基于动态 MSN 的软交换多业务环境性能研究

马旭涛, 廖建新, 朱晓民
(北京邮电大学网络与交换技术国家重点实验室, 北京 100876)

摘 要: 在下一代网络中将提供具有不同性能要求的语音、数据和多媒体业务, 研究这些业务性能的相互影响并采用有效的方法保证这些业务的性能指标对下一代网络的成功实施具有重要意义. 本文设计了一种多队列的软交换多业务控制框架, 在此基础上提出并分析了一种动态最大会话数(MSN, Maximum Session Number)方法来分配系统资源, 以有效利用系统资源确保业务性能. 仿真结果表明: 在软交换多业务环境中, 采用动态 MSN 方法能确保系统的稳定性并有效地满足各种软交换业务的性能要求.

关键词: 软交换; 多业务; 动态 MSN; 业务性能
中图分类号: TN916.2 文献标识码: A 文章编号: 0372-2112 (2005) 10-1761-05

The Performance Study of Softswitch Multi-Service Environment Based on Dynamic MSN

MA Xu-tao, LIAO Jian-xin, ZHU Xiao-min
(State Key Lab of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract: The next generation network will provide voice, data and multimedia services, which are with different performance requirements. To research the impact of these services' performances between each other and to assure these performances is significant to implement the next generation network successfully. The paper designs a softswitch multi-service control framework with multi-queue, and a dynamic MSN(maximum session number) method to allocate system resources is proposed and analyzed to effectively utilize system resources to insure service's performances, which are based on the control framework. The simulation results prove that the dynamic MSN can ensure the system's stability and satisfy each softswitch service's performance requirement in the softswitch multi-service environment.

Key words: softswitch; multi-service; dynamic MSN; service performance

1 引言

随着 IP 技术的迅猛发展和下一代网络技术研究的不断深入, 智能业务在业务提供、控制和管理上将实现分布化、开放性[1~3]. 软交换作为下一代网络中集成各种网络控制协议的控制层关键实体在这一过程中将扮演重要的角色, 它和业务层应用服务器实体一起提供内容丰富的基于软交换的业务. 软交换可以把底层网络控制协议如事务能力应用部分(TCAP, Transaction Capability Application Part)、会话初始化协议(SIP, Session Initiation Protocol)等封装成具有各种业务能力特征的应用编程接口(API, Application Programming Interface), 如 Parlay API[4] 提供给应用服务器, 应用服务器使用这些 API 构建业务逻辑, 实现对软交换业务会话过程的智能控制.

与提供传统智能网业务的业务控制点(SCP, Service Control Point)相比, 提供软交换业务的应用服务器运行的业务种类更多更复杂, 这些业务具有不同的时延和吞吐量要求[5~14], 这样的应用服务器可以称为软交换多业务环境. 本文主要研究在软交换多业务环境中, 由不同软交换发起的不同业务会话请求在不同业务和系统负载情况下, 各业务的性能及相互的影响. 并在充分利用系统资源和保证系统稳定性的前提下, 提出使用动态最大会话数(MSN, Maximum Session Number)方法来确保各业务的会话请求等待时间、吞吐量等性能指标, 并通过分析和仿真证明了该方法的有效性.

2 软交换多业务控制

我们构建了如图 1 所示的软交换多业务控制框架, 它主要由软交换多业务前台和软交换多业务后台组成. 软交换前台接受来自不同软交换设备的不同业务会话请求消息, 并根

收稿日期: 2004-12-10; 修回日期: 2005-05-11
基金项目: 高等学校博士学科点专项科研基金(No. 20030013006); 国家移动通信产品研究开发专项项目: 下一代移动智能网络系统的开发及应用; 电子信息产业发展基金项目: 移动通信增值服务平台及应用系统

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

据软交换设备号和业务类型号, 把这些请求分发到不同的队列, 如果软交换设备的个数是 p, 业务类型个数是 q, 则业务会话请求消息队列的个数是 $p\times q$. 每个会话请求消息队列对应一个请求消息分发线程, 该线程根据当前队列负载情况和软交换多业务后台业务处理线程负载情况, 把队列头的会话请求消息分发给后台一空闲业务处理线程进行处理. 软交换多业务后台是由多台服务器组成的集群系统, 他们的软硬件配置是相同的, 他们通过前台系统测量线程传递业务处理线程的忙闲状态, 各队列的请求消息分发线程从系统测量线程处获得后台空闲业务处理线程所在的后台号、线程号和其当前允许的最大同时会话线程数, 以确定会话请求消息的分发目标.

