文章编号:1001-506X(2010)02-0423-05

网络处理器中的快速 GTP 隧道检查方法

刘震宇,傅予力,赖 粤
(华南理工大学电子与信息工程学院,广东 广州 510641)

摘 要:针对网络处理器中快速存储资源有限和微引擎缺乏存储管理方法等制约实现快速、大量 GPRS 隧道协议(GPRS tunneling protocol,GTP)隧道检查的问题。提出了基于布隆过滤器(Bloom filter,BF)的数据隧道端点标识(data tunnel endpoint identifier Bloom filter,DTEID BF)方法,该方法将冲突问题转化为误判率问题,通过合理利用快速存储资源减少读写操作周期,采用并行运算提高处理速度,给出了应用该方法的具体步骤和参数选择方法。通过在 IXP2850 中实现 DTEID BF 方法并进行吞吐量和延时测试,证明该方法在吞吐量上接近 1 Gb/s,在最大延时和平均延时指标上优于 Trie 方法。

关键词:GPRS 隧道协议;网络处理器;布隆过滤器;微引擎
中图分类号:TP 393 **文献标志码**:A

Bloom filter method for checking GTP channels

LIU Zhen-yu, FU Yu-li, LAI Yue
(*School of Electronic and Information Engineering, South China Univ. of Technology, Guangzhou 510641, China*)

Abstract: To solve the problems that fast storage resources are limited and memory management methods are absence in micro-engines which restrict the network processor to achieve a large number of GTP (GPRS tunneling protocol) tunnels inspections quickly. DTEID BF method based on Bloom filter is put forward. DTEID BF method transforms the confliction problems into a discusse on false judgment, reduces storage operation by utilizing the fast storage rationally and adopts parallel computing to improve processing speed. The principle of parameters selection and detailed steps are introduced. Through experiments of throughput and latency in IXP 2850, the result shows that DTEID BF achieves nearly 1 Gb/s of processing speed. The maximum delay and average delay of DTEID BF are superior to those of Trie.

Keywords: GPRS tunneling protocol; network processor; Bloom filter; micro-engine

0 引 言

GPRS 隧道协议(GPRS tunneling protocol,GTP)是应用于3G/GPRS骨干网中 GSN 节点之间的重要接口协议。GTP 协议只是对 IP 包进行封装,不提供鉴权。因此,必须对 GTP 隧道状态进行检测[1-4]。

网络处理器是用来执行网络数据处理和转发的高速可编程处理器。网络处理器由于灵活性强等特点,给网络设备应用以及网络平台构建带来了极大的方便,被认为是下一代网络的核心技术之一。目前 3G 网络应用业务种类繁多,网络处理器为 GTP 隧道检查提供了一个灵活的硬件架构[5-6]。

GTP 隧道检查需要大量的存储资源和较多的读写操作,网络处理器存在着快速存储资源有限和微引擎中缺乏有效的内存管理等问题,使得许多包分类查找算法的使用受到限制。

网络处理器的 IXA(intel internet exchange)开发平台提供了用于静态路由查找的变步长 Trie 方法,该方法相对于基本的 Trie 方法加快了查找速度,缺点是存储中存在较多空树叶。该方法如果用来实现隧道检查存在以下问题:① 变步长 Trie 方法利用 Xscale 动态管理存储资源,需要和微引擎进行多次交互,在出现隧道建立请求洪水攻击时,微引擎和 Xscale 的频繁交互容易造成系统崩溃;② 如果采取静态分配存储空间的方法,存储资源必须按照最大容量设计,对于十万条隧道的系统要求,需要分配较大的存储空间;③ Trie 查找过程为串行操作,如果存在冲突,需要进行多次匹配,性能明显下降。

本文针对 GTP 隧道的 GTP-U 包提出了基于布隆过

收稿日期:2009-02-23;**修回日期**:2009-04-30。
基金项目:教育部科技创新工程重大项目培育资金(707047)资助课题
作者简介:刘震宇(1976-),男,博士后,主要研究方向为智能信号、信息处理和网络安全。E-mail:zy.liu@mail.scut.edu.cn

