OpenFlow 交换机动态共享限速机制的研究

杨 骥 许 琛 龚志敏 胡成臣 管晓宏
(西安交通大学智能网络与网络安全教育部重点实验室 西安 710049)

摘 要 每流限速一直以来都是互联网服务质量保障(QoS)的一大挑战,由于受到硬件资源的限制,很难找到一种快速高效的算法在数据包级对流量进行区分限速. OpenFlow1.3 协议对每流限速(Meter)部分,给出了具体定义和描述,该文参照该协议标准,提出了具体的 Meter 实现方案,并在 ONetSwitch 硬件平台上,进行了实验验证和性能测试. 本文通过对网络流量的统计分析,发现网络中共存的流数量往往只占总流量的万分之一,因此使多条流分时共享硬件资源,可以有效解决每流限速的难题. 参考动态队列共享(DQS)的思想,该文提出动态共享限速(DMS)算法,在多条流之间进行调度,实现多条流分时共享限速通道,在硬件资源受限,限速通道数量一定,同时需要限速的流数量大于限速通道数量时,能够有效地实现对各条流进行限速的功能. 实验结果表明,Meter 能够为每条流有效分配限速通道,从而在网络中对多条流以不同的速率区分限速.

关键词 OpenFlow 1.3; ONetSwitch; QoS; 每流限速; 动态共享限速; 软件定义网络; 下一代互联网
中图法分类号 TP393 **DOI 号** 10.11897/SP.J.1016.2016.01224

A Mechanism of Dynamic Meter Sharing on OpenFlow Switch

YANG Ji Xu Chen GONG Zhi-Min HU Cheng-Chen GUAN Xiao-Hong
(*Intelligent Network and Network Security Key Laboratory of Ministry of Education, Xi'an Jiaotong University, Xi'an* 710049)

Abstract Per-flow rate-limiting is one of quality of the service (QoS) challenge in networking system. It is difficult to find a method that is both quick and efficient to manage flows which need to be limited to different rates due to the limitation of hardware resources. However, in OpenFlow-1.3 specification, the Meter part describes the definition of per-flow rate limiting. According to the description of Meter, this paper provides a detailed implementation of Meter based on ONetSwitch hardware platform and verifies the performance through multiple groups of experiment. Through the statistic of flows in network, there are only 100 active flows in million network in-progress flows. Therefore, the per-flow rate limiting problem may be resolved through multiple flows time sharing hardware resources. Referencing the idea of Dynamic Queue Sharing (DQS), Dynamic Meter Sharing (DMS) was proposed in this paper to schedule among different flows that sharing the same rate limiting pipe, but during different period. However, the number of rate limiters is limited due to the limitation of hardware resources. When flow numbers exceed the number of rate limiters, it is unavoidable to allocate some flows to a shared rate limiter waiting for idle ones. Although the critical "per-flow rate limiting" is not achieved, Meter has an advantage over traditional rate limiting method. The results show that Meter allocate an independent rate limiter for per-flow, then realize limiting multiple flows in different rate.

收稿日期:2014-12-16;在线出版日期:2015-07-23. 本课题得到国家自然科学基金(61221063,61272459,U1301254)、国家"八六三"高技术研究发展计划项目基金(2012AA011003)、教育部新世纪人才计划(NCET-13-0450)、江苏省未来网络课题(BY2013095-1-12)及中国电子科技集团公司第五十四研究所课题(ITD-U14001/KX142600008)资助.**杨 骥**,男,1987 年生,博士研究生,主要研究方向为软件定义网络与网络安全. E-mail: yangji@sei.xjtu.edu.cn. **许 琛**,女,1992 年生,硕士研究生,主要研究方向为软件定义网络. **龚志敏**,男,1990 年生,硕士研究生,主要研究方向为软件定义网络. **胡成臣**(通信作者),男,1981 年生,博士,副教授,博士生导师,中国计算机学会(CCF)会员,主要研究领域为数据中心网络与软件定义网络. E-mail: chengchenhu@mail.xjtu.edu.cn. **管晓宏**,男,1955 年生,博士,教授,博士生导师,主要研究领域为复杂网络优化与网络安全.

Keywords　OpenFlow 1.3；ONetSwitch；QoS；per-flow rate limiter；dynamic meter sharing；software defined networking；next-generation Internet

1　引　言

随着网络中协议的多样性趋势，交换机-路由器等设备的复杂度正在逐渐增加，而传统的网络架构由于诸多问题，如缺乏灵活性、对需求变化的响应速度缓慢、无法实现网络虚拟化以及成本高昂等，已不能满足当今互联网用户的需求. 因此，迫切需要新型网络架构的提出，从而为互联网带来一场革命性的变革. 软件定义网络(Software Defined Networking，SDN)的概念应运而生，并且经过近些年的发展与成熟，它正在成为下一代互联网的主要趋势之一.

美国斯坦福大学 Clean Slate 研究计划于 2006 年提出了 SDN 的概念，将控制功能从网络交换设备中分离出来，即用软件定义网络，使得网络可编程化，在满足用户的需求方面具有很大的灵活性. 同时，SDN 还简化了网络基础设备的工作，底层无需执行冗杂的协议标准，只需要从上层接收指令和规则完成转发即可[1]. SDN 实现了对网络流量的灵活控制，为核心网络及应用的创新提供了良好的平台，可满足网络设备上 QoS、访问控制和路径选择等需求[2].

SDN 的 3 层架构从上到下依次为应用程序、控制平面和数据平面，其中应用程序与控制平面之间通过北向应用程序编程接口(API)通信，控制平面和数据平面之间通过南向接口安全通道通信. 网络编程人员只需在逻辑中心化的控制平面上编写应用程序，而将规则下发到数据平面的复杂工作则由控制平面来完成，从而在很大程度上减轻了网络管理人员的工作负担[2-3].

在 SDN 架构中，OpenFlow 协议最早被定义，也是当前 SDN 设备通信的事实标准，工作在网络设备控制层面与数据层面之间[4]，主要负责南向接口，即 SDN 的控制器与交换机之间的通信，交换机中的流表用于匹配进入网络中的流，控制器可以增加、修改和删除流表项的内容，实时更新并下发规则到流表，如图 1 所示. 在流表中找到匹配项的流按照流表中安装的规则执行相应的丢弃或转发动作，否则产生未匹配流表的消息，将其转发到控制器处理①.

OpenFlow 支持精确的流分类，并允许网络编程对每流进行操作，为网络提供细粒度的控制，使网络实时响应应用、用户及会话层的请求，已经被大量应用到各种网络交换设备和应用软件上[5]. 利用 SDN 的 OpenFlow 技术，可以解决很多传统网络中难以解决的问题，例如配置网络设备路由、访问控制和防火墙等复杂功能[6-7]，实现信息中心网络(ICN)[8]，解决互联网中负载均衡的问题[9]. 在本文中，将详述利用 OpenFlow 对流表中的每条表项按照控制器下发的规则执行不同操作的优势，实现对网络中的各条突发流以不同速率区分限速.

在版本号高于 1.3 的 OpenFlow 协议中，针对网络服务质量保障(QoS)支持，新增了每流限速(Meter)的功能，即实时测量网络中各流的速率，并对指定流进行速率控制. 任何一个流表项都可以在其指令动作中声明其对应的限速指令，当到达的数据包匹配该流表项后，需要继续通过 Meter 以指定速率对流量进行控制，满足要求后，数据包才会进行转发，以达到限速的目的②.

图 1 所示为 Meter 在 SDN 网络中的部署，控制器具有全局视图，可对网络中的交换机进行配置. 控制器通过 OpenFlow 协议下发流表项和限速参数到交换机，对匹配了某条流表项的流按照相应的参数限速并转发，否则执行指令域定义的其他动作.

