

一种网络延迟界限精确控制的调度算法

王 勇,江开忠,顾君忠,吕 钊

(华东师范大学 计算机科学系,上海 200062)

(ywang@ica.stc.sh.cn)

摘 要:在网络数据传输调度中,基于最早时限优先(EDF)的算法具有单点最优的延迟界限控制能力。现有的各种 EDF 改进算法,主要着眼于提供延迟上界的保证能力,而当采用机顶盒之类的缓冲能力较弱的设备作为客户终端时,还需要网络提供精确的延迟下界控制能力。在原有 EDF 改进算法的基础上提出了精确延迟界限控制的最早时限优先算法。该算法不但能同时保证延迟上界和下界,还使得节点可以独立地决定为数据流分配的缓冲区大小,并增加了节点允许抖动量的取值范围,从而提高了节点数据的转发效率。

关键词:最早时限优先;精确延迟界限;服务质量;调度

中图分类号: TP393.09 **文献标识码:** A

Scheduling algorithm for precision delay bounds

WANG Yong, JIANG Kai-zhong, GU Jun-zhong, Lü Zhao

(Department of Computer Science, East China Normal University, Shanghai 200062, China)

Abstract: The scheduling algorithms based on Earliest Deadline First (EDF) have best control-capability of delay bounds in packet switch network. Most of variants of EDF concentrate the guarantee of upper delay bounds. For the terminals with poor buffer resources, lower delay bounds also need to be provided by network. A new algorithm, PD-EDF, based on some improved EDF algorithms was proposed to provide precision delay bounds. This new algorithm empowers any node to decide the buffer size for data streams independently and improves the data transmitting efficiency of node by increasing the scope of value ε .

Key words: Earliest Deadline First(EDF); precision delay bound; QoS; scheduling

网络 QoS 主要体现在数据流使用的网络带宽、到达延迟和延迟抖动等几个方面的服务参数^[1]。IPTV 的大范围推广需要部署众多低成本的机顶盒设备——一种以嵌入式技术为主的瘦客户终端。通常这类终端资源有限,不能够提供足够的缓冲能力以平滑网络的延迟和抖动,而必须由网络来进行 QoS 保证。目前的网络 QoS 保障技术主要包括 IntServ, DiffServ, MPLS 和流量工程 TE 等,它们提供的是数据流级别的粗粒度 QoS 控制,不能提供精确的端到端延迟和延迟抖动控制能力。各种网络 QoS 技术主要是通过网络节点对数据包转发的控制和调度来实现,通常采用对不同优先级的输出队列实施不同的调度策略来分配带宽和调整延迟时间,从而实现 QoS 服务保障。近年来,研究人员开发出多达几十种的分组调度算法,主要包括基于静态优先级、基于轮循、基于 GPS 模型、基于时间延迟、分层链路共享、核心无状态算法、基于服务曲线、比例区分算法、结合缓冲管理的算法等^[1]。各个类别的算法还存在很多改进算法。在这些算法中,只有 WFQ (Wighted Fair Queuing) 和 EDF (Earliest Deadline First) 能提供单个数据包级细粒度的 QoS 控制能力^[2]。它们都采用了包级动态优先权,提供单个数据包的端到端延迟控制。而 EDF 算法被证明具有单个节点最优的延迟界限控制能力^[3],并且如果采用每个节点的流量整形,具有比 WFQ 更好的端到端的延迟控制能力^[4]。但这些算法的主要目标都是最有效地提供端到端延迟上界的保证,对于延迟下界的保证

没有更多的探讨。当客户终端是缓冲能力有限的设备时,不但要求网络提供延迟上界的保证,还需要提供端到端延迟下界的控制。本文分析了各种 EDF 改进算法的优缺点,提出了能够满足低缓冲能力的瘦客户端需求的精确延迟控制的 EDF 改进算法(PD-EDF)和网络延迟控制参数的计算方法。

1 基本 EDF 算法

当用作队列调度算法的时候,基于时延控制的 EDF 算法思想是:单个节点(考虑数据只经过一个节点的情况)给某个网络数据流 i (后面简称流 i) 分配一个时延参数 d_i 作为该流所有数据包在该节点的时延上界;当流 i 中的第 j 个数据包在 $t_{i,j}$ 时刻到达时,为该数据包计算一个时间标签 $a_{i,j} = t_{i,j} + d_i$ 作为其在节点的到期时间,然后放入输出队列。输出队列调度算法每次都根据队列中各个数据包的到期时间进行排序,选择具有最小到期时间的数据包发送出去。因此,该类调度算法具有 $O(\log N)$ 的排序复杂度。