滤器(bloom filter, BF)的数据遂道端点标识布隆过滤器方法(data tunnel endpoint identifier Bloom-filter, DTEID-BF)方法,BF 是一种多哈希函数的算法[8],近年来取得了令人满意的效果[9-10]。该方法通过调整参数满足指标要求,可以合理利用存储资源,减少存储操作,通过并行计算提高处理速度。在 IXP2850 中实现的 DTEID BF 方法可以达到接近 Gb/s 的处理速度,在吞吐量和延时方面优于 Trie 方法。

1 GTP 与网络处理器

1.1 GTP 概述

GTP 是应用于 3G/GPRS 骨干网中 GSN(GPRS surporting node)之间的重要接口协议[1-2]。GGSN(gateway GPRS support node)与 SGSN(serving GPRS support node)之间的接口首先对 GTP 协议进行处理,从而实现了从 GGSN 到 SGSN 的虚拟传输通路,即隧道。GTP 协议只对 IP 包进行封装,在 GGSN 和 SGSN 之间使用,由于 GTP 不提供鉴权,因此在鉴权和授权方面存在着威胁。所以,必须对 GTP 隧道进行检查,避免出现非法攻击[4]。

TEID(tunnel endpoint identifier)用于表示一条隧道,TEID 分为数据面 TEID 和控制面 TEID。TEID 由 SGSN 和 GGSN 接收端分配,长度为 4 Bytes[2-3]。图 1 表示 TEID 在隧道建立、删除和传输过程中 GTP 包内 TEID 相关字段的变化。

图 1 GTP 隧道中 TEID 变化示意图

由图 1 可见,TEID 包括了上下行控制 TEID 和上下行数据 TEID 四个值,本文作如下标记:

上行控制 TEID——uplink control TEID(UCTEID);

下行控制 TEID——downlink control TEID(DCTEID);

上行数据 TEID——uplink data TEID(UDTEID);

下行数据 TEID——downlink data TEID(DDTEID)。

在 GTP 隧道建立、拆除和传输过程中,GTP 数据包都会根据不同状态携带不同 TEID,TEID 直接反映隧道状态变化,因此将 TEID 作为检查 GTP 通信状态的参数。

1.2 网络处理器

IXP 系列的网络处理器由于强大的灵活性受到各方面研究的重视。IXP2850 网络处理器具有 16 个 1.4 GHz 的微引擎,每个微引擎有 8 个硬件线程,并且有一个 700 MHz 的处理核心 Xscale[5]。Scratch、SRAM 和 DRAM 构成了网络处理器的三大存储资源,如表 1[6]。

表 1 网络处理器中主要存储资源比较

存储器	实际应用空间	片内存储	读速度(指令周期)	写速度(指令周期)
Scratch Pad	16 KB	是	100	40
SRAM	32 MB	否	130	53
DRAM	704 MB	否	295	53

表 1 中,Scratchpad 速度最快,但仅为 16 KB,常用于重要数据结构的缓存。SRAM 的速度与 Scratch 接近,容量较大,但是 SRAM 中的许多资源被系统的占用,例如路由表占用了约 6 MB 的 SRAM 空间。DRAM 的容量最大,但读取速度最慢。

2 GTP 隧道检查方法

2.1 BF

定义 1 BF 用三元组$\{n,m,k\}$来标识,n 表示集合 S 中元素个数,m 表示向量 $\boldsymbol{V}$ 长度,k 表示哈希函数个数。bloom filter 用一个长为 m 的位向量 $\boldsymbol{V}$ 来表示一个域 U 上 n 个元素的集合 $S=\{s_1,s_2,\cdots,s_n\}$,位向量 $\boldsymbol{V}$ 初始化设置为全 0。使用 k 个独立的取值范围在$\{0,\cdots,m-1\}$的哈希函数$\{h_1,h_2,\cdots,h_k\}$,这些哈希函数对于 $\forall s_i\in S,1\leqslant i\leqslant n$ 的元素都映射到位向量 $\boldsymbol{V}$ 上,即 $\forall s_i\in S,1\leqslant i\leqslant n$,位向量 $\boldsymbol{V}$ 上对应于 $h_j(s_i)$ 的比特位都设置为 1,其中 $1\leqslant j\leqslant k$。