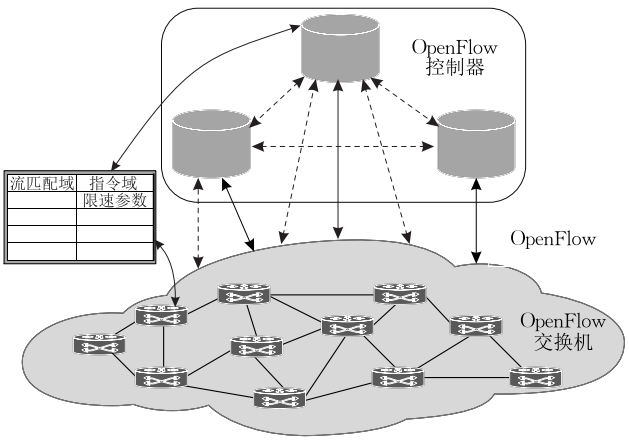


图 1　Meter 在 SDN 网络中部署

每流限速的前景毋庸置疑，相较于传统的限速技术有着很大的优势. 由于网络中一段时间内只有部分流会出现突发传输的现象，只有在这种情况下，才需要对这些流量的速率进行一定限制，以避免湮没下行节点. 然而，突发传输的流往往仅占网络总流

① Software-defined networks and openflow. Internet Protocol J 16(1). http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_16-1/161_sdn.html

② Foundation O N. OpenFlow switch specification. version 1.3.0. https://www.opennetworking.org/images/stories/downloads/specification/openflow-spec-v1.3.0.pdf

量的一部分,并不总是需要对所有的流进行限速.

目前网络中的限速技术仍旧是对所有流量不加区分地以统一速率限速,从而导致网络带宽不能够得到充分利用,造成资源的浪费;同时,由于当前网络中流的数目非常庞大,如果对每条流独立地进行限速,则需要为每一条进入网络的流分配一个限速通道,这将会耗费大量的硬件资源空间,且随着流数的增多,硬件消耗线性增加,可扩展性较差.此外,多个限速通道输出队列的管理与调度,进一步增加了硬件设计的复杂度,这就导致了目前网络中很难实现每流限速的功能,因此迫切需要一种高效的限速方案来有效地解决这一问题.

随着网络技术的发展成熟,以及复杂网络应用对 QoS 的需求,OpenFlow 定义的每流限速(Meter)对传统的实现方法提出了挑战.本文旨在硬件资源有限的情况下,以不影响其他流为前提,仅对突发传输的流进行独立的速率控制以实现每流限速,为网络提供更好的 QoS 保证.

在 DQS(Dynamic Queueing Sharing)中[10],作者提到,网络中某个时间段内并发传输的流数目往往只有几百条,而且更进一步,在这几百条流中大多数的流是无需限速操作而可以直接进行转发的,仅仅在网络发生拥塞时,需要控制某些突发流的速率,而保证大多数的流按照正常速率转发.所以,当网络中的一条流由"活动"(Active)状态变为"休眠"(Silent)状态时,占用的限速通道中也将暂停数据包的传输,转为空闲状态,此时则可以通过释放该通道,供其他新进入网络的流使用,而不用为流重新开辟限速通道,从而使得多条流分时共享有限的硬件限速通道资源.

如图 2 所示,并存的突发流,即需要同时限速的流仅占网络总流量的少部分,在不同的时段 $t1$ 和 $t2$ 需要对不同的流限速,通过动态调度和分时共享,使得有限的限速通道为远超过限速通道数目的流进行限速.

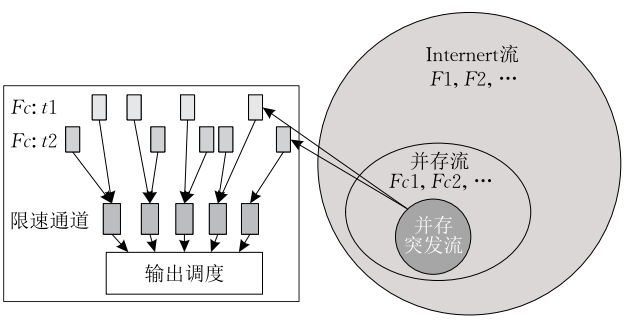


图 2 分时共享限速通道

基于以上理论,结合 OpenFlow 对每流区分处理的特性,本文提出了动态共享限速(Dynamic Meter Sharing,DMS)算法,实现多流分时共享限速通道,Meter 在多条流之间动态调度,使得每条流在独立的通道中限速传输,等效于为该流分配了专用通道.在功能上,DMS Meter 可解决网络中的每流限速需求与硬件资源限制的矛盾.

考虑到在某段时间内,网络中同时需要限速的流的数量会超过限速通道的数量,由于硬件资源的限制,一段时间内只能保证有 N 条流独立地限速(其中 N 是限速通道的数量),不能确保每条流对应一个限速通道,有些流被分到共享限速通道是不可避免的.对于某条流 f,在一段时间内,如果限速通道数不够,将会被分到共享通道限速,当有限速通道被其他流释放时,Meter 又可为其分配独立的限速通道,相对于传统的限速方式,Meter 能够处理更多的流,因此对 f 的限速效果更优.

本文在课题组已实现的 OpenFlow1.3 交换机的基础上,提出了一种高效快速地为流分配限速通道的算法 DMS,基于 ONetSwitch 硬件开发平台,设计并实现了 OpenFlow 1.3 标准的每流限速功能模块,并用实际流量对其性能进行了测试评估,结果表明本文提出的 Meter 通过动态共享的方法实现了对每流进行限速的功能,对网络中大量数据流以不同的速率限速,并且在多条流之间共享限速通道.

本文第 2 节介绍本文DMS算法设计的主要思想;第 3 节详细描述 Meter 的硬件设计方案;第 4 节对 Meter 的算法及相关参数进行理论上的分析;第 5 节介绍硬件实验平台 ONetSwitch 相关内容;第 6 节通过实验测试,评估Meter的每流限速性能;第 7 节对比每流限速的相关工作;第 8 节对本文的工作作出总结.

2 DMS 基本思想和系统结构

2.1 DMS 整体设计思想

Meter 的设计采用软硬件相结合的思想,构建出如图 3 所示的系统,在软件层维护一张软件限速表以及一个用来实时监测网络中流的速率并与软件限速表中预先设置的限速速率进行比较的速率选择模块;硬件层维护一张硬件流表和每流限速表,若干限速通道以及从限速通道中轮询读取数据包的输出仲裁模块.软件层通过写寄存器信号对硬件表进行配置,通过读寄存器信号读取硬件寄存器的信息,统计分析限速流的数量、时延等.