其数学表示如下:令 $A_i(s, t)$ 为流 i 在 $[s, t]$ 这段时间内到达的数据包($s \geq 0$),且 $A_i(t) = A_i(0, t)$ 。同时定义一个增函数 $A_i^*(t)$ 作为流 i 的流量限制函数, $A_i^*(t)$ 函数的特征为:当 $t < 0$ 时 $A_i^*(t) = 0$;对于任意的 t_1, t_2 , 当 $t_1 < t_2$ 有 $A_i^*(t_1) \leq A_i^*(t_2)$ 。如果在任意的时间间隔 $[s, t]$ 都有 $A_i(s, t) \leq A_i^*(t - s)$, 则称 $A_i(t)$ 是符合流量限制函数 $A_i^*(t)$ 的,记作 $A_i(t) \leq A_i^*(t)$ 。

收稿日期:2006-02-20;修订日期:2006-04-18

作者简介:王勇(1974-),男,重庆人,工程师,博士研究生,主要研究方向:流媒体编码与传输、数字电视;江开忠(1965-),男,四川乐至人,讲师,博士研究生,主要研究方向:多媒体技术、知识发现、协同设计;顾君忠(1949-),男,上海人,教授,博士生导师,主要研究方向:分布式数据库、CSCW;吕钊(1970-),女,四川江油人,副教授,博士,主要研究方向:多媒体技术、CSCW、信息安全。

当任意的流 i 的流量函数 $A_i(t) \leq A_i^*(t)$, 只要该节点的带宽都能够满足公式(1), 则能够保证该流的每个包在节点的延迟上界为 d_i , 并且该条件是一个充分必要条件。

当 $t \geq 0, i \in$ 所有流的集合:

$$\sum A_i^*(t - d_i) + l_{\max} * I\{d_{\min} \leq t \leq d_{\max}\} \leq ct \quad (1)$$

其中, c 是该节点的出口带宽, l_{\max} 是所有流最大的数据包尺寸, $d_{\min} = \min_i(d_i)$, $d_{\max} = \max_i(d_i)$, $I\{E\}$ 是条件 E 的指示函数。

2 各种改进的 EDF 算法分析

基本的 EDF 算法并不是专门为网络队列调度而设计的。应用于网络环境时, 为了在业务流发生突发的时候保证端到端的延迟时间, 每个节点还需要引入速率控制机制对收到的流量进行调整。由于基本 EDF 算法的充分必要条件限制的计算太复杂, 通常采用一个简化的充分条件来判断能够保证流的延迟上界。

当 $t \geq d_{\min}, i \in$ 所有流的集合, 满足:

$$\sum A_i^*(t - d_i) + l_{\max} \leq ct \quad (2)$$

加入速率控制后的 EDF 算法在一个节点内分为速率控制和输出队列调度两部分

2.1 EDF 改进算法一览

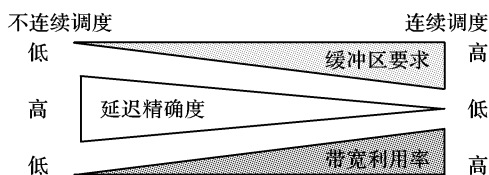


图1 性能特征比较

速率控制可以采用抖动控制或者流量整形技术, 同时节点还可以对到来的数据包采取连续调度或者不连续调度的方法。采用的不同技术造就了多种不同的 EDF 改进算法, 包括 DC-EDF^[2], RC-EDF^[4], Jitter-EDD^[5], Delay-EDD^[6], EEDF^[7], JT-EDF^[8] 等, 它们采用的不同技术组合如表1所示。连续调

度和不连续调度具有的不同性能特征见图1。

表1 技术的差异

| 速率控制技术 | 不连续调度 | 连续与不连续相结合 | 连续调度 |
|--------|------------|-----------|-----------|
| 流量整形 | RC-EDF | EEDF | Delay-EDD |
| 抖动控制 | Jitter-EDD | JT-EDF | DC-EDF |

