DPDK 流量管理API使用指南

原创 DPDK开源社区 2018-03-30

作者 Jasvinder S. etc

DPDK 流量管理API使用指南

本文介绍了DPDK 17.08版本中新引入的API，用于流量管理（TM，traffic management）。 该API为服务质量（QoS, quality of service）流量管理配置提供了通用界面，该界面集合了由网卡（NIC），网络处理单元（NPU），专用集成电路（ASIC），现场可编程门阵列（FPGA），多核CPU等提供的一套标准特性， 其中包括：分层调度，流量整形，拥塞管理，数据包标记等其他功能。

此API是通用的，因为它不受底层硬件，软件或混合硬件/软件实现的影响。 它作为DPDK ethdev API的扩展，与流API类似。 包含一个典型的DPDK函数调用序列来演示实现。

主要特性

分层调度

TM API允许用户为具有特定实现支持的分层节点选择严格优先级（SP）和加权公平队列（WFQ）。无论在树中的节点级别/位置如何，SP和WFQ算法都可以在调度分层结构的每个节点使用。 SP算法用于在不同优先级的同级节点之间进行调度，而WFQ用于在具有相同优先级的同级节点组之间进行调度。

示例：如图1所示，根节点（节点0）具有三个具有不同优先级的子节点。因此，根节点将根据其优先级使用SP算法调度子节点，其中零（0）作为最高优先级。节点1有三个ID分别为11,12和13的子节点，并且所有子节点具有相同的优先级（即优先级0）。因此，节点1将使用WFQ机制来对它们进行调度。某一子节点的WFQ权重，是相对于具有相同优先级的所有兄弟节点的权重总和而得出的，其中一 (1) 作为最低权重。