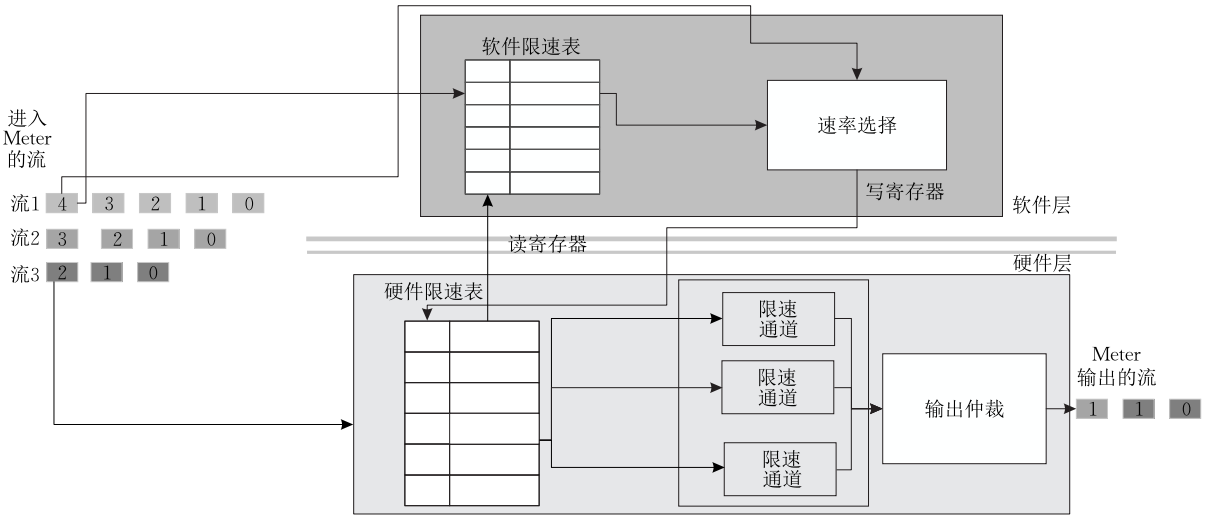


图 3　Meter 软硬件整体架构

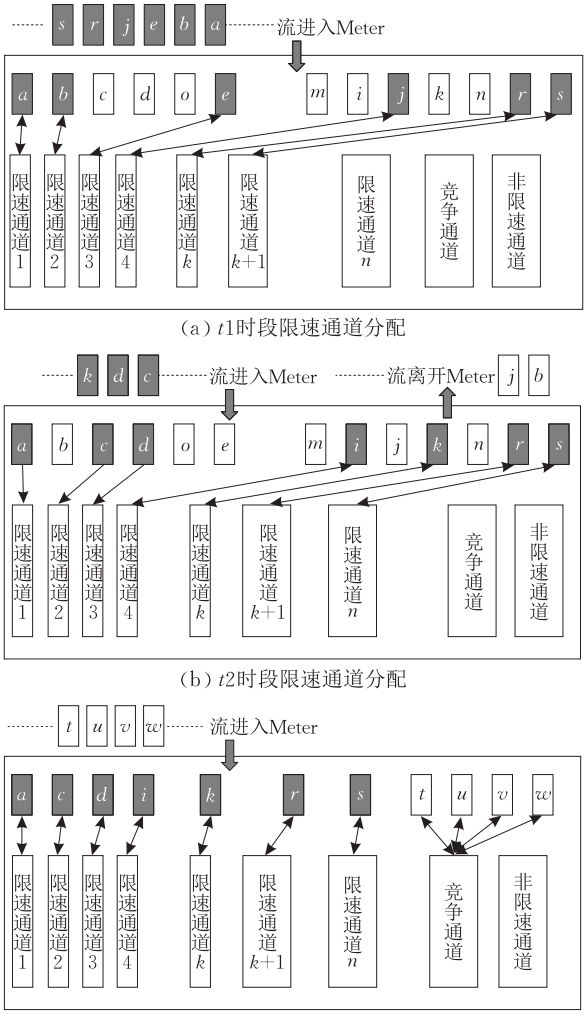
软件层实时监测网络流量，对每一条进入网络的流，在软件层面计算其速率并查找软件流表，匹配该流对应的表项，该表项中设置若干限速参数，在其中通过比较得到小于流当前速率的最大速率值 Ri，生成一条硬件流表项{$flow\_id$，$Meter\_id$}以及一条限速表项{$Meter\_id$，Ri}，表示匹配了流标号 $flow\_id$ 的流以限速表中第 $Meter\_id$ 条表项配置的参数 Ri 进行限速，这两条表项通过写寄存器下发到硬件表上.

在硬件中匹配了 $flow\_id$ 中的流，根据其指令域中的 $Meter\_id$ 查找硬件每流限速表，得到其对应的限速参数，接着该流被分配到某个限速通道，以查找到的速率 Ri 进行传输；如果软件层没有为流配置限速信息，则该流无需限速，直接转发.

本文每流限速设计的核心思想即前言所述的动态限速共享(DMS)机制，主要体现在硬件层面的设计上，在一条新流进入网络时为其分配空闲的限速通道，而当一条流在网络中传输结束之后，释放其占用的限速通道，实现分时共享.

硬件Meter通过数据包的所属的流标号$flow_id$识别出某条流，在其第一个包进入网络时，Meter 为其分配一个新的限速通道，后继到达的属于该流的数据包都被分配到同一通道；当一个限速通道在一段时间 Δt 之内没有数据传输，则说明占用该限速通道的流已经不再有效，即变为“休眠”状态，检测到该通道超时，将其解除占用并加入空闲通道队列，等待分配给新加入的流；而如果该流在一段时间之后重新恢复 Active 状态，则为其重新分配另外一个空闲的限速通道；当所有竞争通道都被占用时，后进入网络的流则被分配到竞争通道中以统一的速率限速，直到有限速通道被释放；无需限速的流被分到非限速通道执行正常转发处理. Meter 使多条流分时共享有限的限速队列，同时保证每流独立限速，不受其余共存流的影响.

图 4 所示为用 Meter 实现每流限速的图形化解



(a) $t1$时段限速通道分配

(b) $t2$时段限速通道分配

(c) 限速通道占满时分配情况

图 4　流与限速通道映射关系图

释,举例说明 Meter 如何使多条流动态共享限速通道.

在图 4(a)中,$t1$ 时段,流$\{a,b,e,j,r,s\}$是网络中处于"活动"状态的流,双向箭头线表示该时段限速通道的分配情况:$a\rightarrow1$,$b\rightarrow2$,$e\rightarrow3$,$j\rightarrow4$,$r\rightarrow k$,$s\rightarrow k+1$;在图 2(b)中,$t2$ 时段,流$\{b,e,j\}$由"活动"状态变为"休眠"状态,而新增加了"活动"状态的流$\{c,d,i,k\}$,这时网络中$\{a,c,d,i,k,r,s\}$是"活动"状态的流,虚线表示该时段限速通道分配情况:$a\rightarrow1$,$r\rightarrow k$,$s\rightarrow k+1$ 保持不变,将流$\{b,e,j\}$占用的通道释放,分配给流$\{c,d,j\}$;在图 2(c)中,当所有的限速通道都被占用,此时新进入网络的流$\{t,u,v,w\}$都被分到竞争通道中限速.在此例中,通过上述动态限速共享的方法,10 条流分时共享 7 个限速通道,从"外部"看实现了对每流的独立限速.

2.2 硬件设计架构

如图 5 所示,Meter 包含 3 级功能模块:(1)查表匹配模块.对进入 Meter 的流,查找限速表匹配之后,为其配置独立的限速通道并转发;(2)限速通道.对进入该通道的流进行数据包级的限速处理,之后将数据包存放入输出缓存队列中;(3)输出仲裁.从限速通道的输出队列仲裁取出数据,由轮询算法选择一路到输出队列,从相应的输出端口转发.

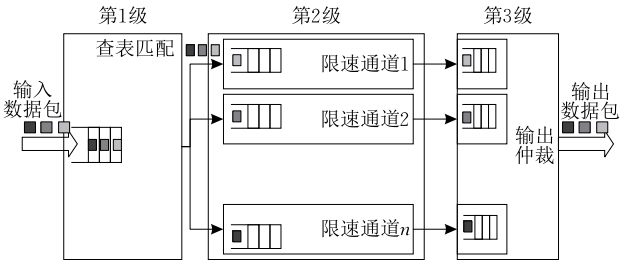


图 5 Meter 硬件整体架构

2.2.1 查表匹配模块

查表匹配模块提取数据包匹配的限速表项号和流标号信息,分别查找硬件限速表和活动流限速记录表,判定该数据包属于哪一条流,将其分发到该流相应的限速通道,之后在限速表和活动流记录表中注册;而当某个限速通道在一段时间 Δt 之内都没有数据发送,那么说明其对应的流已经传输结束,则该模块接收流的注销请求,释放其占用的限速通道,同时删除其注册信息.当数据包连续到达时,并行完成数据包的处理和传输,在传输一个数据包的同时,进行对下一个包的查找匹配和限速通道分配工作.

2.2.2 限速通道模块

限速通道的主要功能是使用漏桶算法控制数据注入到网络的速率,平滑网络上的突发流量,把输入的突发数据流以一个平缓的速率输出,使得包与包之间以固定的间隔传输.

该流水级设置 3 种限速通道,当共存流的数目不超过限速通道的数目时,每个标准的限速通道对应一条独立的流;而当共存流的数目大于限速通道数目时,独立的限速通道已经全部被占用后,后续的流都被分配到该竞争的共享限速通道;对于不需要限速的流,将其分配到"虚拟"的限速通道中,保持其输入速率,收到则立即转发.

2.2.3 输出仲裁

查表匹配模块将输入数据分发到 n 个通道并行限速,限速完成后暂存在相应的 n 个输出队列中,而输出只能处理一路数据,因此需要在这些队列中执行调度仲裁.由于并非所有的队列都同时有数据要输出,在传统的轮询调度机制中,大部分的时间都在询问输出队列是否非空,当一个队列传输完成之后,另一个队列已经有了数据,而循环队列指针还未询问到该队列,从而导致该队列空等,增加了传输时延,效率低下.

