

FAST TCP 快速收敛算法研究^①陈晓龙^{②*} 章云* 田义强** 刘治*

(* 广东工业大学自动化学院 广州 510090)

(** 茂名学院计算机与电子信息学院 茂名 525000)

摘要 针对 FAST TCP 静态选择协议参数无法同时保证网络小的排队时延和快速收敛的缺陷,通过理论分析和 NS-2 仿真探讨了 FAST TCP 协议各参数和收敛速度的关系。在此基础上,将各连接留在链路缓冲区的个数作为慢启动阈值,提出了一种根据各连接协议参数确定慢启动阈值,根据各连接源端局部信息协同估计网络运行状态,调整控制律增益参数的改进方法。NS-2 仿真结果表明该方法保证系统处于稳定状态和具有小的排队时延,同时提高了系统收敛到平衡点的速度。

关键词 排队时延,快速收敛,慢启动阈值,增益参数,协同

0 引言

FAST TCP^[1-5](简称 FAST)是加州理工学院网络实验室 Low 等人针对高速、长延迟、大容量的下一代高性能网络提出的一种新型传输控制协议。FAST 协议将拥塞控制器建立在各连接源端,采用估测的排队时延作为拥塞反馈信号,其基于平衡的设计思想使其稳定性、快速响应、吞吐量和公平性等性能指标要优于 TCP Reno、HSTCP 和 STCP 等网络传输控制协议^[1],但其存在如何选择合适的协议参数,使 FAST 协议表现出良好性能的公开问题^[2,5,6]。

由文献[1,2,5]和分析 FAST 代码(<http://www.cubinlab.ee.Unimelb.edu.au/ns2fasttcp/>)可知,目前在建立 FAST 连接时,是采用静态表映射方法初始化 FAST 的协议参数 α 、控制律增益参数 γ 和慢启动阈值 m 。其中协议参数 α 是 FAST 达到平衡态时,期望留在链路缓冲区数据包的个数^[1,5,6],所以在系统稳定的前提条件下,协议参数 α 越小,系统平衡时链路的排队时延越小^[1,4],但文献[1,5,6]表明若选择较小的协议参数 α ,系统收敛到平衡点的速度就会变慢,从而降低网络的利用率。因此现有的静态选择 FAST 参数的方法是无法同时保证系统获得较小的排队时延和较快的收敛速度。文献[6]利用分布在网络中的测量设施监控主干链路,再根据链路性能采用模糊控制技术指导各 FAST 连接源端选

择适当的协议参数 α , 稳定排队时延,但该算法没有研究如何提高系统收敛速度。

为了克服上述缺陷,本文在确保系统稳定选择尽可能小的 α 的条件下,提出了各连接根据源端局部信息,自适应调整慢启动阈值 m 和控制律增益参数 γ 的改进算法,在保证系统处于稳定和小的排队时延的同时,提高系统收敛到平衡点的速度。

1 FAST 拥塞控制算法

1.1 基本符号

如图 1 所示,假设哑铃网络系统中包含 N 条具有相同链路属性的 FAST 连接(以下简称连接),共享唯一瓶颈链路 l ,定义连接集合 $I = \{1, 2, \dots, N\}$ 。假设链路端缓冲区足够大,采用丢尾的队列管理算法。

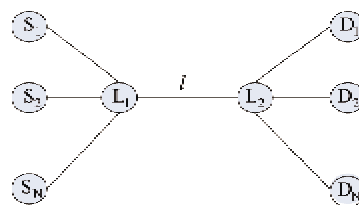


图 1 哑铃型网络拓扑

FAST 系统的性能指标定义如下:

δ : 超调量,指各连接获得的传输流量最大值减

① 国家自然科学基金(60604006),教育部博士点基金(20070562005)和广东省联合基金(U0735005)资助项目。

② 男,1971 年生,博士生,副教授;研究方向:网络拥塞控制,非线性控制;联系人,E-mail: xlycd@126.com

(收稿日期:2009-05-11)