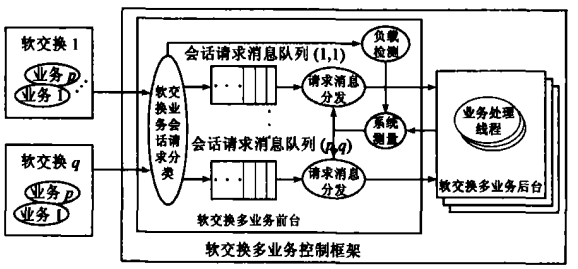


图 1 软交换多业务控制框架

对每个会话请求消息队列中的消息负载, 我们采用负载检测线程执行如下检测算法:

$$\begin{cases} v_i = m/(arr_time_{im} - arr_time_{(i-1)m}) \\ \lambda_i = (1-\alpha)\lambda_{i-1} + \alpha v_i \end{cases} \quad (1)$$

其中 m 为负载检测周期, 系统每接受到 m 个消息就作一次负载检测, i 为负载检测次数. arr_time_{im} 表示第 im 个消息到达系统的时间, v_i 为第 i 个负载检测周期检测到的 m 个消息的平均到达率. λ_i 为第 i 次负载检测后, 系统认定的业务到达率, α 介于 0 和 1 之间, 是系统前一次认定的业务到达率和系统本次检测到的到达率之间的加权平滑常数. 此算法具有对业务负载进行预测的功能[15~16], 当 α 大则表示新近业务到达率在系统后续认定业务负载中占有较大比例, 如果 α 小表示系统原先认定的业务负载在系统后续认定的业务负载中占有较大比例. 我们可以通过对负载变化进行长时间的观察, 以选取合适的 α, 这样既可以较好地反映系统最新的负载情况, 又可有效地防止系统负载的颠簸.

系统以可同时运行的最大业务线程数作为系统容量, 该值可用系统压力测试的方法得到, 在测试中由系统管理员根据各会话请求消息队列的统计平均到达率、统计平均服务率和系统容量, 确定各队列到达率过载的门限值 λ_d, 系统根据各会话请求消息队列的负载情况动态分配允许和该队列的会话请求消息相关的同时最大业务处理线程数即系统容量. 我们采用如下动态策略分配各会话请求消息队列对应的系统容量:

设未过载业务队列组成的集合为 T, 过载业务队列组成的集合为 G; 系统总容量即系统同时最大业务处理线程数为 C, 队列 (i,j) 在会话请求消息队列均未过载时分配的实际系统容量为 $C_{(i,j)}$, 而在有会话请求消息队列过载时分配的实际系统容量为 $C'_{(i,j)}$; β 为系统容量动态分配因子, 其取值范围一般设为 $[0.05, 0.2]$. 容量分配需满足以下条件:

(1) 会话请求消息队列均未过载时

$$\sum_{(i,j)\in T} C_{(i,j)} = C \quad (2)$$

(2) 有会话请求消息队列过载时

$$\begin{cases} C'_{(i,j)} = C_{(i,j)} + \dfrac{C_{(i,j)}}{\sum\limits_{(p,q)\in G} C_{(p,q)}}\beta C & (i,j)\in G \\ \sum\limits_{(i,j)\in T}(C_{(i,j)} - C'_{(i,j)}) = \beta C \end{cases} \quad (3)$$

在式(2)中系统分配给各未过载会话请求消息队列的业务处理线程数要求满足各会话请求消息队列的性能指标, 在式(3)中要求系统分配的业务处理线程数首先满足每个未过载消息队列的性能指标要求. β 表示系统从未过载消息队列中借用过来并分配给过载消息队列的容量所占系统总容量的最大比例, 该值越大, 则系统抗过载的能力越强, 但轻负载会话请求队列的业务处理性能会受到较大影响; 该值越小, 则系统容量动态调整的范围就小, 会话请求队列过载时业务性能将会受到较大的影响, 但不影响此时轻负载会话请求队列的业务处理性能. 我们根据队列负载变化的范围设置合适的 β, 要求既能在一定程度上满足轻负载会话请求队列的业务处理性能, 又能有效利用系统容量, 确保过载业务的性能指标和系统稳定性.