本文输出仲裁模块的设计中,增加非空队列链表机制,同样用轮询算法询问队列,当有队列非空时,立即将其加入到输出队列链表中,等待排在它之前的队列发送完数据之后,就可以立即从该队列中取出数据,提高了输出效率.

在上述每流限速 Meter 的设计的 3 级模块中,查表匹配为每流分配限速通道是其中最复杂的逻辑,主要涉及到两个重点解决的难题:(1)如何快速判定新进入网络的数据包属于哪一条流?(2)对每个进入网络的数据包,在转发之前都需要解析包头并查表为其分配限速通道,这将造成很大的延迟,如何最大程度上降低延迟,提高处理效率?

针对上述两个问题,本文通过两项重要技术来解决,即硬件限速表和活动流记录表的二级查找技术以及流水处理数据包技术,这两部分内容将在第 3 节中分别作详细说明.

3 Meter 详细设计方案

3.1 二级查找技术

Meter 为每流配置一个限速通道,维护一张由限速通道号$(0,1,\cdots,n-1)$到流标号的直接映射表,即活动流记录表,对每个进入 Meter 处理的数据包,根据其携带的所属流标号,判断是否已为其分配

了限速通道，如果是，则直接将其转发到流标号对应的限速通道传输，否则说明其为一条新流的第 1 个数据包，为之分配空闲的限速通道，将新的映射关系加入活动流记录表中，后续到达的属于该流的数据包则能匹配该项，转发到相应限速通道.

由于流标号是不连续的，如果顺序查找活动流记录表，则平均每包需要耗费 $n/2$ 个时钟周期，最坏情况要将整张表搜索一遍，这种情况下网络中的延时将会非常大.

Meter 采用二级查找的方式，如图 6 所示，左侧限速表中 Meter 号表示限速表的表项编号，活动流链头列表示匹配了该限速表项的活动流链表头在活动流记录表中的编号；右侧活动流记录表中，限速通道号列表示限速通道的编号，流标号列表示分配到该通道限速的活动流标号，后继空闲通道列是链表指针，指向匹配同一限速表项的下一条活动流. 因为 Meter 可以使多条流共享一条限速表项，限速表为每一个表项维护一个链表，链式存储匹配该限速表项的所有活动流对应的限速通道号，在限速表项记录链头；在活动流记录表中将对应同一限速表项的所有限速通道链接，那么每包进入 Meter 之后，先直接映射查找限速表，再由链头开始顺序查找对应的链表，直至找到匹配的流标号或至链尾，结束返回查找结果，相比于顺序查找方式大大降低了时间复杂度.

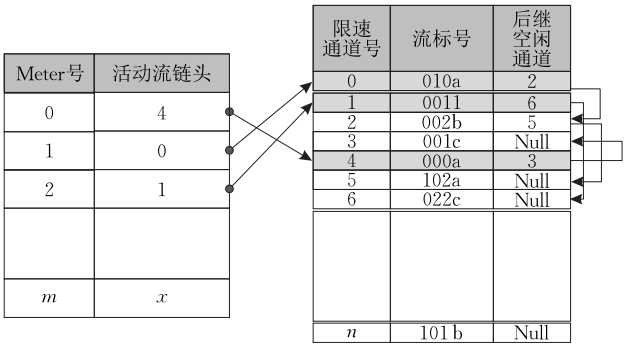


图 6　Meter 二级查找技术

3.2　流水处理

在硬件资源足够的情况下，流水处理技术能够显著地提高系统的效率. 将包的处理过程分成两段流水线，在传输一个数据包的同时，进行下一个包的限速通道号的分配处理工作，降低数据包在该模块处理时的总时延以提高效率.

如图 7 所示，状态机 1 和状态机 2 分别为流水阶段 1 和 2，即限速通道的分配与数据包的传输，阴影箭头表示一个包的处理流程. 在状态机 2 启动之前，先检测当前需要输出的数据包是否已经分配好了正确的限速通道，如果是，则直接将数据包传输给下一级模块，否则等待上述过程（即状态机 1）完成之后才可启动数据的传输；在一个数据包开始传输时，如果输入队列非空，则说明有新的包要处理，启动状态机 1，为下一个包的按时传输做好准备；每个流水段完成一次数据包的处理之后，都需要检测启动条件是否成立，如果是，则进行下一次处理，否则等待触发信号.

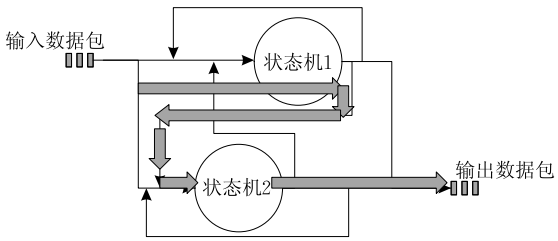


图 7　流水处理流程

3.3　输出队列调度

Meter 中设置多个限速通道，如 62 个限速通道，对应 64 个输出队列，包括一个共享限速队列和一个非限速的队列，输出管理模块一次只能从一个队列读取数据，继而转发到相应端口. 在之前的工作中，每个输出端口对应一个输出队列，由于硬件平台上仅有 8 个输出端口，使用 Round-Robin 算法即可以有效地处理各队列中的数据包，使其按序从对应端口转发出去.

然而，在 Meter 中有 64 个输出队列，而且由于进行了速率控制而导致多个队列驻留数据包. 按照轮询各队列控制数据包输出的算法，造成部分输出队列“空等”，从而加大了数据包的时延，尤其是对无需限速的数据包. 在本文中，通过设置“非空队列链”有效地解决这一问题. 对于有数据包的队列，维护一个链表，当有队列非空时，将其加入链尾，当从一个队列中取出数据包后，将其从链头删除，继续从其后续非空队列中读取数据包.

对于有 64 个限速通道的 Meter，输出队列的有效调度可提高 Meter 的效率，为流提供更可靠的时延保证.

4　Meter 性能分析

4.1　二级查找表的时间复杂度

限速表的查找是直接映射查找，时间为常量 1，而线性链表的查找与其长度 L 有关，其中 L 是由软件配置的“最多有 L 条流共享一条限速表项”所决

定的,而与活动流记录表的长度无关.

为减少 L 的值,优化查找时间,将部分限速表项冗余,即多个限速表项对应同一限速速率,可成倍减小链表的长度. 如有 N 条流需要以同一速率 R 限速,则将这 N 条流分为两组,在硬件限速表中对应两条限速速率都为 R 的表项,那么活动流链的长度即可由 N 降低至 $N/2$,缩短了一半的查找时间.

4.2 冲突流与限速通道数量之间的折衷

网络中共存流的数目时刻在变化,限速通道的数量并不总是满足需求. 当流的数目大于限速通道数时,如有一条新流进入网络中,则所有的限速通道都被占用,在这种情况下,放松"每流独立限速"的需求,单独开辟一条共享的限速通道,当无空闲的限速通道时,该通道处理在这之后所有的新流,称之为冲突流,直到有通道被释放. 冲突流的数量 C 与限速通道的数量 R 存在着负关系,当 R 值增加时,C 值减少;而当 R 值减少时,C 值增加. 需要根据对实际流量的测试,找到这二者之间的平衡点,以实现最优的性能,既能充分保证每流限速,又能最大程度上减少硬件资源的消耗.

5 实验环境

本文的实验环境,是基于课题组和叠锶 MeshSr 联合开发的 ONetSwitch45 网络创新平台[11],如图 8 所示,ONetSwitch45 是一个基于 FPGA 的网络创新实验平台,其核心器件是 Zynq 7045 SoC[12],其内部包含一个 ARM Cortex-A9 双核 CPU 以及一个 Kintex FPGA,ARM 处理器运行在 667 MHz,具有较强处理能力,能够运行完整的 Linux 操作系统;同时 FPGA 提供约 355K Logic Cell,是传统的NetFPGA 所提供逻辑资源的 6 倍以上,能够支撑更加复杂的硬件逻辑,内部能够提供超过 40 GB 网络数据的处理能力,方便研究者在其内部编写逻辑进行硬件加速,两者之间使用高速总线 AMBA 互联. ONetSwitch45 板卡资源还包括 72 MB QDR2 SRAM,提供约 48 Gbps 的吞吐率用于线速网络数据缓存或其他高速应用.