去稳态值,与稳态值之比的百分数;

st : 收敛时间,指新建连接到达并保持在稳态值附近 $\pm \epsilon$ 内所需的最短时间(s); ϵ 为一给定的较小正数。

对于任给的第 $i(i \in I)$ 连接有:

$W_i(t)$: 源端发送窗口大小(packet(包));

d_i : 传播时延(s),假设各连接都获得准确的传播时延,不存在公平性问题^[7];

$q_i(t)$: 排队时延(s);

$D_i(t)$: 往返时延, $D_i = d_i + q_i(s)$;

$x_i(t) = W_i(t)/D_i(t)$: 传输流量(packet/s);

$z_i(t) = x_i(t) \cdot q_i(t)$: 实际留在链路缓冲区数据包的个数^[1,6](packet);

α_i : 协议参数(packet),平衡时,期望留在链路缓冲区数据包的个数;

m_i : 慢启动阈值(s);

γ_i : 控制律增益参数;

$y_i(t) = W_i(t)/d_i$: 标准化窗口大小,单位时间内留在缓冲区的包及离开包的和^[1];

T : 窗口更新周期(s)。

对于链路 l :

C : 带宽(packet/s);

$b_l(t)$: 队列长度(packet);

$q_l(t)$: 排队时延,其中 $q_l = b_l/c_l(s)$ 。

1.2 FAST 拥塞控制算法

由文献[1,2,5]和分析 FAST 代码知,TCP 连接建立时其协议参数 α_i 根据链路带宽选择,慢启动阈值 $m_i = 0.00075(s)$,控制律增益参数 $\gamma_i = 0.5$ 。

其发送窗口按照算法 1 进行更新^[1-5]。

算法 1:

每收到一个确认帧 {

If $q_i(t) < m_i$

$$W_i(t) = W_i(t) + 1 \quad (1)$$

式(1)//执行慢启动窗口更新算法;

Else if $z_i(t) \neq \alpha_i$

每隔一周期 T // T 为窗口更新周期

$$W_i(t+T) = (1 - \gamma_i) \cdot W_i(t) + \gamma_i \left(\frac{d_i}{d_i + q_i(t)} \cdot W_i(t) + \alpha_i \right) \quad (2)$$

式(2) //执行 FAST 窗口更新算法;

}

由算法 1 的描述可知,FAST 协议的收敛速度与

参数 α_i, m_i, γ_i 有关^[1-5],其中参数 α_i 决定了 FAST 系统平衡点位置, α_i 越小,收敛速度越慢;参数 m_i 决定了源端何时由慢启动算法转为执行 FAST 窗口更新算法,参数 γ_i 决定每次执行 FAST 算法时,窗口变化的比例大小,因此这两个参数的选择都会影响收敛速度。因此本文希望在确保系统稳定的条件下选择较小的参数 α_i (较小的排队时延),改进现有的静态选择参数 m_i, γ_i 的方法,通过各连接源端能够获得局部信息,以动态调整参数 m_i, γ_i , 提高 FAST 协议收敛速度,并保持较小的排队时延。

2 慢启动阈值对协议收敛速度的分析及改进

2.1 排队时延比

由算法 1 可知,慢启动阈值的大小决定了慢启动算法何时转到 FAST 窗口更新算法。

首先定义图 2 所示各变量的含义:

$T_1 = m_i$, 表示执行慢启动算法排队时延段;

$T_2 = N \cdot \alpha_i / C - m_i$, 表示由慢启动结束到系统平衡时执行 FAST 窗口更新算法排队时延段;

n_i : 当 $q_i(t) = m_i$ 连接 i 留在缓冲区的包的个数。

由文献[1,5,6]可得排队时延比:

$$\frac{T_1}{T_2} = \frac{1}{N \cdot \alpha_i / (C \cdot m_i) - 1} = \frac{1}{\alpha_i / n_i - 1} \quad (3)$$