3 动态 MSN 分析

在软交换多业务环境下, 同一个会话请求消息队列中的消息是来自同一个软交换设备的同一种业务会话请求. 我们设计的动态 MSN 用来保证这些队列中会话请求的性能指标, 这种动态 MSN 方式主要有两部分构成, 第一部分是根据当前队列中消息会话请求的性能动态调整该队列消息缓存区的大小, 保证队列消息过载时的业务会话请求等待时间和吞吐量指标; 第二部分是系统根据各队列负载变化情况动态调整允许各队列会话请求消息相关的同时最大业务会话数, 以有效利用系统资源优化业务性能.

设一业务会话请求消息队列中, 消息到达率为 λ, 每个业务处理线程对该队列业务的平均服务率是 μ, 系统允许该队列业务建立的最大同时会话数是 n, 当 $\lambda < n\mu$ 时, 系统是稳定的, 但如果不限制最大消息缓存区的大小则业务的性能指标将得不到保证, 在该队列业务处于轻负载状态时, 系统将动态地把分配给该队列的系统资源调配给其他队列业务, 以在不影响本队列业务性能指标的前提下, 优化系统资源提高其他重负载队列业务的性能. 当 $\lambda \geqslant n\mu$ 时, 必须限制消息缓存区的大小保证系统的稳定性, 并根据系统当前状态, 尽可能多调配一些业务处理线程为该队列消息服务. 而当系统中所有业务队列都处于过载状态, 在优化分配业务处理线程的基础上, 采用动态调整各队列消息缓存区大小来确保业务性能指标. 这时如果增大缓存区, 虽然会增大业务吞吐量, 但其会话请求等待时间也会迅速增加, 如果减小缓存区, 则会话请求等待时间减小的同时吞吐量也会减小.

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

设 W 为业务会话请求等待时间指标, TP 为业务吞吐量指标, 在系统动态分配业务处理线程资源和各业务队列消息缓存区大小时, 需满足以下约束条件:

(1) $\lambda \leqslant TP$ 时

$$\begin{cases} n^n \rho^{n+m} p_0 / n! \longrightarrow 0 \\ w \geqslant L_q / (\lambda (1 - n^n \rho^{n+m} p_0 / n!)) \end{cases} \tag{4}$$

(2) $\lambda > TP$ 时

$$\begin{cases} TP \leqslant \lambda (1 - n^n \rho^{n+m} p_0) / n! \\ W \geqslant L_q / TP \end{cases} \tag{5}$$

其中 m 表示系统分配给该业务队列消息缓存区的大小, L_q 表示业务队列消息缓存区中消息的平均个数, $\rho = \lambda / n\mu$ 是业务队列消息负载强度, p_0 表示系统分配给该业务队列的所有 n 个业务处理线程都空闲的概率.

我们设计了如下仿真算法来计算采用动态 MSN 方法的会话请求消息队列中的消息到达队列后等待服务的时间. 用一个随机产生的泊松流 status(1, :) 数组表示会话请求消息到达时间间隔, 用一个符合指数分布的 status(2, :) 数组表示和会话请求消息对应的一次会话过程中业务处理时间, status(3, :) 数组表示每个会话请求消息在队列中的等待服务时间, status(4, 1: $i-1$) 数组表示第 i 个消息到达后前面每个 $i-1$ 个消息将在系统中逗留的时长. n 表示系统允许该业务队列同时建立的业务会话数, m 表示该业务队列消息缓存区大小. 执行如下算法步骤:

(1) 首先计算在第 i 个消息到达后, 前面所有 $i-1$ 个消息仍将在系统中逗留的消息个数和逗留时间: 计算 status(4, :) 数组中数值大于 0 的个数 front_num, 即第 $i-1$ 个消息到达后系统中逗留的消息个数. 在 front_num 中, 把前 n 个逗留消息表示为 k, 则值 status(4, k) = status(4, k) − status(1, i). 后 front_num − n 个消息在系统中的逗留时间可以通过一个并行处理算法求得: 用数组 $p(1:n)$ 表示该业务队列对应的所有后台业务处理线程从当前状态到空闲状态所需要的最小时间, 初值等于 front_mes(1, 1: n); 先计算逗留消息中前 n 个消息的逗留时间 status(4, front_mes(2, 1: n)) = status(4, front_mes(2, 1: n)) − status(1, i), 同时 $p(1:n) = p(1:n) -$ status(1, i); 然后计算等待队列中第 j 个消息的逗留时间, $n < j \leqslant$ front_num, 其方法是在 $p(1:n)$ 中选择最小值 p(min), 如 p(min) < 0, 则 p(min) = p(min) + status(2, j), 同时 status(4, j) = p(min). 这时 status(4, :) 中大于 0 个消息个数为第 i 个消息到达系统中仍将在系统中逗留的消息个数, 表示为 stay_num, 则等待队列中的消息个数为 wati_num = stay_num − n; status(4:) 中大于 0 的值为相应的持续逗留时间, 逗留消息的消息序号保留在 temp(:) 数组中.

(2) 然后计算第 i 个消息在系统中等待服务的时间, 执行如下算法:

```
status(3, i) = 0;
% 会话请求消息队列中没有等待消息
if wait_num == 0  status(3, i) = min(p(:)); end;
if (wati_num > 0 & wait_num < m) % 会话请求消息队列中有等待消息
    for k = 1:wait_num% 获得当前p(:)数组的最小值及其下标
        [min_value, min_suffix] = min(p(:));
        p(:) = p(:) - min_value;
        p(min_suffix) = status(2, temp(k));
        status(3, i) = status(3, i) + min_value;
    end;
end;
% 会话请求消息队列的缓存区已满, 第 i 个消息将被系统丢弃
if wait_num >= m status(3, i) = 0; status(2, i) = 0; end;
```

(3) 最后就可求得在动态 MSN 软交换多业务环境中, 系统接受的会话请求消息消息特征 en_status(3, :). en_status(1, :) 表示请求消息到达时间间隔, en_status(2, :) 表示每个请求消息对应的这次会话过程服务时间, en_status(3) 表示请求消息在队列中等待的时间. 设 reject_num 是初值为 0 连续消息丢弃数, reject_status(:) 为拒绝消息对应的原到达时间间隔. 在检索 status(2, :) 数组时, 当 status(2, i) > 0 时, en_status(2, m) = status(2, i), en_status(3, m) = status(3, i), m 是被接受消息在 en_status(2, :) 数组中的序号, $m = m + 1$, 如果 reject_num > 0 则, en_status(1, m) = status(1, i) + sum(reject_status(1: reject_num), reject_num = 0; 如果 reject_num ⩽ 0 则 en_status(1, m) = status(1, i); 如果 status(2, i) = 0, 则 reject_num = reject_num + 1, reject_status(1, reject_num) = status(1, i).

通过上述算法求得每个被系统接受的业务会话请求消息的等待时间后, 利用在不同时刻消息到达数和离开数的关系就可求得每个时刻系统处理的该业务队列相关的业务会话数, 进而可求得各个时刻对应的系统平均吞吐量.

4　理论计算与仿真结果

我们设置两个软交换, 其中一个软交换提供两种业务, 业务会话请求消息分别对应业务队列 S1 和 S2; 另一软交换提供一种业务, 业务会话请求消息对应业务队列 S3. 按照表 1 的参数配置, 结合 M/M/n/m 模型[17-19]和动态 MSN 仿真算法使用 MATLAB[20]对不同负载情况下业务会话请求等待时间、吞吐量进行了理论计算和仿真计算.

在 MSN 仿真时, 我们使用式(1)检测业务队列负载, 从表 1 中我们容易看出不同业务队列负载情况下系统分配给各队列的业务处理线程资源满足式(2)和式(3)的要求, 并且在不同业务队列在不同轻重负载情况下其对应的业务会话请求等待时间、业务吞吐量满足式(4)和式(5)的约束条件.