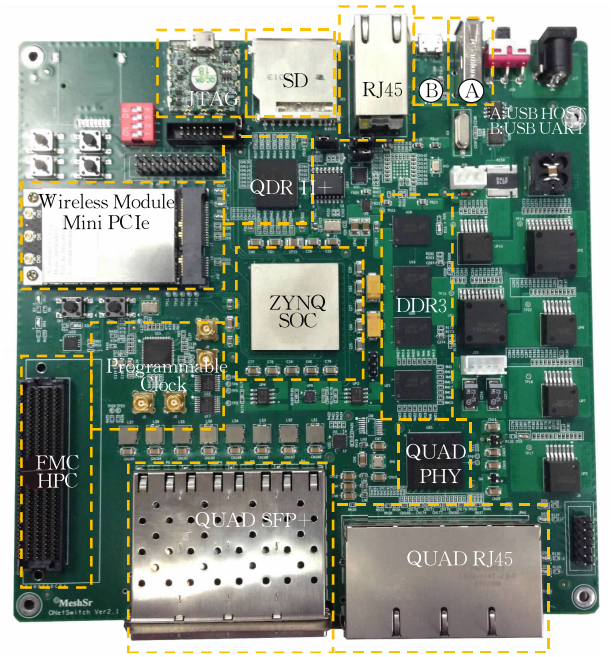


图 8 ONetSwitch 开发平台实物

与传统插卡式的网络实验平台相比,ONetSwitch 开发平台无需额外匹配服务器,单板可以独立工作. 同时 ONetSwitch 采用开放的体系结构设计,允许使用者针对需要进行硬件或者软件编程,同时尽可能地复用已有的设计. 硬件提供丰富的扩展接口,FMC 高速板间互连接口,可以接驳功能扩展卡,或者进行板间互连;Mini-PCIe 扩展接口可以支持对应标准模块,以及控制通信接口针对多种常见开发板进行连接. 研究人员可根据需要扩展对应模块,来支持数据中心网络[13]、移动互联网、命名数据网络等研究以及软件定义网络研究与创新应用,基于 ONetSwitch 搭建理想的网络创新平台,进行网络创新开发. 同时,ONetSwitch 包含 4 个最高支持 10G-BaseR 等万兆以太网协议以及多种光协议的 SFP+ 接口与 4 个三速自适应以太网接口,用于适应不同的实验环境,如图 9 所示.

在此之前,由斯坦福大学研发的 NetFPGA[14] 作为一套灵活易用的网络实验平台在全球有着广泛的部署,但是随着网络带宽提高与应用复杂化,尤其是 SDN 成为研究的主流,NetFPGA 已经无法满足对新协议开发的资源支持.

ONetSwitch45 与 NetFPGA 相比,提供了超过 5 倍的片上资源,以及更高容量与带宽的片外存储. 在提供千兆以太网接口的同时,额外增加 4 个支持万兆以太网协议的 SFP+ 接口,支持更高速的网络实验. 在主机接口上,ONetSwitch 所提供的带宽也完全满足万兆流量实时处理的需求,支持主机应用创新,而 NetFPGA 受限于 PCI 接口的物理性能无法提供超过千兆的主机应用支持.

与 NetFPGA 10G 版本相比,ONetSwitch 能够友善地兼容原有社区的应用,无需做过多修改,同时片上资源更加丰富,易于对新应用进行支持. 在功耗估计方面,ONetSwitch 也可以提供整体和局部电源功耗测量以及绿色网络的支持,这也是 NetFPGA 系列所没有涉及到的.

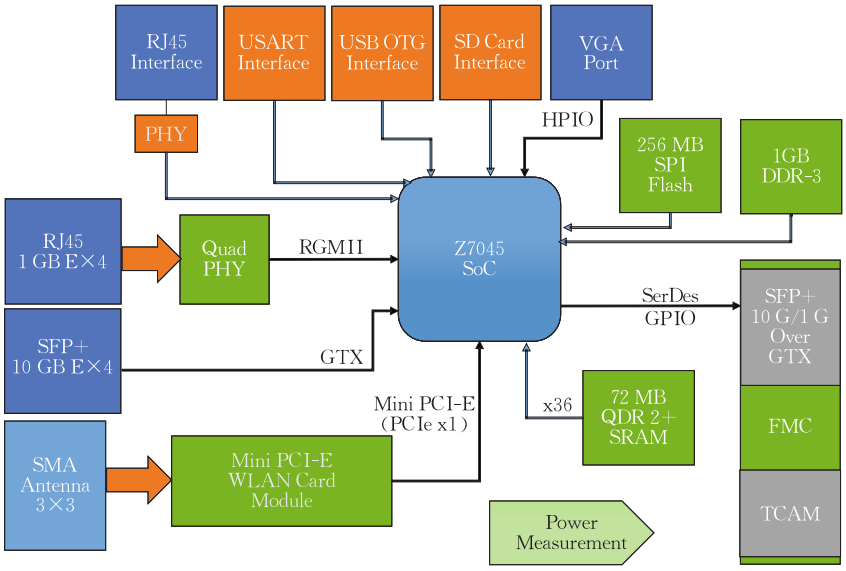


图 9　ONetSwitch45 开发平台结构

6　测试评估

实验测试分为 3 个部分，分别是：(1) 用 Xcap 工具构造 3 种模拟流量，分别包含 64、128 和 256 条流，即流量 SF1～SF3(Simulation Flow)，以恒定的速率发送，限速分为 4 个等级：0.9 Mbps、0.8 Mbps、0.7 Mbps、0.6 Mbps，用 Wireshark 抓包工具统计分析各条流限速后的速率分布；(2) 将网络中的实际流量，分别以不同速率发送，用 Wireshark 统计分析各流的速率，对突发流计算得到其限速参数，配置硬件限速表，对比限速前后的流量速率，并且比较 Meter 对不同流的限速效果；(3) 用 Smartbits 以不同的速率发送数据包，测试在高速网络环境下 Meter 的限速性能.

6.1　模拟流量测试

用 tcpreplay 发包工具以恒定速率 1.87 Mbps 分别发送构造的流量 SF1～SF3，硬件限速表中配置 16 条表项，则这 3 种流量分别有 4、8 和 16 条流共享同一条限速表项，限速参数的配置与流标号 *flow_id* 的关系分为 4 个等级：

flow_id % 4==0：限速到 0.6 Mbps；

flow_id % 4==1：限速到 0.7 Mbps；

flow_id % 4==2：限速到 0.8 Mbps；

flow_id % 4==3：限速到 0.9 Mbps.

6.1.1　各流限速后的速率分布

在限速通道数目为 62 的情况下，分别发送流量 SF1～SF3，得到正确的限速结果，如图 10 所示，横坐标表示模拟流量中各流的标号，纵坐标表示经过 Meter 处理后各流的速率，竞争限速通道的限速速率设置较大，图中速率高的流即表示被分配到竞争限速通道的流，即冲突流.

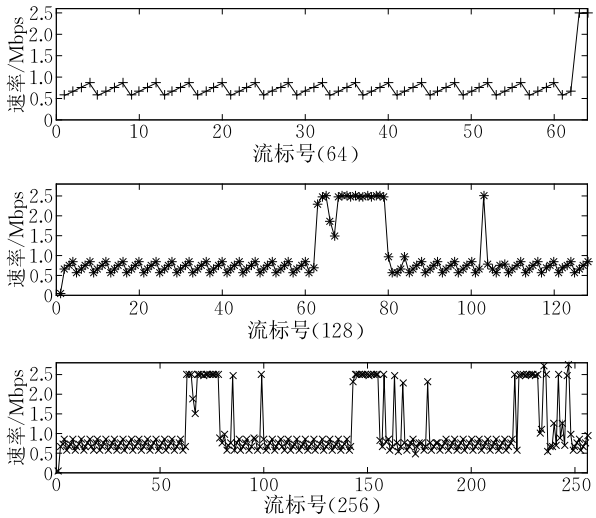


图 10　Meter 限速分析

运行结果显示，对于有 62 个限速通道的 Meter，当分别发送流量 SF1～SF3 时，分别有 2、20 和 61 条流被分到竞争通道. 同时，对于流量 2 和流量 3，当第 63 条流进入 Meter 时，所有限速通道都已被占用，此时后续到达的流都被分配到竞争通道，但是经过一段时间后，最先到达的流变为“休眠”状态，释放其占用的限速通道，后续的流则可被分到空闲的通道执行限速，从而实现了多流动态共享限速通道.