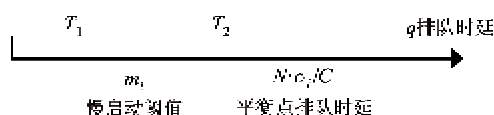


图 2 排队时延比

式(3)表示 FAST 连接从建立到平衡态时执行慢启动算法和执行 FAST 算法排队时延比。

由式(3)知,当相关参数 N, α_i, m_i, C 发生变化时,式(3)值变化,表明慢启动算法执行相对时间发生变化。

2.2 仿真验证排队时延比与 FAST 系统收敛速度的关系

实验 1: 连接数 N 设置为 3,仿真时间为 10(s),每个数据包的大小设置为 1000 个字节。设置链路传输带宽 C 为 96Mb/s (即 $C = 12000$ packet/s)。FAST 取静态映射参数: $\alpha_i = 200$, $m_i = 0.00075(s)$, $\gamma_i = 0.5$ 。FAST 源端的传输时延设置为 $d_i = 0.1(s)$ 。仿真结果如图 3 标记 1 曲线所示(注意图中标

记 1、2、3 分别表示实验 1、2、3 各连接平均传输流量)。由式(3)得 $T_1/T_2 = 0.01522$,从图 3 知该系统收敛时间 $st = 2(s)$,超调量 $\delta = 25\%$ 。系统有较快的收敛速度和较小的超调量。

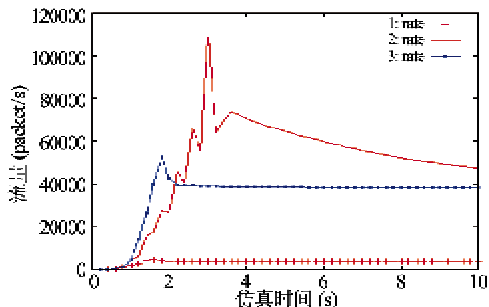


图 3 实验 1、2、3 各连接平均传输流量

实验 2:链路传输带宽 C 变为 960Mb/s (即 $C = 1\,20000\text{packet/s}$),其它参数同实验 1。仿真结果如

图 3 标记 2 曲线所示。

由式(3)得 $T_1/T_2 = 0.1764$,说明相对实验 1 来说各连接在慢启动阶段所占比例较大,故在图 3 可看到该系统收敛时间 $st > 10(s)$,超调量 $\delta > 200\%$ 。

由式(3)知,在带宽增大时,可通过增大协议协议参数 α_i 来保证式(3)比值不变。

实验 3:增大协议参数 $\alpha_i = 2000$,其它参数同实验 2。仿真结果如图 3 标记 3 曲线所示。由式(3)得 $T_1/T_2 = 0.01522$ 。故在图 3 中可看到,与实验 2 相比增大 α_i 后该系统收敛时间 $st = 2(s)$,超调量 $\delta = 25\%$ 。但由文献[1]知实验 2 平衡态时链路排队时延为 $0.005s$,而实验 3 的瓶颈排队时延增大为 $0.05s$ 。

实验 4:在确保网络处于稳定状态下,当网络和 FAST 协议相关参数取表 1 中不同值时,系统超调量、收敛时间的仿真值如表 1 所示。

表 1 慢启动阈值和系统超调量、收敛速度关系

序号	α_i	m_i	C (packet/s)	N	$d(s)$	T_1/T_2	n_i/α_i	δ	$st(s)$
1	400	0.00075	12000	1	0.1	0.023	0.0225	12.4%	2.2
2	400	0.0075	12000	1	0.1	0.29	0.225	61.5%	2.2
3	400	0.0015	12000	1	0.1	0.047	0.045	14.5%	2.1
4	400	0.003	12000	1	0.1	0.099	0.09	8%	1.7
5	400	0.004	12000	1	0.1	0.136	0.12	27.5%	1.9
6	400	0.006	12000	1	0.1	0.219	0.18	43.4%	2.0
7	800	0.00075	12000	1	0.1	0.0113	0.01125	41.6%	1.9
8	800	0.0075	12000	1	0.1	0.126	0.1125	71.5%	2.2
9	800	0.0015	12000	1	0.1	0.0230	0.0225	23.5%	1.95
10	800	0.003	12000	1	0.1	0.047	0.045	27.6%	1.85
11	800	0.005	12000	1	0.1	0.081	0.075	21.3%	1.8
12	800	0.006	12000	1	0.1	0.099	0.09	20%	1.75
13	800	0.007	12000	1	0.1	0.118	0.105	66%	2.1
14	1000	0.00075	12000	1	0.15	0.0091	0.009	21.5%	3.05
15	1000	0.0075	12000	1	0.15	0.099	0.09	20.5%	2.75
16	1000	0.005	12000	1	0.15	0.063	0.06	33.3%	3.3
17	1000	0.009	12000	1	0.15	0.121	0.108	29.1%	2.8