图 2(a)和(b)分别给出了 S1 未过载时业务会话请求平均等待时间和平均吞吐量受 S2、S3 负载情况的影响. 可以看出当 S2、S3 均未过载时, 系统不需要调配出 S1 的业务处理线程资源, 这时 S1 的业务会话请求等待时间较小和 S1 的业务会话吞吐量较大; 当 S2、S3 有过载时, 系统分配给 S1 的业务处理线程资源减小, 这时 S1 的业务会话请求等待时间变大, S1 的业务会话吞吐量变小. S1 的业务会话吞吐量在不同的 S2

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

和 S3 负载变化组合下,都能基本接近会话请求消息的输入速率,S1 的业务会话请求平均等待时间和业务会话吞吐量都能趋于稳定且满足性能指标要求.

表 1 系统会话处理线程动态分配时的软交换多业务性能

S1、S2、S3 为三个会话请求消息队列中会话请求消息对应的业务 μ_1= 3 个/秒, $\lambda d1 = TP_1$= 150 个/秒, W_1= 0.7 秒; μ_2= 4 个/秒, $\lambda d2 = TP_2$= 120 个/秒, W_2= 0.8 秒; μ_3= 5 个/秒, $\lambda d_3 = TP_3$= 100 个/秒, W_3= 1.0 秒. 系统总容量为 100 个同时业务处理线程, 系统容量动态分配因子 β= 0.1, 每个业务会话请求消息缓存容量为 100 个消息. n_i 表示系统分配给业务队列 i 的会话处理线程数

多业务负载情况			n_i	理论计算		MSN 仿真	
				等待时间	吞吐量	等待时间	吞吐量
均未过载	λ1= 120		50	0.0029	120.0000	0.0029	119.9893
	λ2= 90		30	0.0031	90.0000	0.0032	89.8461
	λ3= 60		20	0.0006	60.0000	0.0006	59.7085
有一个业务过载	S1 过载	λ1= 180	60	0.2613	178.3393	0.2684	176.7529
		λ2= 90	24	0.1113	89.9940	0.1132	89.5446
		λ3= 60	16	0.0102	60.0000	0.0108	59.2962
	S2 过载	λ1= 120	46	0.0148	120.0000	0.0149	119.3721
		λ2= 140	36	0.1652	139.8019	0.1703	139.8256
		λ3= 60	18	0.0025	60.0000	0.0027	59.3076
	S3 过载	λ1= 120	48	0.0065	120.0000	0.0061	119.7024
		λ2= 90	28	0.0088	90.0000	0.0093	89.5227
		λ3= 120	24	0.4053	118.8553	0.4044	118.9658
有两个业务过载	S1、S2 过载	λ1= 180	54	0.5617	161.9998	0.5724	161.0507
		λ2= 140	34	0.5227	135.8163	0.5512	134.2131
		λ3= 60	12	0.8249	59.4177	0.8387	58.4487
	S2、S3 过载	λ1= 120	45	0.0227	120.0000	0.0229	119.1472
		λ2= 140	33	0.6341	131.9857	0.6257	132.8976
		λ3= 120	22	0.8092	109.9990	0.7876	111.2078
	S1、S3 过载	λ1= 180	55	0.5394	164.9989	0.5418	164.2472
		λ2= 90	23	0.3160	89.7885	0.3258	88.4075
		λ3= 120	22	0.8092	109.9990	0.8231	108.8751
均过载	λ1= 180		50	0.6333	150.0000	0.6368	149.5577
	λ2= 140		30	0.7833	120.0000	0.7899	118.9868
	λ3= 120		20	0.9500	100.0000	0.9578	99.6791

图 2(c)和(d)分别给出了 S1 过载时业务会话请求平均等待时间和平均吞吐量受 S2、S3 负载情况的影响. 这时 S1 的业务性能受其他业务负载变化的影响和 S1 轻负载时的情况基本类似. 区别在于当 S1 过载时,受其他过载业务的影响,它的业务会话请求等待时间变化幅度会很多,当 S1 的负载进一步增加时,必须通过减小 S1 相关的会话请求消息缓存大小来保证会话请求等待时间指标,这时的代价是业务吞吐量将略有下降,如表 2 所示.