6.1.2　限速等级与冲突流数量关系

在限速通道数目为 62 的情况下，分别以 1.44 Mbps～1.94 Mbps 的速率发送上述 3 种流量，对各流的限速配置保持不变，统计分析在不同限速

等级下被分配到竞争通道的流数量，得到如图 11 所示结果，横坐标表示模拟流量的发送速率，纵坐标表示冲突流数目，从上到下 3 条折线分别对应流量 SF1～SF3.

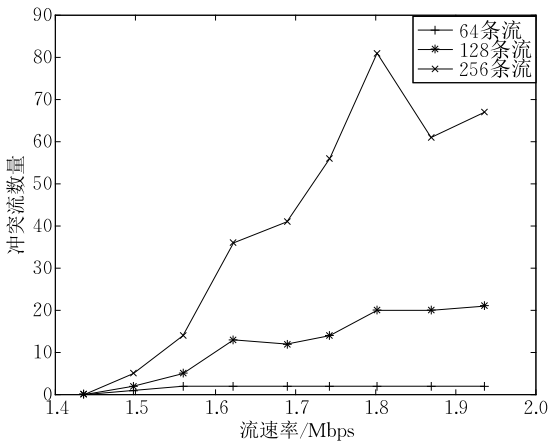


图 11 发送速率与冲突流数量曲线关系

令 $r(\mathrm{ini})$为流的原始速率，$r(\mathrm{lim})$为流限速后的速率，则限速等级 L 定义为 $r(\mathrm{ini})/r(\mathrm{lim})$，例如，流原始速率为 1.44 Mbps，配置限速速率为 0.9 Mbps，则 $L=1.6$，对一条流的限速等级越大，则该流占用限速通道的时间就相对较长. 如图 11 所示，随着发送速率的增加，Meter 对流的限速等级由平均 1.9 增至 2.6，被分到竞争通道的流数量随着限速等级的加大而呈线性增加趋势.

6.1.3 限速通道数与冲突流数的关系

分别用 14、30 和 62 个限速通道对上述构造的流量 SF1～SF3 进行限速测试，统计分析在不同限速通道数下被分到竞争通道的流数量，得到如表 1 所示共 9 组实验结果，如对于 64 条流，14 个限速通道的一组数据，“14/50”中 14 表示正确限速的流(即限速流)的数目，而 50 表示被分到竞争通道的流(即冲突流)数目.

表 1 模拟流量冲突流数与限速通道数的关系

流类	$r1$ 类的冲突流数		
	14	30	62
64 flow	14/50	30/34	62/2
128 flow	25/103	53/75	108/20
256 flow	47/209	96/160	195/61

由表 1 可见，随着限速通道数量的增长，被正确限速的流数目呈相同的增长趋势，而冲突流数量呈递减趋势；并且在限速通道数不变的情况下，网络中需要限速的流数量越多，则被分到竞争通道的流数量也就越多.

6.2 实际流量测试

用网络中真实的流量对 Meter 进行测试，ONetCard 有 4 个千兆以太网接口，分别发送同一流量到 1、2、3、4 个网口，从而可以构造出 4 种流量 $RF1$～$RF4$(Real Flow)，分别包含 32、64、96 和 128 条需要限速的流. 用 Wireshark 工具统计在限速通道数变化的情况下，多种流量在 Meter 中的处理情况，并最终分析得到影响 Meter 对流限速性能的因素.

6.2.1 Meter 限速性能分析

首先，在 62 个限速通道的情况下，发送流量 $RF1$，得到限速前后的速率比较结果，如图 12 所示，取其中 3 条进行分析，横坐标表示实际流和限速流中依次到达的数据包，纵坐标表示用 Wireshark 统计的每个数据包到达时间，∘和∗标记分别表示限速前后的流.

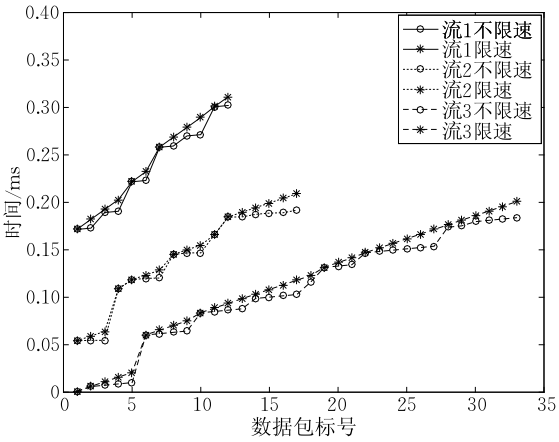
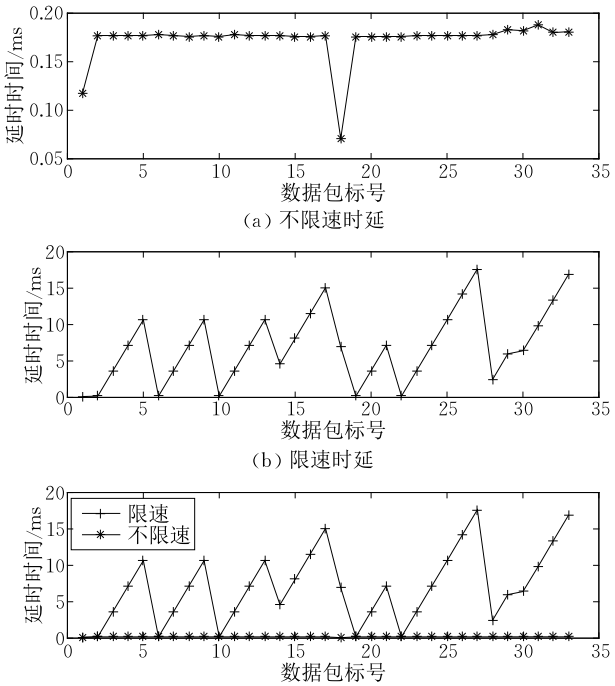


图 12 Meter 对不同流的限速结果

经过 Meter 限速处理后的各流相较于初始值，两个数据包之间的间隔增大，表明流速得到控制，即抑制了突发流的传输，且限速后曲线渐趋平缓，数据流的速率在一段时间内基本保持不变，起到了平滑网络流量的作用.

在图 12 中，只分析了 Meter 对少量数据流量的限速结果，对比了未限速和限速时数据包到达的时间；然而，在高速网络环境下，要处理大量的数据流，分别发送 1 s 和 45 s 的实际流量，统计一条流中最小的两个包之间时间间隔是 0.000 13 s，然后设置限速参数使得时间间隔为 0.001 s，相当于最高以 1/10 的等级限速，在发送 1 s 的实际流量时，该流包含 244 个数据包，限速后收到 55 个；在发送 45 s 的实际流量时，该流包含 9456 个数据包，收到 2470 个，限速与不限速的时延对比与图 13 类似，部分数据包被丢弃，两数据包之间的最大时间间隔不超过预先设置的 0.001 s.

接着，进行 Meter 的时延分析，对实际流量中的一条包含 33 个数据包的流 $f1$，分别为其配置 Meter 限速参数和不配置 Meter 参数直接转发，在这两种情况下，统计每个数据包接收时间与发送时间的间隔，得到如图 13 结果.



(a) 不限速时延

(b) 限速时延

(c) 不限速时延与限速时延的对比

图 13　Meter 对流 $f1$ 处理的时延结果

在图 13 中，横坐标表示流 $f1$ 中包的序号，纵坐标表示每包接收时间与发送时间的差值，单位是 ms. 图 13(a) 表示不经过 Meter 限速处理而直接转发的包时延；(b) 图表示经过限速处理后的包时延；(c) 图表示两者的对比，由此图可以得到，在流突发传输的一段时间内，Meter 对流的速率起到了有效的控制作用.