由实验 1、2、3、4 可知:

(1)当网络参数发生变化时,式(3)的排队时延比值会发生变化,该比值的变化会影响 FAST 协议的性能(超调量和收敛时间)。

(2)相同条件下增大协议参数 α_i 提高了系统收敛速度,但也增大了链路排队时延。

(3)当取不同的慢启动阈值,网络呈现出不同的性能。由表 1 的第 1、7、14 组数据和第 4、12、15 组数据比较知,目前采用的慢启动阈值默认取值并不是

最优的。由第 4、12、15 组数据知,当取相应的慢启动阈值,使得 $T_1/T_2 \in (0.09, 0.1)$ 或 $n_i/\alpha_i \in (0.09, 0.1)$ 区间时,网络呈现出较小的超调量和较快的收敛速度。

2.3 算法改进

由上述实验和式(3)知,现有的算法采用排队时延参数作为慢启动阈值,当网络相关参数发生改变时, T_1/T_2 值会发生变化,从而使得 FAST 协议性能会产生变化,因此本文定义各连接留在缓冲区个数

参数 $l_i = \alpha_i / G$ ($G = 10$) 作为各连接的慢启动阈值, 这样即使带宽、连接数等其他参数发生变化, 仍可确保 T_1/T_2 值始终保持在 $(0.09, 0.1)$ 区间内不会改变, 从而确保良好的网络性能。具体改进算法如下:

算法 2: Flag = 0; $l_i = \alpha_i / G$;

每收到一个确认帧 {

If ($z_i(t) < l_i$ && Flag = 0) // $z_i(t)$: 第 i 连接实际留在链路缓冲区数据包的个数^[1](packet/s);

$W_i(t+1) = W_i(t) + 1$ // 执行慢启动窗口更新算法;

Else if ($z_i(t) \neq \alpha_i$)

每隔一周期 T

{Flag = 1;

$W_i(t+T) = (1 - \gamma_i) \cdot W_i(t) + \gamma_i \left(\frac{d_i}{d_i + q_i(t)} \cdot W_i(t) + \alpha_i \right);$

// 执行 FAST 窗口更新算法; }

下面验证算法 2 的有效性。

实验 5: 参数设置同实验 2, 但源端采用改进的算法 2。仿真结果如图 4 标记 5 曲线所示(注意图中标记 2、5、6 分别表示实验 2、5、6 各连接平均传输流量)。从图 4 看出, FAST 系统收敛时间 $st = 6(s)$, 超调量 $\delta = 6\%$ 。同实验 2 相比, 算法 2 在不需要增加协议参数 α_i (即排队时延) 的同时, 大幅降低了系统的超调量和提高了系统收敛速度。

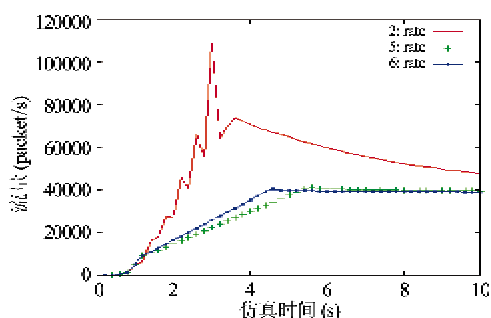


图 4 实验 2、5、6 各连接平均传输流量

3 控制律增益参数对协议收敛速度的分析及改进

文献[2,3]引入了包含自同步和积分链路的更为精确的 FAST 协议模型, 应用乃奎斯特稳定性判定方法给出在单瓶颈链路, 对于有界的往返时延, 当控制率增益参数 γ 的取值范围为 $(0, 0.93)$ 时, 系统渐进稳定。