2.2 基于 EDF 的算法分类与分析

2.2.1 基于流量整形的方法

流量整形方法是所有节点都按照该流的流量限制函数对输入流量进行调整, 对超出允许流量的数据包产生一个额外的延迟时间, 使其最大延迟时间大于原定延迟上界。

基于流量整形的 EDF 改进算法^[4,6,7]的基本过程为: 定义流量整形延迟函数 $D(I_i \| A_i^*(t))$, 其中 I_i 为进入节点的数据流 i 的流量函数, $A_i^*(t)$ 为流 i 的流量限制函数; 在任意节点 k 都采用流量整形技术对到达的任意流 i 进行速率控制, 即对流 i 中超过流量限制的部分数据包产生的额外延迟时间, 使得该流的流量符合整形函数。流量整形可以采用连续和不连续两种调度方式。当采用不连续调度时, 流量整形功能对进入节点过快的数据包先延迟一定时间, 再发送到输出队列。这种方式下会造成带宽的浪费。而采用连续调度的 EDF 改进算法将该延迟时间量加到数据包的时间标签中, 从而降低该数据包在输出队列中被调度的低优先级, 尽量使之到达下一节点时不会过快。

如图2所示: (a) 显示了每个节点根据函数 $A_i^*(t)$ 进行流量整形的特征曲线; (b) 显示了当节点 k 负载较轻的时候, 提前输出该流的数据包, 即对数据包的实际延迟时间为 $d_i^{k'}$ (小于 d_i^k)。产生的影响可以从图3(c)中看到: $k+1$ 节点在预计的最大延迟时间之前就收到数据流, 但它进行整形的时候并不处理该提前时间量, 而只是按照算法对输入流量中后续的数据包按照函数 $A_i^*(t)$ 进行流量整形。流量限制曲线为 $OWXZ$, 输入流量曲线为 OXY 。因此, 采用流量整形的方法不提供延迟下界的控制能力。

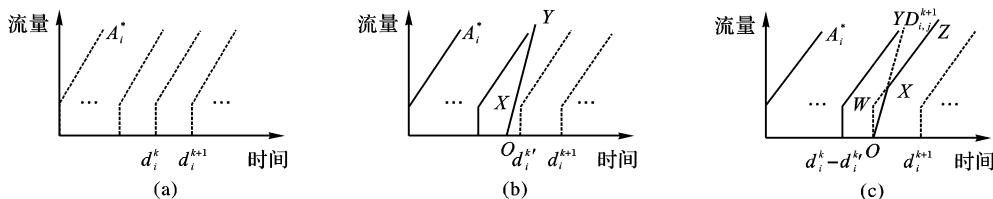


图2 使用流量整形算法的示例

2.2.2 基于抖动控制的方法

基于抖动控制的算法^[2,5,8]的主要特征是: 除了第一个节点对输入流量按照 $A_i^*(t)$ 整形外, 其余节点采用延迟抖动控制方法来进行速率控制。抖动控制算法考虑每个数据包到达本节点的时候, 已经经过的总延迟时间与预计的最大延迟相差多少, 从而采取相应的控制技术。如图3(c), 节点 k 对任何数据包的提前发送都不会引起 $k+1$ 节点的速率控制部件对该数据流的输出时间。同样, 基于抖动控制的 EDF 算法也

包含连续调度和不连续调度两种方法。不连续调度算法对下一个节点的缓冲能力要求很高, 并且不能统计利用不同节点的空闲带宽。连续调度算法具有统计利用各节点带宽的能力, 但它通过输出队列的调度算法控制时间延迟具有不稳定性, 因此不能保证端到端延迟的下界。文献[8]提出的 JT-EDF 算法综合了这两种方法, 通过一个参数来控制节点究竟采用那种算法, 并详细证明了该算法的有效性和时延界的保证能力。