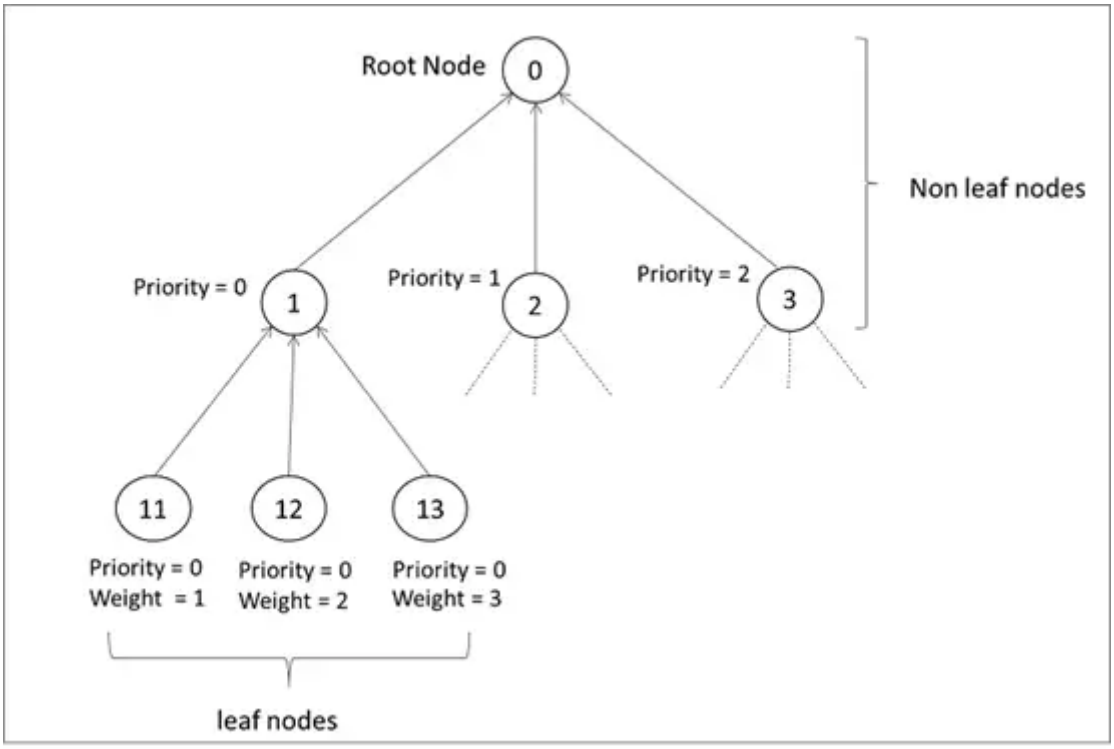

图1: 分层服务质量调度程序

流量整形

TM API支持为层次结构节点提供单速率和双速率整形器（速率限制器）两种选择，受限于可用的特定实现支持。每个层次结构节点有零个或一个私有整形器（仅供一个节点使用）和/或零个，一个或多个共享整形器（多个节点使用同一个整形器）。私有整形器用于为单个节点执行流量整形，而共享整形器用于为一组节点执行流量整形。层次结构节点的私有和共享整形器的配置都需要通过整形器配置文件的定义完成，如下所示：

```
/**
  * Shaper (rate limiter) profile
  */
struct rte_tm_shaper_params  {
        /** Committed token bucket */
        .committed = {
                /* bucket rate (bytes per second) */
                .rate = 0,
                .size = TOKEN_BUCKET_SIZE
        },
        /** Peak token bucket */
        .peak = {
                .rate = TM_NODE_SHAPER_RATE,
                .size = TOKEN_BUCKET_SIZE
        },
        /** framing overhead bytes */
        .pkt_length_adjust = RTE_TM_ETH_FRAMING_OVERHEAD_FCS,
};
```

图2：整形器配置文件参数。

如图1所示，每个非叶节点有多个输入（它的子节点）和单个输出（输入到它的父节点）。因此，非叶节点使用调度算法（例如，SP，WFQ等），在整形限制（速率限制）的条件下，调度它们的输入数据包到输出。

拥塞管理

可以通过API选择的拥塞管理算法是尾部丢弃（Tail Drop），头部丢弃（Head Drop）和加权随机早期检测（WRED，Weighted Random Early Detection）。它们可用于层次结构中的每个叶节点，受限于可用的特定实现支持。

发生拥塞时，尾部丢弃算法会丢弃新数据包，不对已有队列进行修改。头部丢弃算法则正相反，它会丢弃位于队列最前端的数据包（队列中等待最久的数据包）。

随着队列占用率的增加，随机早期检测（RED，The Random Early Detection）算法通过主动丢弃越来越多的输入数据包来工作。启用WRED作为其拥塞管理模式的每个层次结构叶节点有零个或一个私有WRED 环境（仅供一个叶节点使用）和/或零个，一个或多个共享加权随机早期检测（WRED）环境（多个叶节点使用同一个WRED环境）。私有WRED环境用于为单个叶节点执行拥塞管理，而共享WRED环境用于为一组叶节点执行拥塞管理。叶节点WRED私有和共享环境的配置都需要通过WRED配置文件的定义完成，如下所示：

```
/**
  * Weighted RED (WRED) profile
  */
struct struct rte_tm_wred_params {
        /* RED parameters */
         .red_params = {
                  [RTE_TM_GREEN] = {.min_th = 48, .max_th = 64, .maxp_inv = 10,
.wq_log2 = 9},
                  [RTE_TM_YELLOW] = {.min_th = 40, .max_th = 64, .maxp_inv = 10,
.wq_log2 = 9},
                  [RTE_TM_RED] = {.min_th = 32, .max_th = 64, .maxp_inv = 10, .wq_log2
= 9}
           },
};
```

图3：WERD配置文件参数

包标记

TM API支持各种类型的数据包标记，例如虚拟局域网丢弃合格指示符（VLAN DEI）包标记（IEEE 802.1Q），TCP和流控制传输协议分组的IPv4 / IPv6显式拥塞通知标记（IETF RFC 3168）以及IPv4 / IPv6区分服务码点包标记（IETF RFC 2597）。