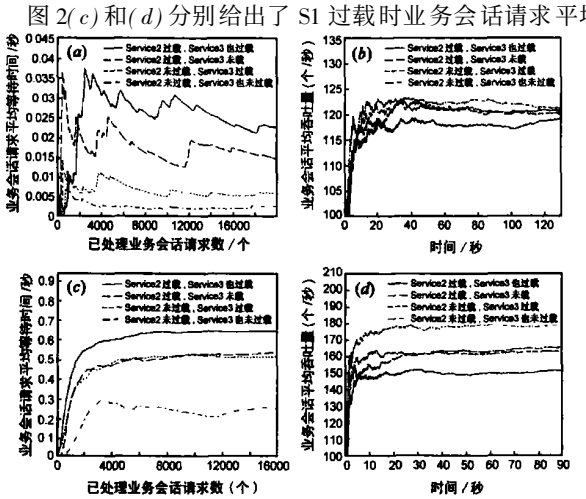


图 2 S1 业务性能受 S2 和 S3 负载的影响

表 2 系统动态分配会话请求消息缓存大小时 S1 的业务性能

μ_1= 3 个/秒, $\lambda d1 = TP_1$= 150 个/秒, W_1= 0.7 秒. n_i 表示系统分配给业务队列 i 的会话处理线程数

业务负载情况	n_i	业务会话请求消息缓存大小	理论计算		MSN 仿真	
			等待时	吞吐量	等待时间	吞吐量
S1 未过载, λ1= 120	50	10	0.0020	119.7742	0.0021	119.0816
		20	0.0027	119.9759	0.0027	119.1689
		50	0.0029	120	0.0028	119.3358
S1 过载, λ1= 180	60	10	0.0176	170.2109	0.0181	169.9813
		50	0.1234	176.9172	0.1255	175.2245
		150	0.3998	178.8635	0.3891	177.9752

从表 2 中可以看出,在 S1 在未过载时,业务性能受业务会话请求消息缓存大小的变化并不显著,随着缓存的增大,等待时间和吞吐量趋于恒定. 但在 S1 过载时,业务性能对业务会话请求消息缓存大小的变化较为敏感,随着缓存的增大,等待时间也越来越大,而吞吐量只是缓慢增加并趋向于 λ_1,这时动态调整缓存,以牺牲小的业务吞吐量换来业务会话请求等待时间的较大改善对确保业务性能就显得由为重要. S2 和 S3 在不同业务负载情况下的业务性能受其他业务负载变化的影响和 S1 相类似,当有更多软交换和更多种业务类型时,系统中将有更多的业务队列,这些业务队列相关的业务性能在不同负载情况下受其他业务负载变化的影响也和 S1 相类似,在此不再赘述.

5 结束语

在下一代网络中将提供各种语音、数据、多媒体业务[2],这些业务具有不同的性能指标要求,如何在一个统一的软交换多业务环境中充分利用有限的系统资源来保证各种业务的性能要求,是以业务为驱动的下一代网络能否取得成功的重要问题. 本文的贡献在于设计了一种软交换多业务控制框架,在此框架中来自不同软交换的不同业务对应不同的业务队列,针对这些业务队列提出并分析了一种动态 MSN 的系统资源分配方式,即根据业务队列消息负载情况和系统其他业务负载情况,结合业务性能指标的要求,动态分配系统业务会话处理线程和会话请求消息缓存区大小,有效利用系统资源,确保各业务队列对应的业务会话请求等待时间指标和业务吞吐量指标. 最后通过仿真证明在软交换多业务环境中,采用动态 MSN 方法能有效地满足各种软交换业务的性能要求.

参考文献:

[1] 廖建新. 移动智能网技术的研发现状及未来发展[J]. 电子学报, 2003, 31(11): 1725-1731.

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

LIAO Jian-xin. The study status and perspective of mobile intelligent network technology[J]. Acta Electronica Sinica, 2003, 31(11): 1725-1731. (in Chinese)

[2] Brennan R et al. Evolutionary trends in intelligent networks[J]. IEEE Communications Magazine, 2000, 38(6): 116-123.

