a)和(b)，图中

符号含义见表 1。



(a) 状态转移图

Probing 状态：
每 RTT 执行：
if $\sigma<\alpha \| \sigma>\beta$
保持在 *Probing* 状态
else{
cwnd_pb=cwnd;
$BR\_pb = BaseRTT$
转移到 *Observing* 状态[注]
}
注：所有状态转移发生在下一个 RTT。
	Observing 状态：
	每 RTT 执行：
	if OT 结束{
	计算 $\overline{RTT}$ 和 Δ；
	$\eta = \overline{RTT} / BaseRTT$
	$BaseRTT = BaseRTT + \Delta$
	转移到 *Spurring* 状态
	}
	else if $RTT \notin$ cwnd_intvl ‖
	$BaseRTT < BR\_pb$
	转移到 *Probing* 状态
	else
	保持在 *Observing* 状态
		Spurring 状态：
		每 RTT 执行：
		if $\sigma>\beta \| RTT \geq \overline{RTT} + \eta \cdot \Delta$ {
		$BaseRTT = BaseRTT - \Delta$
		转移到 *Probing* 状态}
		else { $\eta = RTT / BaseRTT$
		$BaseRTT = BaseRTT + \Delta$
		if
		$BaseRTT >= BR\_pb + cwnd\_pb \cdot \Delta$ {
		$BaseRTT = \overline{RTT}$
		转移到 *Probing* 状态
		}
		else 保持在 *Spurring* 状态
		}

(b) 算法伪码

图 1　主动激励机制状态转移图及算法伪码

表 1　图 1 中符号的含义

σ	Vegas 连接期望的积压分组数，即：$\sigma = Diff \times BaseRTT$
OT	观察时间(Observation Time)，OT 设为 4 个 RTT 时间
$\overline{RTT}$	OT 内最小 RTT
cwnd_pb	Probing 状态下，cwnd(单位是 segment)的大小
BR_pb	Probing 状态下，*BaseRTT* 的大小
cwnd_intvl	cwnd 波动区间，cwnd_intvl=[cwnd_pb−1，cwnd_pb+1]
Δ	*BaseRTT* 的增加步长，$\Delta \triangleq \frac{\overline{RTT} - BR\_pb}{cwnd\_pb}$
η	时延增加因子

上述算法中，η 是一个非常重要的参数，它决定 *BaseRTT* 的更新。下面着重对 η 进行讨论，考虑不等式：

$$\frac{x}{y} > \frac{x+a}{y+\zeta \cdot a} \tag{5}$$

式中 x、y 和 a 均为正数且已知。显然，不等式(5)成立的条件是 $\zeta > \frac{y}{x}$。分别用 *BaseRTT*、RTT、Δ 和 η 代替(5)中的 x、y、a 和 ζ。当 *BaseRTT* 增加 Δ 后，若引起下一次测量的 RTT 的增加量大于 $\eta \cdot \Delta$，则由(5)式，有

$$\frac{BaseRTT}{RTT} > \frac{BaesRTT+\Delta}{RTT+\eta \cdot \Delta}$$

从而，$1 - \frac{BaseRTT}{RTT} < 1 - \frac{BaseRTT+\Delta}{RTT+\eta \cdot \Delta}$ (6)

在 Spurring 状态中，转移条件 $\sigma > \beta$ 不成立，表明上一次和当前 *RTT* 的拥塞窗口满足：

$$cwnd_{last} \leqslant cwnd_{cur} \tag{7}$$

由(6)和(7)式，可得：

$$cwnd_{last} \cdot (1 - \frac{BaseRTT}{RTT}) < cwnd_{cur} \cdot (1 - \frac{BaseRTT+\Delta}{RTT+\eta \cdot \Delta}) \tag{8}$$

由(3)式可知，(8)式不等号两侧的表达式就是 Vegas 连接期望的积压分组数。(8)式表明，当 RTT 的增加量大于 $\eta \cdot \Delta$，Vegas 连接期望的积压分组数将增加，预示着瓶颈链路有拥塞的趋势。显然，这是由于上一 RTT 内增加 *BaseRTT* 引起的，因而主动激励机制将停止增加 *BaseRTT*，并用 *BaseRTT*−Δ(上一 RTT 内的 *BaseRTT*)对 *BaseRTT* 进行更新。

此外，在 Spurring 状态中，若转移条件 $BaseRTT \geq BR\_pb + cwnd\_pb \cdot \Delta$ 满足，则意味着 *BaseRTT* 超过 $\overline{RTT}$。因为 $\overline{RTT}$ 是 Observing 和 Spurring 状态所有已测量 RTT 的最小值，所以根据 BaseRTT 的定义，当 *BaseRTT* 不断累加超过 $\overline{RTT}$ 时，应该用 $\overline{RTT}$ 对 *BaseRTT* 进行更新，并在下一 RTT 转移到 Probing 状态。