根据 BF 的定义,从三个方面考虑基于该算法建立 GTP 隧道检查方法。

首先,在存储方面,位向量利用比特位表示规则,可以充分利用快速存储空间,节省存储资源。

其次,在存储操作方面,BF 匹配时,只对位向量操作,如果位向量在快速存储空间,可以提高处理速度,并且,网络处理器提供位指令加速位操作。

最后,在并行计算方面,尽管 BF 的时间复杂度为 $O(k)$,但是 k 个哈希运算之间互相独立,且微引擎具有多个硬件线程,可以利用并行处理减少运算时间。

2.2 DTEID BF 结构

检查图 1 中的 GTP-U 包时只需考虑上下行包中的 UDTED 或 DDTEID 是否为有效的 TEID,将上下行隧道的数据 TEID 统称为 DTIED,引入 BF 进行处理。因此,处理上下行 GTP 包中 TEID 的方法命名为 DTEID BF 方法。DTEID BF 方法的实现框图如图 2 所示。

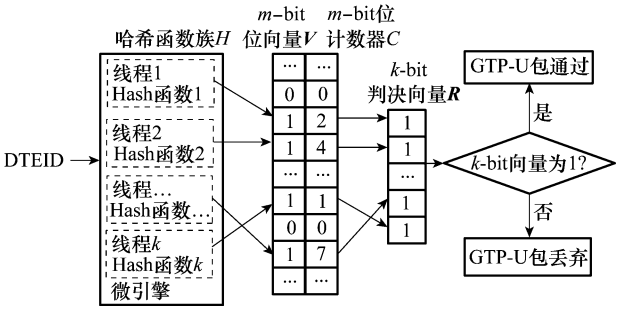

图 2　DTEID BF 框图

DTEID BF 方法由哈希函数簇 H、m-bit 位向量 $\boldsymbol{V}$、位向量计数器 C 和 k-bit 判决向量 $\boldsymbol{R}$ 四部分组成。

哈希函数簇 $H\{h_1,h_2,\cdots,h_k\}$:由多个相互独立的哈希函数组成,$0\leqslant h_j\leqslant m-1$,$1\leqslant j\leqslant k$,每个哈希函数可以在微引擎的一个线程中运行。网络处理器提供了类似多项式除留余数法的硬件哈希单元,利用哈希单元可以构建多个哈希函数。

m-bit 位向量 $\boldsymbol{V}\{v_1,v_2,\cdots,v_m\}$:用比特位记录有效的 DTEID,当 $h_j(\text{DTEID})=l$,$0\leqslant l\leqslant m-1$ 时,有 $v_l=1$。当 $v_l=0$ 时,表示无有效 DTEID 对应。$\boldsymbol{V}$ 存储在速度较快的 SRAM 中,初始化时,$\boldsymbol{V}$ 为全 0。

位向量计数器 $C\{c_1,c_2,\cdots,c_m\}$:记录位向量 $\boldsymbol{V}$ 各比特位中累积的有效 DTEID 数量。当 $c>0$ 时,表示该位有多个有效 DTEID 对应,当 $c=0$ 时,表示无有效 DTEID 对应。C 存储在容量较大的 DRAM 中,初始化时,C 为全 0。

k-bit 判决向量 $\boldsymbol{R}\{r_1,r_2,\cdots,r_k\}$:用于判断 DTEID 是否有效,当 $h_j(\text{DTEID})=l$,$v_l=1$ 时,有 $r_j=1$,表示该 DTEID 对应于哈希函数 h_j 是有效 DTEID;当 $h_j(\text{DTEID})=l$,$v_l=0$ 时,有 $r_j=0$,表示该 DTEID 对应于哈希函数 h_j 是无效 DTEID。如果 $\forall r_j\in R$,$r_j=1$,$1\leqslant j\leqslant k$,表示待查找的 DTEID 为有效。R 存储在微引擎的内部寄存器中,初始化时,R 为全 0。

2.3　DTEID BF 操作

在 DTEID BF 方法中分为添加、删除和查找三种操作,对应 GTP 隧道中的建立、拆除和传输等过程。以下是这三种操作的伪码:

添加 DTEID
for $j=1$ to k\{　　//括弧内步骤可并行运算
(1) $l=h_j(\text{DTEID})$;　//计算 DTEID 的哈希值 l
(2) $v_l=1$;　　//向量 $\boldsymbol{V}$ 的第 l 位置为 1
(3) $c_l=c_l+1$;　　//向量计数器 C 的第 l 位累加 1\}

删除 DTEID
for $j=1$ to k　{　　//括弧内步骤可并行运算
(1) $l=h_j(\text{DTEID})$;
(2) $c_l=c_l-1$;　　//向量计数器 C 的第 l 位减 1
(3) if $c_l==0$ then $v_l=0$//如果计数器 C 的第 l 位为 0,则向量 V 的第 l 位置为 0\}

查找 DTEID
(1) for $j=1$ to k $r_j=0$;　//判决向量 $\boldsymbol{R}$ 置 0
for $j=1$ to k　　{　　//括弧内步骤可并行运算
(2) $l=h_j(\text{DTEID})$;
(3) if $v_l==1$ then $r_j=1$//如果向量 $\boldsymbol{V}$ 的第 l 位为 1,则向量 $\boldsymbol{R}$ 的第 j 位为 1\}
(4) if($\forall r_j\in\boldsymbol{R}$,$(1\leqslant j\leqslant k,r_j=1)$then(Pass)else (Drop)
//如果向量 $\boldsymbol{R}$ 全为 1,则 GTP 包通过,否则丢弃

以上操的部分步骤可以放在一个微引擎的 k 个线程中处理,提高对 GTP-U 包的检查速度。

2.4　DTEID BF 参数分析

2.4.1　正误判率

定义 2　正误判率

存在一定的概率 p,将不属于 S 中的元素 e 判断为属于 S。

BF 的引入会带来一定的误判率,但是不会出现漏判事件,即把本属于 S 中的元素 s 判断为不属于 S。用 p 表示向量 $\boldsymbol{V}$ 中一位为 0 的概率,则 $1-p$ 为 1 的概率。假设哈希函数取值服从均匀分布,当集合中所有元素都映射完毕后,$\boldsymbol{V}$ 向量任一位为 0 的概率为

$$p'=\left(1-\frac{1}{m}\right)^{kn}\approx \mathrm{e}^{-kn/m}=p \tag{1}$$

不属于集合的元素误判属于该集合时,元素的误判率为

$$f^{\mathrm{BF}}(m,k,n)\approx(1-p)^k=(1-\mathrm{e}^{-kn/m})^k=\exp(k\ln(1-\mathrm{e}^{-kn/m})) \tag{2}$$

令 $g(k)=k\ln(1-\mathrm{e}^{-kn/m})$,将 k 视为连续的实数进行求导,得

$$\frac{\mathrm{d}g(k)}{\mathrm{d}k}=\ln(1-\mathrm{e}^{-kn/m})+\frac{kn}{m}\times\frac{\mathrm{e}^{-kn/m}}{1-\mathrm{e}^{-kn/m}} \tag{3}$$

令 $\dfrac{\mathrm{d}g(k)}{\mathrm{d}k}=0$,从式(3)得 $k=k_{\min}=\ln 2\left(\dfrac{m}{n}\right)$ 时,在 $k_{\min}$ 附近取一个整数,可以得到一个近似最优解,使 g 达到最小,这时 $p=(1-p)=1/2$,误判率最小值为

$$f^{\mathrm{BF}}(k_{\min})=\left(\frac{1}{2}\right)^{k_{\min}}=\left(\left(\frac{1}{2}\right)^{\ln 2}\right)^{\frac{m}{n}} \tag{4}$$

同样,当指定了哈希函数的个数 k,需要计算位向量的长度 m,由式(3)可以得到

$$m=-\frac{kn}{\ln(1-(f^{\mathrm{BF}})^{1/k})} \tag{5}$$

式(5)中,将 k 视为连续值,对 m 取 k 的导数,得

$$\frac{\mathrm{d}m(k)}{\mathrm{d}k}=-\frac{\ln(1-(f^{\mathrm{BF}})^{1/k})\cdot n-n\cdot\dfrac{1}{1-f^{\mathrm{BF}}}(f^{\mathrm{BF}})^{1/k}\cdot\ln((f^{\mathrm{BF}})^{1/k})}{(\ln(1-(f^{\mathrm{BF}})^{1/k}))^2} \tag{6}$$