图 13 是 Meter 对限速流的处理时延，在图 14 中，分析 Meter 对于无需限速的流处理的时延，在没有 Meter 时，数据包在网络中的平均处理时延约为 0.106ms，而引入 Meter 后，对于不需要限速的流而言，其平均处理时延增加至 0.173ms，仍能够保证高速网络环境下的对数据包的快速转发.

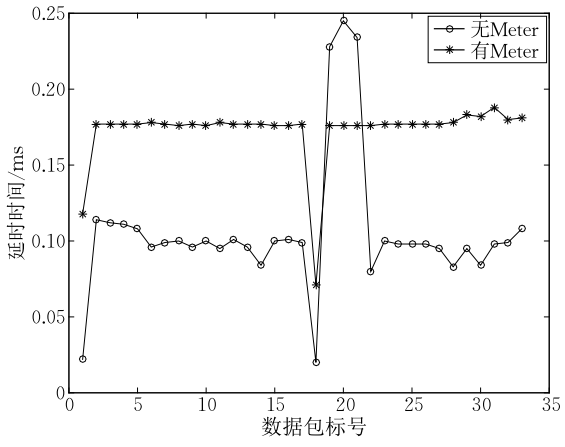


图 14　Meter 对无需限速的流的时延

6.2.2　限速通道数与冲突流数的关系

对特定数目的流，改变限速通道的数量，当限速通道数分别为 14、30、62 时，统计被分到竞争通道的流数，如对于 32 条流，14 个限速通道的一组数据，“17/19”中 17 表示正确限速的流(即限速流)的数目，而 19 表示被分到竞争通道的流(即冲突流)数目. 由表 2 可见，随着限速通道数的增加，被分到竞争通道的流(即冲突流)的数目递减. 表 2 中限速流与冲突流的和大于总的流数，是因为在实际流量中，由于流中的数据包分布不均匀，在有些流中后到达的数据包进入 Meter 后，该流已经注销，且所有的限速通道已被其他流占用，则该数据包被分到竞争通道中.

表 2　实际流量冲突流数与限速通道数的关系

流类	$r1$ 类的冲突流数		
	14	30	62
32 flow	17/19	30/9	0
64 flow	27/51	50/31	64/0
96 flow	40/79	78/60	96/0
128 flow	58/103	114/46	128/0

6.2.3　超时时间与冲突流数的关系

对于 6.2.2 节中的 62 个限速通道，当限速通道的超时时间为 0.1s 时，所有的流都被正确限速，冲突流的数目为 0. 在此基础上，研究超时时间与冲突流的数量关系，以相同的速率发送流量，将超时时间分别设置为 0.4s 和 0.6s 时，得到如表 3 所示结果.

表 3　实际流量冲突流数与超时时间的关系

流类	超时时间/s		
	0.1	0.4	0.6
64 flow	64/0	62/2	62/2
96 flow	96/0	91/28	70/34
128 flow	128/0	126/6	124/20

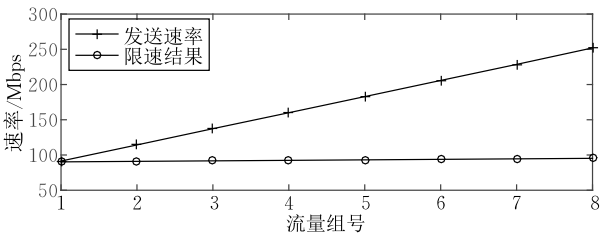
分析表 3 中数据可以得到结论，随着超时时间的延长，被分到竞争通道的流数目增加. 超时时间过短，则流频繁在各限速通道中切换，而超时时间过长，则会造成限速通道资源的浪费，因此，在实际流限速中，需根据这两者之间的平衡关系折衷，取相对最优的超时时间.

如前所述，超时时间与限速通道的个数是影响 Meter 限速性能的两个重要参数，同时决定系统限

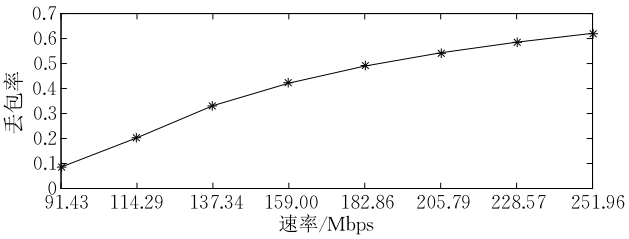
速精度以及硬件利用效率.如何设置这两个参数,进而需要在既节省硬件资源同时又不影响 Meter 限速性能的前提下,得到最优的结果,以使得 Meter 以较少的限速通道数为大量的流量限速,需要进一步用更多的实际流量来测试分析,这将在以后的工作中逐步完成.

6.3 Smartbits 流量测试

用 Smartbits 工具分别以不同的速率发送数据包,受限于 Smartbits 构造数据流量的简单性,而本实验的目的是为了测试 Meter 在高速网络下的性能,故而仅构造 32 条流,对其中 31 条进行限速,第 32 条直接转发处理,得到如图 15 所示结果.



(a) 不同速率下限速结果



(b) 不同速率下的丢包率

图 15 高速网络下限速结果

图 15(a)为以 91.43Mbps～251.96Mbps 的 8 组速率发送数据包,纵轴表示 8 组限速前与限速后的速率值,随着发送速率的增加,限速后速率保持在 92.4Mbps 附近,在高速网络下,Meter 能以不同等级对流量进行限速,控制网络中流的速率;图 15(b)为 8 组发送速率下的丢包率,发送相同数量级的流量,丢包率随着发送速率的增加而增大,之后趋于平缓.当网络中流的速率很大,而设置的限速速率与原速率相差较大时,必然会引起较大程度的丢包.

7 相关工作

在 Raghavan 等人[15]的分布式限速工作中,用多个分布式的限速模块协同工作,对分散到各个域的云服务流量做单一聚合的全局限制.其采用 GRD 和 FPS 算法对流量的速率进行控制,各个分布的限速模块之间通过协议通信并更新自身信息,最终统一集中地限速,这是传统的限速方案,并没有对流进行精确的划分.

Covington 等人[16]在其实现的 packet generator 中实现了对流的区分限速.其按照数据包在硬件平台上的输出端口号对流进行划分,硬件平台上有 4 个以太网接口,分别对应 4 个限速通道,从同一个端口输出的数据包按照同样的速率限速.packet generator 没有对流进行解析和分类,只是按照其转发端口进行区分限速.

Melman 等人[17]实现了网络路由交换设备到主机 CPU 的流量限速.将网络中流量划分为数据流量和控制流量,捕获控制流量到 CPU,控制数据包分类有 BPDU,LACP,GVRP,RIPvl,RIPv2,OSPFVZ,PIM,TELNET,HTTP 等等,用一个包处理器对数据包分类,每一类对应一个 CPU 通道,通过阈值控制每个通道数据包的数量以达到限速的目的.

Cheriton[18]的近似每流限速方案中,对流进行了分类,每条流对应一个限速的队列,各个限速队列相互独立,系统中设置一个控制器,为每个队列分配限速的令牌,实时地控制各流的限速情况.

Covington 等人的工作中,对流量进行了简单的划分,按照输出端口号限速,但是并没有对流进行更细粒度的划分;Melman 等人的工作和 Cheriton 的近似每流限速对流量的划分更为精确,但是每条划分的流对应一个限速队列,而 DMS 算法可以使得多条流共享限速通道,使用较少数量的限速通道为网络中多种类流量限速;在限速方案上,packet generator 与 Meter 都采取漏桶算法,相对于单一的数据包计数效果更优.除此之外,在 Meter 每流限速中,可根据上层软件需求定制需要限速的流,如根据 IP 协议、IP 地址、传输层端口号等,可满足不同应用的需求;且本文提出的 DMS 算法为流量动态分配限速通道,效率高,可有效解决硬件资源不足的问题.

FlowQoS[19]通过使用两个交换机分别进行分类和限速,来达到家庭网络的每流限速,但是这种结构无法支持高带宽低延迟的网络.