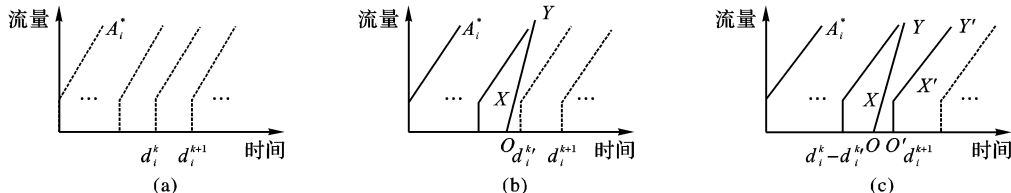


图3 使用抖动控制的整形算法的示例

3 新的端到端精确时延控制算法 PD-EDF

在 EDF 的改进算法中,各种 EDF 算法都能够很好地保证数据流的端到端时延上界。而基于不连续调度的抖动控制的 EDF 算法则还可以提供端到端时延的下界控制,但是现有的这类 EDF 改进算法对于每个节点的能力要求较高,也没有很好地统计利用整个链路中各个节点的能力。Jitter-EDD 和 JT-EDF 都给出了对于时延下界至关重要的控制变量,但是对于这个变量该如何设置以及如何动态修改来最大化节点的吞吐量,没有很好的说明。本文在此基础上提出了能较好解决这些问题的算法。

3.1 Jitter-EDD 和 JT-EDF 保证时延上下界的算法分析

流 i 在第一个节点被整形后符合流量限制函数 $A_i^*(t)$,设整形产生的最大时间延迟为 $d_i^{shmax} = \max_j(e_{i,j}^1)$,最小的时间延迟为 $d_i^{shmin} = \min_j(e_{i,j}^1)$;后续的每个节点 k 对于流 i 的延迟上界为 d_i^k ,允许的抖动量(即可对数据包输出的最大时间提前量)为 ε_i^k ;流 i 从发送端到接收端共经过 h_i 个节点;则当每个网络节点 k 的带宽都满足公式(2)时:

流 i 的延迟下界为:

$$D_i^{upper} = d_i^{shmax} + \sum_{k=1}^{h_i} d_i^k \quad (3)$$

流 i 的延迟下界为:

$$D_i^{lower} = d_i^{shmin} + \sum_{k=1}^{h_i-1} d_i^k - \sum_{k=1}^{h_i} \varepsilon_i^k \quad (4)$$

数据包 j 到达节点 k 时的提前时间为: $e_{i,j}^k = D_{i,j}^{k-1} - t_{i,j}^k$ 。

输出包 j 在抖动控制器的延迟时间为: $e_{i,j}^k - \varepsilon_{i,j}^k$ ($0 \leq \varepsilon_{i,j}^k \leq e_{i,j}^k$)。

实际上,当 $\varepsilon_{i,j}^k = 0$,就是不连续调度抖动控制算法;当 $\varepsilon_{i,j}^k = e_{i,j}^k$,就是连续调度抖动控制算法。

在 Jitter-EDD 中还指出了节点 k 需为流 i 的速率控制器分配用于抖动控制的缓冲空间为:

$$b_i^k = j_i^{k-1} * \max(A_i^*(t)) \quad j_i^{k-1} = d_i^{k-1} + \sum_{k=1}^{h_i} \varepsilon_i^k \quad (5)$$

其中, j_i^{k-1} 代表节点 k 的收到数据包的最大提前时间, $\max(A_i^*(t))$ 为流 i 的最大速率。从公式(5)可以看出,缓冲空间分为两个部分:一部分用于平滑上一个节点调度的延迟不稳定而产生的延迟抖动为 $[0 - d_i^{k-1}]$ 范围;另外一部分专门用于平滑前面所有节点允许的提前时间,该值是逐节点增加,后续节点的缓冲空间限制将降低前面节点的吞吐能力。

3.2 改进的算法 PD-EDF

当每个节点的带宽都满足公式(2)的情况下,节点 k 为流 i 分配的缓冲区包含两个部分:用于平滑节点 $k-1$ 调度抖动的缓冲空间为 $bs_i^k = d_i^{k-1} * \max(A_i^*(t))$;用于平滑提前时间累积的部分 b_j^k ,而 b_j^k 只需要满足条件 $0 \leq b_j^k \leq +\infty$ 就可以。当节点 k 在时间 $t_{i,j}^k$ 收到数据包 j ,根据 $k+1$ 节点的 b_j^{k+1} 来计算 $\varepsilon_{i,j}^k$ 的取值范围: $\min(b_j^{k+1}/\max(A_i^*(t)), e_{i,j}^k) \geq \varepsilon_{i,j}^k \geq 0$, 其中 $e_{i,j}^k = D_{i,j}^{k-1} - t_{i,j}^k$ 。端到端经过 h_i 个节点,每个节点 k 提供最大延迟保证 d_i^k 则可以精确保证端到端的延迟上界与公式(3)相同,端到端的延迟下界为:

$$D_i^{lower} = d_i^{shmin} + \sum_{k=1}^{h_i-1} d_i^k \quad (6)$$

上下界的差值最大为: $d_i^{h_i} + d_i^{shmax} - d_i^{shmin}$ 。

Jitter-EDD 和 JT-EDF 都证明了在不考虑缓冲区的情况

下,只要节点带宽符合公式(2),每个节点都能够保证延迟上界。下面将证明上述算法能够保证缓冲区不溢出和端到端的延迟上下界。

推理1 节点 k 能提供用于调度抖动的 bs_i^k 是算法正常运行的必要条件。

用反证法:假设节点 k 分配的缓冲空间可以少于 $d_i^{k-1} * \max(A_i^*(t))$,则有:

因为第一个节点是按照 $A_i^*(t)$ 输出,可以合理假设所有节点最先都是按照 $A_i^*(t)$ 和最大延迟 d_i^n 转发数据包。这个时候,节点 $k-1$ 内聚集的数据量最大为 $d_i^{k-1} * \max(A_i^*(t))$ 。因为考虑到节点 $k+1$ 的缓冲区少于该数值,所以节点 $k-1$ 不能让所有的数据包都无延迟地进入调度队列,即不能让所有的数据包在抖动控制器中被先行延迟 $\Delta d_{i,j} = e_{i,j}^k - \varepsilon_{i,j}^k$ 。如果在包 j 到达队列之后,包括 k 在内的每个节点的输出调度都处于满负荷状态,只能以最大保证延迟 $d_{i,j}^n (n \geq k)$ 输出,那么该数据包 j 到达终端的总延迟时间将比最大延迟时间大 $\Delta d_{i,j}$ 。所以这种情况下,无法保证端到端延迟上界。推理1 得到证明。

推理2 当缓冲区 bs_i^k 满足条件后,节点 k 分配任意大小的缓存为 b_j^k 。对于经过该节点的数据包 j ,根据算法灵活设置其到达时间提前量为 $\varepsilon_{i,j}^k$,不会导致缓冲区溢出,且任何一个节点都具有将数据流整形为 $A_i^*(t)$ 的能力。

采用归纳法证明:

1) 因为节点 1 是通过 $A_i^*(t)$ 整形后进入到输出调度队列的,它没有任何提前到达的数据包需要发送到节点 2 中去。只要节点 2 具有基本的缓冲区 bs_i^2 ,就可以将节点 1 来的抖动数据流重新恢复成符合 $A_i^*(t)$ 要求的流,且缓冲区不溢出。

2) 假设到节点 k ,推理2 成立,缓冲区不溢出,且能将数据流整形成为符合 $A_i^*(t)$ 的流,则有:节点 k 在 $t_{i,j}^k$ 时刻收到节点 $k-1$ 转发来的数据包 j ,当将数据包 j 输出到节点 $k+1$ 时,到达节点 $k+1$ 的时间 $t_{i,j}^{k+1}$ 符合条件: $t_{i,j}^k + e_{i,j}^k - \varepsilon_{i,j}^k \leq t_{i,j}^{k+1} \leq t_{i,j}^k + e_{i,j}^k + d_i^k$ 。根据算法中 $\varepsilon_{i,j}^k$ 取值范围的计算方法可得: $t_{i,j}^k + e_{i,j}^k - \min(b_j^{k+1}/\max(A_i^*(t)), e_{i,j}^k) \leq t_{i,j}^{k+1} \leq t_{i,j}^k + e_{i,j}^k + d_i^k$ 。

1) 首先证明:该数据包 j 以最小延迟到达节点 $k+1$ 不会导致其缓冲溢出。

当包 j 以最小延迟 $a_{i,j}^{k+1} = t_{i,j}^k + e_{i,j}^k - \min(b_j^{k+1}/\max(A_i^*(t)), e_{i,j}^k)$ 到达节点 $k+1$ 时, $k+1$ 节点收到的总的最大提前数据量为:

$$b_{need}^{k+1} = (D_{i,j}^k - a_{i,j}^{k+1}) * \max(A_i^*(t)) \quad (7)$$

根据 $e_{i,j}^k = D_{i,j}^{k-1} - t_{i,j}^k$, $b_{need}^{k+1} = d_{i,j}^k * \max(A_i^*(t)) + b_j^{k+1}$ 。