当式(6)为 0 时,哈希函数个数 k 是 n 和 f^{BF} 固定的情况下,使得 m 最小的 k 值,k 取整数。

2.4.2　参数讨论

由于 BF 存在一定的正误判,所以必须考虑 DTEID BF 方法的参数 k 和 m 值,使得 DTEID BF 在高速处理的同时满足系统指标。系统要求能对十万条 GTP 隧道进行检查,满足线速处理的要求,而且 $f^{\mathrm{BF}}\leqslant 10^{-6}$。

在微引擎的 8 个并行硬件线程中，每个线程可以运行一个哈希函数，如果哈希函数超过 8 个，会产生跨引擎操作，降低处理速度，而且微引擎中应保留一个线程负责协调，所以哈希函数个数 $k\leqslant 7$。对 k 值为 4～7 的误判率进行分析，如图 3 所示。

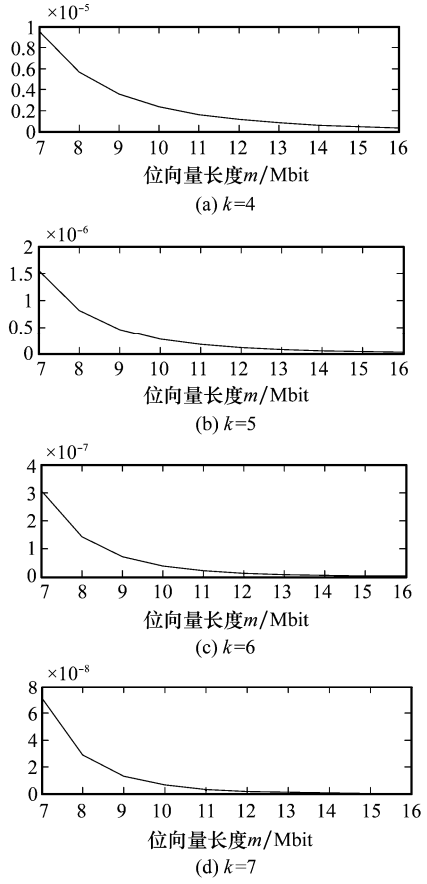

图 3 哈希函数个数 k 与误判率 f^{BF} 的分析

从图 3 可以见，当 $k=4$，$m=12$ Mbit；$k=5$，$m=8$ Mbit；$k=7$和 $k=8$，$m=7$ Mbit 时，误判率小于 10^{-6}。由此可见，在满足一定存储空间的要求下，$4\leqslant k\leqslant 7$ 可以在微引擎中实现，并达到系统要求。

2.5 DTEID BF 性能分析和比较

2.5.1 Trie 性能

由于微引擎没有操作系统提供存储管理方法，所以必须考虑变步长 Trie 的最坏情况复杂度。在最坏情况下，变步长 Trie 最坏情况下时间复杂度为 $O(W)$，其中 W 为步数。每步存储复杂度为

$$O(w_i)=\min\left(n,2^{\sum_{i=0}^{i-1}w_j}\right)\times 2^{w_i}\times\beta$$

式中，w_j 为每步步长，β 为存储下一节点地址入口的空间，在微引擎中要用 4 个字节存储节点地址，所以 $\beta=4$。由此可得，变步长 Trie 存储复杂度为

$$O(N,B,W)=\sum_{i=1}^{W}\min\left(n,2^{\sum_{j=0}^{i-1}w_j}\right)\times 2^{w_i}\times\beta \qquad (7)$$

式中，N 为字段数量；B 为字段长度，$B=\sum_{j=0}^{W}w_j$，$w_0=0$。变步长 Trie 方法在字段稀疏的时候，一次查找完成匹配的概率较大，所以 IXA 将第一步所需的空间分配在 SRAM 中。由于 16－4－4－4－4 Trie 方法的第一步匹配存在较高的冲突概率，所以优化后采用步长为 20－4－4－4 的 Trie 方法。表 2 对 16－4－4－4－4 和 20－4－4－4Trie 方法在最坏情况下的复杂度进行比较。