首先分析当 $\gamma_i \in (0, 0.93)$ 时, 控制律增益参

数 γ_i 对 FAST 系统收敛速度的影响。

将式(2)两边均除以传播时延 d_i 得

$$y_i(t+T) = \left(1 - \frac{\gamma_i \cdot q(t)}{d_i + q(t)}\right) \cdot y_i(t) + \gamma_i \cdot \hat{\alpha}_i \quad (4)$$

其中 $\hat{\alpha}_i = \alpha_i / d_i$ 。

定理 1: 假设各 FAST 流的 γ_i 值相等, 且 $\gamma_i \in (0, 0.93)$, 假设 $Y(t) = \sum y_i(t)$, FAST 系统在退出慢启动和平衡点时大小分别为 $Y(0)$ 、 Y^* , 则对于任意小的正数 ϵ , 存在收敛时间 $st = \frac{\ln \frac{\epsilon}{|Y(0) - Y^*|}}{\ln(1 - \gamma_i)}$, 当 $t > st$ 时, 有 $|Y(t) - Y^*| < \epsilon$ 成立, 其中 $Y^* = c + \sum_i \hat{\alpha}_i$ 。

证明:

由(4)得

$$y_i(t+T) = (1 - \gamma_i) \cdot y_i(t) + \gamma_i \left(\frac{d_i(t) \cdot y_i(t)}{d_i + q(t)} + \hat{\alpha}_i \right) \quad (5)$$

由文献[1]知 $\sum_i \frac{d_i(t) \cdot y_i(t)}{d_i + q(t)} = C$, 对式(5)各连接源端求和得

$$Y(t+T) = (1 - \gamma_i) \cdot Y(t) + \gamma_i \cdot (c + \hat{\alpha}) \quad (6)$$

其中 $\hat{\alpha} = \sum_i \hat{\alpha}_i$ 。

把式(6)进行变形得:

$$Y(t+T) - (c + \hat{\alpha}) = (1 - \gamma_i) \cdot (Y(t) - (c + \hat{\alpha})) \quad (7)$$

式(7)表示首项为 $Y(0) - (c + \hat{\alpha})$, 公比为 $1 - \gamma_i$ 的等比数列, 这样对任意 $st = k \cdot T$ 满足

$$Y(st) = c + \hat{\alpha} + (Y(0) - (c + \hat{\alpha})) \cdot (1 - \gamma_i)^{st} \quad (8)$$

由式(8)可知 $Y(st)$ 收敛于平衡点 Y^* , 且 Y^* 满足式

$$Y^* = c + \hat{\alpha} \quad (9)$$

这样式(8)就可以表示为

$$Y(st) = Y^* + (Y(0) - Y^*) \cdot (1 - \gamma_i)^{st} \quad (10)$$

由式(10)得, 对于任意小的正数 ϵ , 当时间

$$t^3 \geq \frac{\ln \frac{\epsilon}{|Y(0) - Y^*|}}{\ln(1 - \gamma_i)} \quad (11)$$

时, 有 $|Y(t) - Y^*| \leq \epsilon$ 成立, 证毕。

注意到式(11)的分子为负数, 故控制律增益参数 γ_i 在满足系统稳定的情况下取值越大, 系统收敛到平衡点的时间越短。所以本文修改原 FAST 算法

静态设置增益参数 $\gamma_i = 0.5$ 的情况,根据各连接源端能够获得的局部信息的值 ($W_i(t), D_i(t), d_i, \alpha_i$), 协同判断网络运行状态 $diff_i$ (是否远离平衡点)^[8,9], 采用指数函数公式 $\gamma_i = 0.33 + 0.6 \cdot (e^{\kappa \cdot diff_i} - 1) / (e^{\kappa} - 1)$ ($\kappa < -1$), 确保 $\gamma \in (0.33, 0.93)$, 从而在保证 FAST 系统稳定的条件下, 动态调整控制律增益参数。