以上算法的设计过程中，并没有对原有的 Vegas 窗口调整算法进行任何修改，这表明引入主动激励机制的算法开销是很小的。因而在实际的应用中，可以将该机制设计为一个独立的模块内嵌于 Vegas 或其增强算法中，从而可以容易地对当前的 Vegas 或其增强算法进行扩充。

4　性能仿真和讨论

本文利用 ns2[10] 仿真平台对主动激励机制的性能进行仿真，图 2 给出了仿真拓扑结构和一些基本的仿真参数。图中

发送端 A 和 B 分别向接收端 A 和 B 发送数据。路由器缓冲区大小为 15 个分组，采用队尾丢弃策略。为了模拟重选路，路由器 2 与接收端 A 之间接入链路时延 x 可变。仿真中假设所有业务流有足够多的发送数据，数据分组大小均为 1000 字节。此外，为了减少暂态过程对仿真结果的影响，本文只对 10s 后的仿真数据进行统计，所有仿真持续 100s。

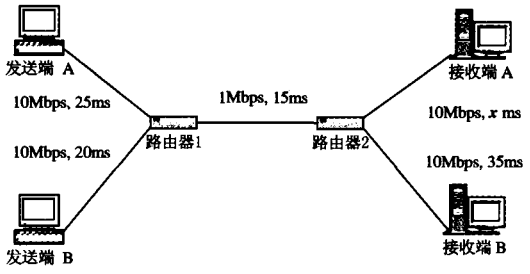
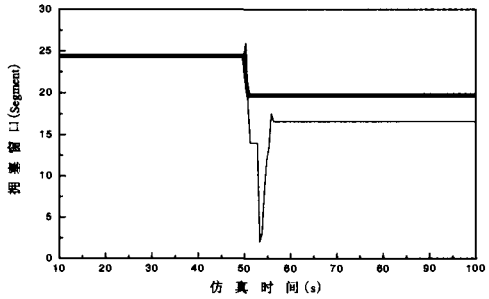


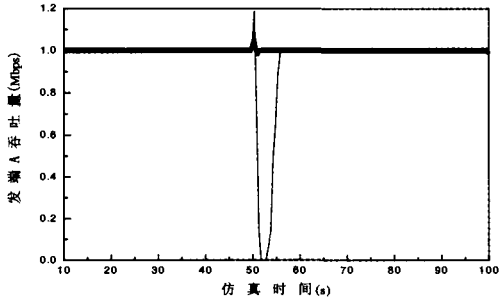
图 2　仿真拓扑和参数

4.1　重选路后连接经历更短 RTPT 的情况

第 2 节指出，在这种情况下，Vegas 连接的发送速率将会出现跃升，引起短暂的数据突发，本节通过仿真进一步进行验证。图 2 中，发送端 A 配置成 FTP/Vegas 源，50s 之前 x 为 40ms，50s 时重选路，x 变为 10ms。



(a) 拥塞窗口变化情况



(b) 发端吞吐量变化情况

图 3　重选路后更短的 RTPT 引起的拥塞窗口和吞吐量的变化情况

虽然重选路后更短的 RTPT 可能会造成连接吞吐量的大幅波动，但是传统的 Vegas 窗口调整机制能够有效地恢复吞吐量直至重选路之前的水平。表明这种情况仅仅会导致连接吞吐量暂时的劣化，其对 Vegas 连接性能的影响是有限的。

4.2　重选路后连接经历更长 RTPT 的情况

叙述方便起见，把采用主动激励机制的 Vegas 记为 Vegas_AS。为了突出重选路问题对 Vegas 连接吞吐量的影响，本节首先分别考察单个 Vegas 和 Vegas_AS 连接在重选路前后拥塞窗口和对应吞吐量变化情况。图 2 中，发送端 A 分别配置成 FTP/Vegas 和 FTP/Vegas_AS 源，50s 之前 x 为 10ms，50s 时重选路，x 变为 30ms。

图 4(a)显示当 50s，x 从 10ms 增加到 30ms 时，传统 Vegas 源的拥塞窗口从 16 下降到 13 并保持不变。图 4(b)中对应的吞吐量从 1Mbps 下降到 700kbps 左右并保持不变，此时的瓶颈链路显然没有拥塞。但是传统的 Vegas 窗口调整机制无法增加拥塞窗口，导致 Vegas 连接的吞吐量下降大约 300kbps。定义 INC、DEG 分别表示 RTPT 变化和连接吞吐量劣化程度，即：

$$INC=\left(\frac{PD'-PD}{PD}\right)\times 100\%,$$
$$DEG=\left(\frac{Thght(PD)}{Thght(PD')}-1\right)\times 100\% \qquad (9)$$