[3] Finkelsterin M, et al. The future of the intelligent network[J]. IEEE Communications Magazine, 2000, 38(6): 100-106.

[4] Parlay working Group, Parlay Specification 4.0[EB/OL]. http://www.etsi.org. July 2002.

[5] Donald E Smith. Ensuring robust call throughput and fairness for SCP overload controls[J]. IEEE/ACM Trans on Networks, 1995, 3(5): 538-548.

[6] Kihl M, Nyberg C. Investigation of overload control algorithms for SCPs in the intelligent network[J]. IEE Proceedings-Communications, 1997, 144(6): 419-423.

[7] Ma Y X, et al. The research on overload control algorithms for SCP of intelligent network[A]. WCICA'00[C]. Hefei, China: IEEE Standards Office, 2000, 4: 2622-2625.

[8] Li T H, Liao J X, Chen J L. The study of SCP overload control in multi-service environment[J]. Journal of Electronics(China), 1999, 16(3): 230-237.

[9] S Dhakal, B S Paskaleva. Dynamical discrete-time load balancing in distributed systems in the presence of time delays[A]. Proceedings of the 42nd IEEE Conference on Decision and Control[C]. Maui, HI, USA: Institute of Electrical and Electronics Engineers Inc, 2003, 12: 5128-5134.

[10] Wei Wu. The study on overload control of application service in next-generation networks[A]. ICCT2003[C]. Beijing China: Institute of Electrical and Electronics Engineers Inc, 2003, 4: 1429-1432.

[11] Dorina Petriu, Christiane Shousha. Architecture-based performance analysis applied to a telecommunication system[J]. IEEE Trans on Software Engineering, 2000, 26(11): 1049-1065.

[12] Ludmila Cherkasova, Peter Phaal. Session-based admission control: A mechanism for peak load management of commercial web sites[J]. IEEE Trans on Computers, 2002, 51(6): 669-685.

[13] H Chen, P Mohapatra. Session-based overload control in QoS-aware Web Servers[A]. IEEE INFOCOM'2002[C]. New York, USA: IEEE 2002, 2: 516-524.

[14] M A Kashem, V Ganapathy. Network reconfiguration for load balancing in distribution networks[J]. IEEE Proc-Gener Trans Distrib, 1999, 146(6): 563-567.

[15] Kihl M, Nyberg C. A study of methods for protecting SCP from overload[A]. IEE Telecommunications' Conference Publication no. 404[C]. Brighton, UK: IEEE, 1995. 125-129.

[16] XU Feng, WANG Zhi-fang, WANG Bao-sheng. Research on AR model applied to forecast trend of vibration signals[J]. Journal of Tsinghua University(Science and Technology), 1999, 38(4): 57-60.

[17] 孙荣恒, 李建平. 排队论基础[M]. 北京: 科学出版社. 2002.

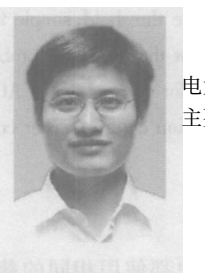
[18] 林闯, 杨士强. 多服务器多队列系统调度方案的性能分析[J]. 电子学报. 2002, 28(5): 17-20.
LIN Chuang, YANG Shi-qiang. Performance analysis of scheduling schemes in multiserver multiqueue systems[J]. Acta Electronica Sinica, 2002, 28(5): 17-20. (in Chinese)

[19] G Cybnko. Dynamic load balancing for distributed memory multiprocessors[J]. IEEE Parallel and Distributed Computing Trans, 1989, 7(10): 279-301.

[20] 范影乐. MATLAB 仿真应用详解[M]. 北京: 人民邮电出版社. 2002.

[21] Stan Moyer, Amjad Umar. The impact of network convergence on telecommunications software[J]. IEEE Commun Mag, 2001, 39(1): 78-84.

作者简介:



马旭涛　男, 1977 年生, 江苏溧阳人, 北京邮电大学网络与交换国家重点实验室博士研究生, 主要研究方向为下一代智能网.

缪建新　(见本期第 1852 页)
朱晓民　(见本期第 1750 页)

© 1994-2010 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net