由上述指数函数公式可知, 当系统远离平衡点时 ($diff_i$ 值较大), 取较大的增益参数 γ_i , 使系统能够快速收敛到平衡点, 而当系统靠近平衡点时 ($diff_i$ 值较小), 为减小超调量和抖动取比较小增益参数 γ_i , 保证系统在平衡点附近抖动较小。算法 3 给出了同时改进慢启动阈值和控制律增益参数 FAST 窗口更新算法。

算法 3: Flag = 0; $l_i = \alpha_i / G$;

每收到一个确认帧 |

If ($z_i(t) < l_i$ && Flag = 0)

$W_i(t+1) = W_i(t) + 1$

Else if ($z_i(t) \neq \alpha_i$)

每隔一周期 T

{ Flag = 1;

$\eta_i = ABS(x_i(t) \cdot q_i(t) - \alpha_i)$;

$diff_i = \min(\eta_i / \alpha_i, 1)$;

$\gamma_i = 0.33 + 0.6 \cdot (e^{\kappa \cdot diff_i} - 1) / (e^{\kappa} - 1)$;

$W_i(t+1) = (1 - \gamma_i) \cdot W_i(t)$

$+ \gamma_i \left(\frac{d_i}{d_i + q_i(t)} \cdot W_i(t) + \alpha_i \right)$;

|

下面验证算法 3 的有效性。

实验 6: 参数设置同实验 2, 但源端采用改进算法 3, 算法 3 参数 $\kappa = -5$ 。仿真结果如图 4 标记 6 曲线所示。从图 4 看出该系统收敛时间 $st = 4$ (s), 超调量 $\delta = 5\%$ 。同实验 2、3、5 相比, 算法 3 在不需要增加协议参数 α_i (即排队时延) 的同时, 降低了系统的超调量和提高了系统收敛速度。

4 动态环境仿真

当链路存在连接动态到达和离开时, 改进的慢启动阈值和控制律增益参数 FAST 算法仍能在不需要增加协议参数 α_i (即排队时延) 的同时, 降低了系统的超调量和提高了系统收敛速度。

实验 7: 链路带宽 C 取 960Mb/s, $\alpha_i = 200$, $d_i = 0.1$ (s), 刚启动时建立 fast1 和 fast2 连接, 在 100s 时

建立 fast3 连接, 200s 时释放 fast2 连接。若采用没有改进的算法, 取 $m_i = 0.00075$ (s), $\gamma_i = 0.5$ 。仿真结果如图 5 所示 (图标标记 fast1: rate 表示为 fast1 连接获得的传输流量)。仿真结果表明系统刚启动时 FAST 系统收敛时间 $st > 20$ (s), 超调量 $\delta > 200\%$ 。当有新 FAST 流加入和离开时 FAST 系统收敛时间 $st > 20$ (s)。在上述相同条件下, 源端采用改进的算法 3, 不需要设置参数 m_i, γ_i (假设各 FAST 连接已获得准确的传播时延, 能够公平分配带宽), 仿真结果如图 6 所示。仿真结果表明无论是新连接建立或连接释放 FAST 系统收敛时间 $st = 4$ (s), 超调量 $\delta = 5\%$ 。