表 2 16－4－4－4－4 和 20－4－4－4 Trie 最坏情况复杂度比较

	16－4－4－4－4	20－4－4－4
时间复杂度 $O(W)$	5	4
第一步所需空间	256 KB	4 MB
其余步所需空间	25.6 MB	19.2 MB

由表 2 可见，尽管 20－4－4－4 第一步所需空间比 16－4－4－4－4大，但是由于第一步较长，所以冲突概率比 16－4－4－4－4 少。

2.5.2 DTEID BF 性能比较

由图 2 可知，位向量 $\boldsymbol{V}$ 的存储空间由长度 m 决定，$\boldsymbol{V}$ 以字节为单位。位向量存储空间表示为 V_{space}，有 $V_{\mathrm{space}}=\mathrm{ceil}(m/8)$ B。

计数器 C 的每一个值 c_l 对应一个 v_l 计数，所以每个 c_l 必须按照最大计数值考虑，即 $c_l\leqslant 100\ 000$。因此，一个 c_l 最少为 17 bit，为了节省存储空间，c_l 拆分为 c_l^H 和 c_l^L，c_l^H 为 c_l 的第 17 位值，c_l^L 为 c_l 的低 16 位值。计数器存储空间表示为 C_{space}，单位为字节，得

$$C_{\mathrm{space}}=C_{\mathrm{space}}^{H}+C_{\mathrm{space}}^{L}=ceil(m/8)+m\times 2 \qquad (8)$$

设 $m_{\min}$为满足 $f^{\mathrm{BF}}<10^{-6}$时的最小 m 值，根据式(8)，表 3 对 $\boldsymbol{V}_{\mathrm{space}}$和 C_{space}在 $4\leqslant k\leqslant 7$ 时进行比较。

表 3 $\boldsymbol{V}_{\mathrm{space}}$和 $\boldsymbol{C}_{\mathrm{space}}$在 $m_{\min}$，$4\leqslant k\leqslant 7$ 时的比较

	$k=4$	$k=5$	$k=6$	$k=7$
$m_{\min}$/Mbit	12	8	6	5
V_{space}/MB	1.5	1	0.75	0.625
C_{space}/MB	25.5	17	12.75	10.625
$f^{\mathrm{BF}}(m_{\min},k)$	0.8×10^{-6}	0.8×10^{-6}	0.7×10^{-6}	0.6×10^{-6}

从表 3 可见，在满足 $f^{\mathrm{BF}}<10^{-6}$的条件下，随着 k 的增加，$\boldsymbol{V}_{\mathrm{space}}$和 C_{space}所需的存储空间减少，但是所需的存储空间递减速度变慢。

DTEID BF 进行一次查找需要进行 k 次哈希运算，由于采用了并行运算，所以时间复杂度近似为 $O(1)$。插入和删除一个元素的操作的时间复杂度也为 $O(1)$。存储空间由 m 确定，$O(m)=\boldsymbol{V}_{\mathrm{space}}+C_{\mathrm{space}}$。

通过对表 2 和表 3 进行比较可以看出 DTEID BF 在存储和处理速度上相对 Trie 方法的优势：

首先，在时间复杂度方面，DTEID BF 的时间复杂度由哈希函数簇 H 中最复杂的哈希函数决定，只要哈希函数只对寄存器操作，不存在外部存储器读取，就会与 Trie 的无冲突情况时的时间复杂度接近，远远优于 Trie 方法最坏情况时的时间复杂度。

其次，在外部存储读取方面，DTEID BF 在匹配中只读

取一次 SRAM,Trie 方法如果在匹配中出现冲突,就要访问 DRAM。由表 1 可见,DRAM 的读取周期多于 SRAM 约 3 倍。因此,DTEID BF 避免了 DRAM 的操作,节省了处理周期。

最后,在存储空间方面,由于 Trie 方法存在较多的空树叶,所以在存储空间的使用上不如 DTEID BF 紧凑。从表 2 和表 3 比较可见,随着哈希函数个数增加,DTEID BF 所需的空间远低于 Trie。