因为上式前面部分缓冲由 bs_i^{k+1} 提供,则有: $b_{need}^{k+1} \leq bs_i^{k+1} + b_j^{k+1}$,所以节点 $k+1$ 的缓冲能够容纳节点 k 按算法发送来的任何数据包,缓冲不会溢出得到证明。

2) 证明数据包 j 以最慢速度(最大延迟)到达 $k+1$,不导致延迟超过上限 $D_{i,j}^k$ 。

由 $t_{i,j}^{k+1} \leq t_{i,j}^k + e_{i,j}^k + d_i^k$ 和 $e_{i,j}^k = D_{i,j}^{k-1} - t_{i,j}^k$, 可得: $t_{i,j}^{k+1} \leq t_{i,j}^k + D_{i,j}^{k-1} - t_{i,j}^k + d_i^k = D_{i,j}^k$ 。数据包 j 到达节点 $k+1$ 的延迟上界能够得到保证。

由 1) 和 2) 得:节点 $k+1$ 也能在缓冲区不溢出的情况下,将流按照 $A_i^*(t)$ 的限制输出。至此,推理2 得证。根据推理2,最后一个节点也具有将数据流整形为 $A_i^*(t)$ 的能力。因为是最后一个节点,数据流将被 $A_i^*(t)$ 发送到输出队列。所以端到端的延迟上界和下界分别为公式(5)和(3)得证。实际上,如

(下转第 1545 页)

突的限制越严格,就要以增加能量消耗为代价。总体来看,与 STPS 算法相比,ECCA 算法总传输功率增加的量很小。

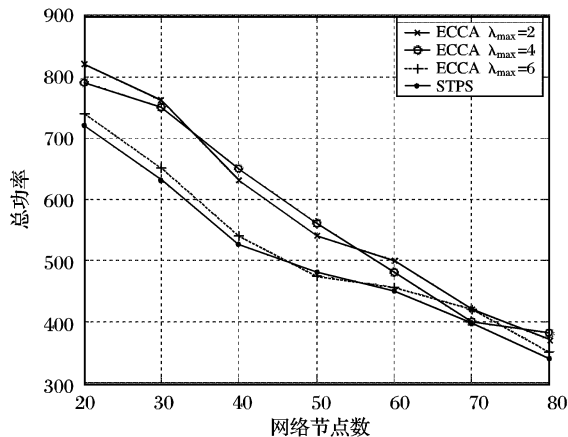


图6 消耗的总功率比较

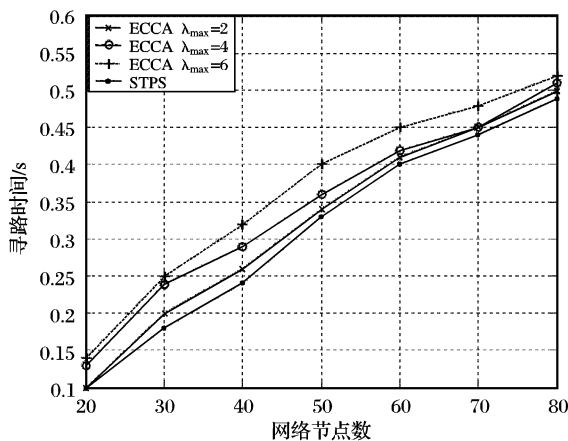


图7 寻路时间比较

通过仿真,比较了 ECCA 算法和 STPS 算法的寻路时间,仿真结果如图7所示。由于 ECCA 算法为了满足特定的冲突限制,在算法中要加入一些关于冲突因子的计算,因此要花费较长的寻路时间。然而,ECCA 算法在 λ_{\max} 分别为 2,4,6 时的寻路时间和经典的最小能量路由算法 STPS 相比,分别最多

增加了 0.02s,0.06s 和 0.07s。因此,ECCA 算法的寻路时间还是比较理想的。

参考文献:

- [1] SINGH S, RAGHAVENDRA CS. PAMAS: Power Aware Multi-Access Protocol with Signaling for Ad Hoc Networks[J]. ACM Computer Communication Review, 1998, 28(3): 5-26.
- [2] IEEE Std. 802.11, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications[S]. IEEE Computer Society LAN MAN Standards Committee, 1998.
- [3] CHEN B, JAMIESON K. Span: An Energy-efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks [A]. MOBICOM 2001[C], 2001.
- [4] ZHENG R, KRAVETS R. On-demand Power Management for Ad Hoc Networks[A]. IEEE INFOCOM'2003[C], 2003.
- [5] TOH C-K. Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks[J]. IEEE Communications Magazine, 2001, 39(6): 2-11.
- [6] TOH C-K, COBB H, SCOTT DA. Performance Evaluation of Battery-life-Aware Routing Schemes for Wireless Ad Hoc Networks[A]. Proceedings of IEEE International Conference on Communications (IEEE ICC)[C]. Finland, 2001.
- [7] SRINIVAS A, MODIANO E. Minimum energy disjoint path routing in wireless ad hoc networks[A]. Proceedings of ACM Mobicom'2003 [C], 2003. 122-133.
- [8] TANG J, XUE G. Node-disjoint path routing in wireless networks tradeoff between path lifetime and total energy[A]. IEEE International Conference on Communications (ICC'2004)[C], 2004. 3812-3816.
- [9] MALEKI M, DANTU K, PEDRAM M. Power-aware Source Routing protocol for Mobile Ad hoc Networks[A]. Proceedings of the 2002 International Symposium on Low power Electronics and Design[C], 2002.
- [10] BANERJEE S, MISRA A. Minimum Energy paths for Reliable Communication in Multi-hop Wireless Networks[A]. MobilHoc [C], 2002. 146-156.
- [11] SUBBARAO MW. Dynamic Power Conscious Routing for MANETs: An Initial Approach[A]. IEEE Vehicular Technology Conference[C], 1999. 1232-1237.

(上接第 1541 页)

果终端具有更多的缓冲能力,也可以作为最后一个网络节点参与上述的计算,而得到充分的利用。

3.3 本算法性能分析

根据上面的分析,原算法中 $\varepsilon_{i,j}^k$ 控制着数据包对节点带宽的需求的紧迫程度,宽范围的取值意味着节点有更高的资源利用率和整体性能。本文提出的 $\varepsilon_{i,j}^k$ 计算方法不但提高了各个节点独立的缓冲区分配的灵活性,而且尽可能增加了 $\varepsilon_{i,j}^k$ 的取值范围,提高了节点数据转发效率。

因为节点的流量通常具有自相关性,其流量变化具有一定的可预测性。而前面的算法中,节点 k 给流 i 分配的缓冲空间 b_j^k 是按照满负荷的情况下分配的。当节点比较空闲的时候,其富余缓冲和带宽可以更多地提供给正在经过的流。通过动态增加 b_j^k ,并通知节点 $k-1$;节点 $k-1$ 可以对收到的数据包计算出来的 $\varepsilon_{i,j}^k$ 将会有更大的取值范围,能够进一步提高节点的吞吐量和服务质量。

参考文献:

- [1] 林闯,单志广,任丰原. 计算机网络的服务质量(QoS)[M]. 北京:清华大学出版社,2004.

- [2] ZHU K, ZHUANG Y, VINIOTIS Y. Achieving End-to-end Delay Bounds by EDF Scheduling without Traffic Shaping[A]. Proceedings of IEEE INFOCOM[C], 2001.
- [3] GEORGIADIS L, GUERIN R, PAREKH A. Optimal Multiplexing on a Single Link: Delay and Buffer Requirements[J]. IEEE/ACM Transactions on Information Theory, 1997, 43(5): 1518-1535.
- [4] GEORGIADIS L, GUERIN R, PERIS V, et al. Efficient network QoS provisioning based on per node traffic shaping[J]. IEEE/ACM Transactions on Networking, 1996, 4(4): 482-501.
- [5] VERMA D, ZHANG H, FERRARI D. Delay jitter control for real-time communication in a packet switching network[A]. Proceedings of TriComm 91[C]. Chapel Hill, 1991.
- [6] FERRARI D, VERMA D. A scheme for real-time channel establishment in wide-area networks[J]. IEEE Journal of Selected Areas in Communications, 1990, 18(3): 368-379.
- [7] LIEBEHERR J, YILMAZ E. Work-conserving vs. non-workconserving packet scheduling: an issue revisited[A]. Proceedings of IEEE/IFIP WQoS'99[C], 1999.
- [8] 王振凯,刘斌,徐光祐. 基于 EDF 调度算法的端到端延迟保证方法[J]. 计算机工程与应用, 2002, 38(4).