图 3 中粗细实线分别是采用(4)式的 Vegas 源和传统 Vegas 源的仿真结果。图 3(a)显示在 50s，链路时延 x 从 40ms 减少到 10ms 时，传统 Vegas 源的拥塞窗口急剧下降。图 3(b)中对应的吞吐量在经历短暂的跃升后急剧下降，并导致 Vegas 重新进入慢启动状态，这表明重选路引起的大量突发数据造成路由器中缓存溢出。尽管 60s 左右，Vegas 经过窗口调整重新进入拥塞避免阶段，对应的吞吐量也恢复到重选路前的水平，但是重选路前后吞吐量的大幅波动显然对 Vegas 连接的稳定性带来不利的影响，这种情况应该避免。然而采用(4)式的 Vegas 源在重选路后，拥塞窗口很快趋于稳定，对应的吞吐量基本保持不变。可见(4)式能够有效地避免重选路后更短的 RTPT 导致的吞吐量大幅波动的问题。另外，图 3(a)中传统的和采用(4)式的 Vegas 源在重选路后拥塞窗口分别稳定在大约 17 和 20 附近，这种情况的发生是两者在重选路后不同 BaseRTT 的更新造成的。传统的 Vegas 源在重选路后经历路由器缓存中积压分组排空的过程(Vegas 连接进入慢启动状态导致发送速率下降)，此时源端更新的 $BaseRTT$ 近似与 RTPT 相等。而采用(4)式的 Vegas 源由于不会出现积压分组排空的现象，源端更新的 $BaseRTT$ 较之实际的 RTPT 大，因此在获得相同吞吐量的情况下，BaseRTT 越小则对应拥塞窗口就越小。

式中 PD、PD' 分别表示重选路前后的 RTPT。Thght(·)表示在"·"时延下，Vegas 连接的吞吐量。显然 DEG 越大，表明 Vegas 连接吞吐量劣化得越严重。分别对 INC(%)为 20、40、60、80 和 100 时重选路前后 Vegas 连接的吞吐量进行统计，则 DEG 随 INC 变化的情况如表 2。

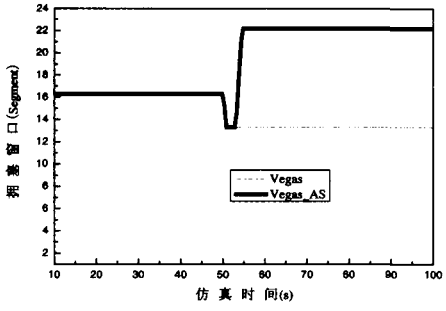
表 2　DEG 随 INC 变化的情况

重选路前(ms)	重选路后(ms)	INC(%)	DEG(%)
10	20	20	10
	30	40	43.8
	40	60	120.5
	50	80	212.5
	60	100	267.6

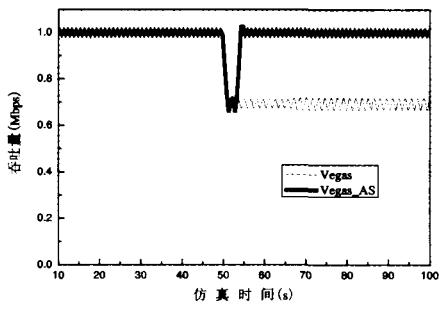
表 2 显示 DEG 随着 INC 的增加而快速增长，这说明重选路后 RTPT 越长，对 Vegas 连接吞吐量的影响越大，严重时将导致 Vegas 算法失效。

另一方面，图 4(a)显示在相同的状况下，尽管 Vegas_AS 源的拥塞窗口也从 16 降到 13，但是随后在主动激励机制的作用下，拥塞窗口不断增加，最终稳定在 22 附近。反映在图 4(b)，连接的吞吐量在 50s 前后经历短暂的吞吐量下降，但是很快恢复到重选路之前的水平。