由此可见,DTEID BF 将冲突问题转化为误判率问题,以此减少存储空间并获得较快的速度。

3　实验分析

实验主要测试表 2 中 20－4－4－4 的 Trie 方法,表 3 中 $k=4$ 和 $k=5$ 的 DTEID BF 对 GTP-U 包的处理速度。测试设备采用思博伦的 TestCenter 高密度综合数据网络测试平台,对各种方法处理 GTP-U 包的吞吐量和延时进行测试。

TestCenter 根据实际应用中获得的一定范围内的 DTEID随机选取,分别产生 64 B,128 B、256 B、512 B 和 1 024 B的 GTP-U 包。TestCenter 从 1 Gb/s 的 80%开始测试最大吞吐量和延时,步进 5%,最后达到 100%,如果出现丢包,则以 1%步进测试吞吐量和延时。TestCenter 与网络处理器用光口连接,GTP-U 包由微引擎负责处理。吞吐量测试结果见表 4,64 B 包的延时测试结果见表 5。

表 4　吞吐量测试结果　　Mb/s

数据包	DTEID BF ($k=4$)	DTEID BF ($k=5$)	Trie(20－4－4－4)
64 B	970	950	860
128 B	980	970	890
256 B	990	980	920
512 B	1 000	990	950
1 024 B	1 000	1 000	960

从表 4 可见,Trie 方法在吞吐量上略低于 DTEID BF 方法,尽管在实现 Trie 方法的时候使用了线程隐藏方法减少存储读取的影响,但是由于 DRAM 的读取时间较长,影响了 Trie 的处理速度。

表 5　64 B 包延时测试结果　　μs

	DTEID BF ($k=4$)	DTEID BF ($k=5$)	Trie(20－4－4－4)
最小延时	9.27	9.5	9.01
最大延时	73.51	85.45	414.02
平均延时	19.22	22.13	231.75

从表 5 可见,Trie 方法的最小延时优于 DTEID BF 方法,但是最大延时和平均延时非常大。这是因为在出现冲突的时候,Trie 方法需要多次读取 DRAM 的数据,造成了较大的延时,而 DTEID BF 只对 SRAM 进行操作,所以保证了快速、稳定的处理速度。

4　结　论

本文通过分析 GTP-U 包在网络处理器上存储资源有限和微引擎缺乏存储管理等问题,提出了基于 BF 的 DTEID BF 方法,该方法一方面通过并行处理提高处理速度;另一方面,分离快速和慢速匹配的存储要求,使匹配过程在快速存储资源中完成,减少存储速度的影响。该方法通过牺牲少量正误判率,减少快速存储资源的使用,提高了对 GTP-U 的处理速度,该方法对类似问题的处理有一定的借鉴意义。

参考文献:

[1] 文志成. GPRS 网络技术[M]. 北京:电子工业出版社,2005:70－74.

[2] 3GPP. GPRS tunneling protocol (GTP) across the Gn and Gp interface [Z]. TS29.060. v3.3.0, 2000:14－30.

[3] 3GPP. Technical specification. 3rd generation partnership project; technical specification group services and system aspects [Z]. TS 23.060. V5.2.0, 2002:25－31.

[4] Hung H N, Lin Y B. Connection failure detection mechanism of UMTS charging protocol[J]. *IEEE Trans. on Wireless Communication*, 2006,5:1180－1186.

[5] 张宏科,苏伟,武勇. 网络处理器原理和技术[M]. 北京:北京邮电大学出版社,2004:112－130.

[6] Intel Crop. *Intel IXP2850 network processor, Hardware reference manual*[Z]. Intel, 2004:1－3.

[7] Hyesook L, Hye R K, Yeo J J. Parallel multiple hashing for packet classification[C]// *Proc. of High Performance Switching and Routing*, USA, 2005:104－107.

[8] Bloom B. Space/time trade-offs in hash coding with allowable errors[J]. *Communications of the ACM*, 1970, 13(7):422－426.

[9] Andrei B, Michael M. Network applications of bloom filters: A survey internet math[J]. *International Journal Mathematics*, 2003, 1(4):485－509.

[10] Dharmapurikar S, Krishnamurthy P, Sproull T, et al. Deep packet inspection using parallel Bloom filters[C]// *Proc. of the Symposium. On High Performance Interconnects*, 2003:44－51.