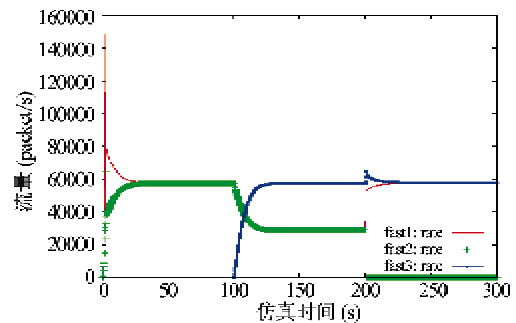


图 5 原 FAST 算法仿真结果

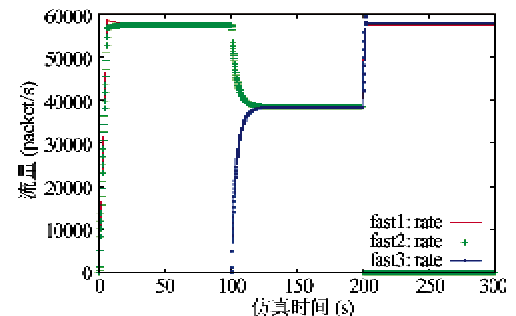


图 6 改进 FAST 算法后仿真结果

5 结论

本文将留在链路缓冲区个数作为慢启动阈值, 虽然与原算法采用排队时延作为慢启动阈值成比例关系, 但这种改进可确保排队时延比值在 (0.09, 0.1) 区间内, 不随着带宽、连接数等其他参数变化而发生改变, 从而确保系统有较快的收敛速度。同时本文还提出了通过各连接源端能够获得的局部信息 (发送窗口、协议参数、RTT、传播时延) 判断网络运行状态, 根据运行状态动态调整控制律增益参数的改进算法, 仿真结果表明该两种改进算法克服了 FAST 原有算法静态选择协议参数的缺陷, 能在协议

参数 α 不变的情况下,保证系统处于稳定和小的排队时延情况下,提高了系统收敛到平衡点的速度。

参考文献

- [1] Wei D X, Cheng J, Low S H. FAST TCP: motivation, architecture, algorithms, performance. *IEEE Trans on Networking*, 2006, 14(6): 1246-1259
- [2] Jacobsson K. Dynamic Modeling of Internet Congestion Control [Ph. D dissertation]. Royal Institute of Technology (KT H), Sweden, 2008.25-40
- [3] Jacobsson K, Lachlan L, Low S H. An improved link model for window flow control and its application to FAST TCP. *IEEE Trans on Automatic Control*, 2009, 54(3): 551-564
- [4] Tan L S, Zhang W, Yuan C. On parameter tuning for FAST TCP. *IEEE Communications Letters*, 2007, 11(5): 1-3
- [5] 黄小猛,林闯,任丰源. 高速传输协议研究进展. 计算机学报, 2006, 29(11): 111-120
- [6] 宋丽华,王海涛,陈鸣. 基于网络测量和模糊控制技术的拥塞控制机制. 华南理工大学学报, 2006, 34(6): 89-95
- [7] Migule R, Sergio H. Achieving fair network equilibria with delay-based congestion control algorithms. *IEEE Communications Letters*, 2008, 12(7): 535-537
- [8] 王建新,龚皓,陈建二. 高带宽延时网络中一种协同时拥塞控制协议. 软件学报, 2008, 19(1): 125-135
- [9] 龙承念,杨会龙,关新平. EHSTCP: 改进的高速 TCP 算法. 计算机学报, 2008, 31(3): 143-152

Study on rapid convergence algorithm of FAST TCP

Chen Xiaolong^{* **}, Zhang Yun^{*}, Tian Yiqiang^{**}, Liu Zhi^{*}

(^{*} School of Automation, Guangdong University of Technology, Guangzhou 510090)

(^{**} School of Computer and Electronics and Information, Mao Ming College, Maoming 525000)

Abstract

Aiming at the problem that the FAST TCP static mapping protocol parameter method can not achieve low queuing delay and rapid convergence at the same time, the paper studies the relation between the convergent velocity and FAST TCP parameters by theoretical analysis and NS-2 simulation, and on the basis of this, takes the number of packets each connection maintained in the link buffers as the dimension of the FAST TCP connection's slow start threshold, and proposes an enhanced method that computes the slow start threshold according to each FAST TCP flow protocol parameter, cooperatively estimates the network running state and dynamically tunes the controller gain parameter based on the TCP source local information. The NS-2 simulation results demonstrate that with the application of this enhanced method, the stability and the low queuing delay can be achieved simultaneously, while the system can also converge to equilibrium rapidly.

Key words: queuing delay, rapid convergence, slow start threshold, gain parameter, cooperative