下面考察当与 Reno 连接共享瓶颈链路的情况下，重选路前后 Reno 连接对 Vegas 和 Vegas_AS 连接吞吐量的影响。将上面仿真模型中发送端 B 配置成 FTP/Reno 源，其他配置保持不变。图 5(a)是 Vegas 和 Reno 共存的仿真结果，在重选路(50s)之前，Vegas 和 Reno 连接能够公平地分享瓶颈链路带宽。然而重选路后，Vegas 连接的吞吐量下降到大约 300kbps，而 Reno 连接的吞吐量则上升到大约 700kbps。表明重选路后，Vegas 连接遭遇不公平，吞吐量进一步劣化。图 5(b)为采用 Vegas_AS 的仿真结果，可以看到除了重选路前后 Vegas 连接吞吐量发生短暂的下降外，其他时间内 Vegas 连接均能和 Reno 公平地分享瓶颈链路带宽，表明主动激励机制能够有效地减少 Vegas 连接在重选路后与 Reno 连接竞争带宽时遭遇的不公平，提高 Vegas 连接的吞吐量。

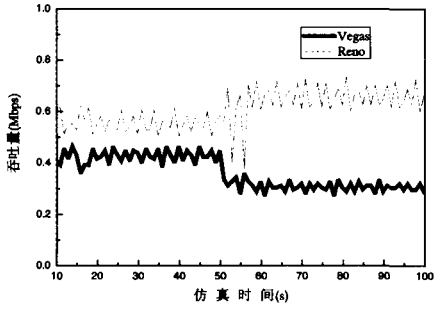


(a) 拥塞窗口变化情况

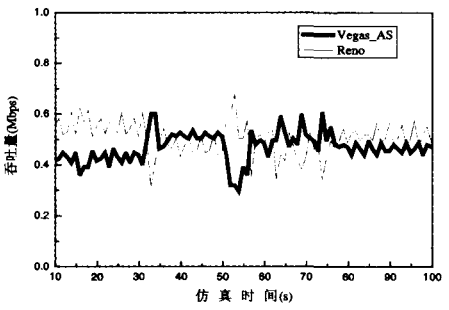


(b) 发端吞吐量变化情况

图 4　重选路后更长的 RTPT 引起的拥塞窗口和吞吐量的变化情况



(a) Vegas 与 Reno 共存



(b) Vegas_AS 与 Reno 共存

图 5　重选路前后 Reno 连接对 Vegas 和 Vegas_AS 吞吐量的影响

结束语　本文首先分析讨论了 Vegas 重选路问题，该问题会引起 Vegas 连接吞吐量劣化。更为严重的是，与其它引起 Vegas 连接吞吐量暂时劣化的问题不同，重选路问题对 Vegas 连接的吞吐量影响很有可能是长期的。其次，根据分析结果，提出一种称为“主动激励”的新机制，该机制能够有效地解决 Vegas 重选路问题并可作为一个独立模块内嵌到 Vegas 或其增强算法中，从而可以容易地对当前 Vegas 或其增强算法进行有效的扩充。仿真结果表明重选路后，“主动激励”机制能够对 Vegas 连接的吞吐量进行有效的恢复。

参 考 文 献

1　Brakmo L S, Peterson L L. TCP vegas: End to end congestion avoidance on a global Internet. IEEE J Select Areas Commun, 1995, 13(8): 1465～1480

2　Ahn J S, Danzig P B, Liu Z, et al. Evaluation with TCP Vegas: Emulation and experiment. In: ACMSIGCOMM 1995[C], Cambridge, MA: ACM Press, Aug. 1995. 185～205

3　Mo J, La R J, Anantharam V, et al. Analysis and comparison of TCP Reno and Vegas[A]. In: Proc. INFOCOM '99[C]. New York: IEEE, 1999, 3: 1556～1563

4　Chen J R, Chen Y C. Vegas Plus: Improving the Service Fairness [J]. IEEE Commun Lett, 2000, 4(5): 176～178

5　Hengartner U, Bolliger J, Gross Th. TCP Vegas Revisited[A]. In: Proc. of IEEE INFOCOM'2000[C], Tel-Aviv, Israel: IEEE Press, March 2000. 1546～1555

6　Fu C P, Liew S C. A remedy for performance degradation of TCP vegas in asymmetric networks[J]. IEEE Commun Lett, 2003, 7(1): 42～44

7　Low S H, Peterson L L, Wang L. Understanding TCP Vegas: A duality model. In: SIGMETRICS/Performance01 [A]. Cambridge, MA: ACM Press, 2001. 226～235

8　Samios C, Vernon M K. Modeling the Throughput of TCP Vegas [A]. In: Proc. of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems [C], San Diego, CA, USA: ACM Press, 2003. 71～81

9　Choe H, Low S H. Stabilized Vegas[A]. In: Proc. of IEEE INFOCOM'03[C], San Francisco CA, USA: IEEE Press, 2003, 3: 2290～2300

10　UC Berkeley, LBL, USC/ISI, Xerox PARC. The Network Simulator: ns-2. [EB/OL]. Available: http://www.isi.edu/nsnam/ns. 2004-11-1