能力API

TM API允许用户查询支持应用程序的的流量管理实现（硬件 / 软件）的能力信息（即关键参数值）。 这些信息可以在端口级别，特定层次级别以及层次级别的特定节点上获得。

设置层次结构的步骤

初始层次规范

调度程序的层次结构是通过逐步添加节点构建出的调度树。调度树由叶节点和非叶节点组成。

每个叶节点位于当前以太网端口调度队列的顶端。因此，每个叶节点都各自具有在0 ...（N-1）这一范围内的预定义ID，其中N是当前以太网端口的调度队列的总数。非叶节点的ID由上述范围之外的应用程序生成，为叶节点保留。创建节点时分配给每个节点的唯一ID还用于节点配置更新或将与其子节点的连接。

添加到层次结构中的第一个节点将成为根节点（节点0，图1），随后添加的所有节点都必须作为该根节点的后代。根节点的父节点必须指定为RTE_TM_NODE_ID_NULL，并且只能有一个具有此父节点ID的节点（即根节点）。

在此阶段，层次结构还需通过一些限制性检查，对象通常是当前节点，其父节点及其同级节点。目前，由于层次没有完全定义，通常底层实现不执行实际动作。

层次结构提交

在端口初始化阶段（以太网端口启动之前），层次结构提交API可被调用，来冻结启动层次结构。该功能通常有以下步骤：

* 它通过节点添加API的连续调用，验证先前为当前端口定义的启动层次结构生效。

* 假设验证成功，它将执行所有必需的特定于实现的操作，以便在当前端口上安装特定的层次结构，并在端口启动后立即生效。

运行时层次结构更新

TM API为调度层次结构的即时更改提供支持;因此，在以太网端口启动后，可以调用节点添加/删除，节点挂起/恢复，父节点更新等操作，受限于可用的特定实现支持。不同实现支持的动态更新集通过端口功能集合进行发布。

典型的DPDK函数调用序列

- **以太网设备配置**
- **创建并添加整形器配置文件**
- **创建初始调度程序层次结构**
- **冻结和验证启动层次结构**
- **启动以太网设备**

附加信息

关于TM API的详细信息，参见以下内容：

•DPDK峰会视频：DPDK服务质量API

•TM API定义：源代码

•TM API实现：Intel XL710 NIC, Intel 82599 NIC, SoftNIC

•TM API和质量服务分层调度程序：DPDK程序员指南

关于作者

Jasvinder Singh, 英特尔的网络软件工程师。他的主要工作是在为DPDK开发数据平面功能，库和驱动程序。

Wenzhuo Lu, 英特尔DPDK的软件工程师。他是dpdk.org上的DPDK的开发者和维护人员。

Cristian Dumitrescu, 英特尔数据面的软件架构师。他是dpdk.org上的DPDK的开发者和维护人员。

DPDK开源社区

往期精选

DPDK Roadmap 18.05
DPDK release 18.02
Hyperscan Release 4.7.0
DPDK美国技术峰会专题——DPDK可以应用于Windows了？
DPDK美国技术峰会专题——DPDK在微软Azure上的应用
DPDK美国技术峰会专题——报文分发重器OPDL
DPDK中国技术峰会专题系列——如何利用DPDK Cryptodev框架加速FD. IO/VPP Crypto工作流
DPDK中国技术峰会专题系列——如何利用DPDK设计高性能应用架构
DPDK 报文调度/保序 终极解决方案 Event Dev 简介
Intel 82599ES 基于DPDK和Linux kernel的L2fwd性能比较
FD.IO/VPP和DPDK Cryptodev，会产生什么样的化学反应？

本文翻译自 Intel Developer Zone，点击阅读原文可查看英文版~

不关注

就捣蛋

长按上方二维码，关注“DPDK 开源社区”

阅读原文 投诉