受限于硬件资源与芯片规模,现在绝大多数商用的硬件 SDN 交换机都不支持每流限速的功能,目前应用广泛的 Open vSwitch[20]是一种软件的 SDN 交换机实现,其中包含一些 QoS 能力,但是实际无法支持较大的流数量以及高的网络负载.因此绝大多数基于 Open vSwitch 的商用交换机也无法有效完成每流限速.

8　总结及未来工作

SDN 新型网络模型的提出以及新业务模式的应用对每流限速提出了强烈的需求. 在 OpenFlow 协议中，为每一条需要限速的流配置一条流表项，定义该流可以转发的最高速率，将其分配到指定的限速通道执行限速. 本文提出了 DMS 算法，在硬件限速通道一定的情况下，通过有效分配限速通道，以及调度流进入共享限速通道，使得硬件能够支持服务更多的流，从而实现每流限速的实际功能. 相比于传统的限速机制，DMS 算法可以用非常少的硬件资源实现同样的限速效果.

本文在 ONetSwitch 平台上的 OpenFlow 1.3 交换机中，利用 FPGA 资源实现了基于 DMS 算法的 Meter，并进行了一系列实验. 相较于传统网络中的限速机制，DMS 算法实现的 Meter 可用 14、30、62 个限速通道有效地为 128 条流限速，节约了硬件资源，同时保证高效的各流区分限速.

文中使用了 3 种流量进行算法的验证，分别是 Xcap 工具构造的模拟流量、Smartbits 测试仪生成的流量和网络中的真实流量. 测试结果表明，Meter 能够按照预先配置好的限速参数对需要限速的流进行正确的速率限制. 本文进一步分别研究了硬件限速通道数目和超时时间对冲突流数目的关系的影响，随着硬件通道数的增加，冲突流数目减少，且随着超时时间的延长，冲突流数目增加.

硬件限速通道的数目和超时时间是影响 Meter 限速性能的两项重要参数. 这将作为下一阶段的工作重点. 通过进行更大量的流量测试，进一步调整 Meter 以优化参数，使之实现更好的性能，为网络提供高效的限速.

致　谢　西安交通大学智能网络与网络安全教育部重点实验室 SDN 小组的同学之前在 OpenFlow 方面做了很多的工作，为本篇论文的完成奠定了良好的基础，给予了很大的帮助和指导，在此衷心表示感谢，同时感谢在百忙之中评阅论文的各位专家！

参　考　文　献

[1] Bakshi K. Considerations for Software Defined Networking (SDN): Approaches and use cases//Proceedings of the Aerospace Conference. Big Sky, USA, 2013: 1-9

[2] Ferguson A D, Guha A, Liang C, et al. Participatory networking: An API for application control of SDNs//Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM. New York, USA, 2013: 327-338

[3] Reich J, Monsanto C, Foster N, et al. Composing software defined networks//Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI). Berkeley, USA, 2013: 1-14

[4] McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: Enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74

[5] Open Networking Fundation. Software-defined networking: The new norm for networks. ONF White Paper, 2012

[6] Martins J, Ahmed M, Raiciu C, et al. Enabling fast, dynamic network processing with clickos//Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. New York, USA, 2013: 67-72

[7] Koldehofe B, Dürr F, Tariq M A, et al. The power of software-defined networking: Line-rate content-based routing using OpenFlow//Proceedings of the 7th Workshop on Middleware for Next Generation Internet Computing. New York, USA, 2012: Article No. 3

[8] Veltri L, Morabito G, Salsano S, et al. Supporting information-centric functionality in software defined networks//Proceedings of the IEEE International Conference on Communications (ICC). Ottawa, Canada, 2012: 6645-6650

[9] Wang R, Butnariu D, Rexford J. OpenFlow-based server load balancing gone wild//Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services. Berkeley, USA, 2011: 12-12

[10] Hu C, Tang Y, Chen K, et al. Dynamic queuing sharing mechanism for per-flow quality of service control. The Institution of Engineering and Technology Communications, 2010, 4(4): 472-483

[11] Hu C, Yang J, Zhao H, et al. Design of all programmable innovation platform for software defined networking. Power, 2014, 75(91W): 21W

[12] Zynq-7000 All Programmable SoC Overview. Xilinx datasheet DS190, Xilinx Inc, 2011

[13] Hu C, Yang J, Gong Z, et al. DesktopDC: Setting all programmable data center networking testbed on desk//Proceedings of the 2014 ACM Conference on SIGCOMM. New York, USA, 2014: 593-594

[14] Lockwood J W, McKeown N, Watson G, et al. NetFPGA—An open platform for gigabit-rate network switching and routing//Proceedings of the Microelectronic Systems Education. San Diego, USA, 2007: 160-161

[15] Raghavan B, Vishwanath K, Ramabhadran S, et al. Cloud control with distributed rate limiting//Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. New York, USA, 2007: 337-348

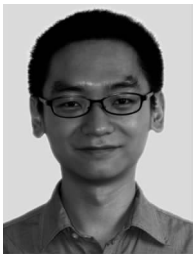
[16] Covington G A, Gibb G, Lockwood J W, et al. A packet generator on the NetFPGA platform//Proceedings of the Field Programmable Custom Computing Machines. Napa, USA, 2009: 235-238

[17] Melman D, Daniel T, Regev E. Rate limiting per-flow of traffic to CPU on network switching and routing devices. USA, 2012-8-28

[18] Cheriton D R. Approximated per-flow rate limiting. USA, 2004-4-20

[19] Bianchi G, Blefari-Melazzi N, Femminella M. Per-flow QoS support over a stateless differentiated services IP domain. Computer Networks, 2002, 40(1): 73-87

[20] Pfaff B, Pettit J, Amidon K, et al. Extending networking into the virtualization layer//Proceedings of the Workshop on Hot Topics in Networks (HotNets-VIII). New York, USA, 2009: 1-6



YANG Ji, born in 1987, Ph. D. candidate. His research interests include software defined networking and network security.

XU Chen, born in 1992, M. S. candidate. Her research interest is software defined networking.

GONG Zhi-Min, born in 1990, M. S. candidate. His research interest is software defined networking.

HU Cheng-Chen, born in 1981, Ph. D., associate professor, Ph. D. supervisor. His research interests include datacenter network and software defined networking.

GUAN Xiao-Hong, born in 1955, Ph. D., professor, Ph. D. supervisor. His research interests include complex network optimization and network security.

Background

Software Defined Networking (SDN) is becoming increasingly popular because of its potential in networking management. Networking virtualization, multi-tenants datacenter and many other new applications have been supported by SDN. As one of the most important mission, QoS achievement plays a critical role in SDN data plane functions. Currently working as a factor SDN protocol, OpenFlow provides meter for per flow rate limiter in its specification version 1.3 to deliver QoS. However, absolutely per-flow rate-limiting cannot be implemented in current work because flows number grows over time to the extent that far exceeding the rate limiters designed on hardware. And only approximate per-flow rate-limiting is implemented and verified.

In this paper, a novel solution DMS is proposed to solve the contradiction between hardware resources and enormous flows. There are only about 100 active concurrent flows in million network in-progress flows during a period, according to statistical result of network flows. Moreover only a small part of flows need to be rate limited. The main idea of DMS is to schedule among different flows to share the same rate limiter in different period. Referencing to the Meter description in OpenFlow-1.3 specification, this paper presents a detailed implementation of DMS Meter based on ONetSwitch network innovation platform.

Multiple groups of experiment verify the performance of Meter. The results show that comparing to traditional rate limiting solutions, DMS Meter effectively divides flows and provide applications with different rate-limiting services. While comparing to current per-flow rate limiting, DMS algorithm realizes enormous flows sharing limited rate limiters in different time period and improve the hardware resources occupation rate.

This paper is mainly supported by projects of the National Natural Science Foundation of China (61221063, 61272459, U1301254), the National High Technology Research and Development Program (863 Program) of China (2012AA011003), the Ministry of Education New Century Talents Support Program (NCET-13-0450), the Future Network Project of Jiangsu Province (BY2013095-1-12), and the Project of 54 Research Institute of China Electronics Technology Group Corporation (ITD-U14001/KX142600008).