# 19.Librte\_pdump库

librte\_pdump库为DPDK中的数据包捕获提供了一个框架。该库将Rx和Tx mbufs的完整复制到新的mempool，因此会降低应用程序的性能，故建议只使用该库进行调试。

该库提供以下API来初始化数据包捕获框架，启用或禁用数据包捕获，或者对其进行反初始化：

* rte\_pdump\_init()：初始化数据包捕获框架。
* rte\_pdump\_enable()：在给定的端口和队列上进行数据包捕获。注意：API中的过滤器选项是用于未来增强功能的占位符。
* rte\_pdump\_enable\_by\_deviceid()：启用在给定设备ID（vdev名称或pci地址）和队列上的数据包捕获。 注意：API中的过滤器选项是用于未来增强功能的占位符。
* rte\_pdump\_disable()：禁用给定端口和队列上的数据包捕获。
* rte\_pdump\_disable\_by\_deviceid()：禁用给定设备ID（vdev名称或pci地址）和队列上的数据包捕获。
* rte\_pdump\_uninit()：反初始化数据包捕获框架。
* rte\_pdump\_set\_socket\_dir()：设置服务器和客户端套接字路径。注意：此API不是线程安全的。

## 19.1.操作

librte\_pdump库适用于客户端/服务器型号。服务器负责启用或禁用数据包捕获，客户端负责请求启用或禁用数据包捕获。

数据包捕获框架作为程序初始化的一部分，在pthread中创建pthread和服务器套接字。调用框架初始化的应用程序将创建服务器套接字，可能是在应用程序传入的路径，也可能是默认路径（root用户的/var/run/.dpdk，非root用户～/.dpdk）下创建。

请求启用或禁用数据包捕获的应用程序将在应用程序传入的路径下或默认路径（root用户的/var/run/.dpdk，非root用户～/.dpdk）下创建客户机套接字，用户将请求发送到服务器。服务器套接字将监听用于启用或禁用数据包捕获的客户端请求。

## 19.2.实现细节

库API rte\_pdump\_init()通过创建pthread和服务器套接字来初始化数据包捕获框架。pthread上下文中的服务器套接字将监听客户端请求以启用或禁用数据包捕获。

库API rte\_pdump\_enable()和rte\_pdump\_enable\_by\_deviceid()启用数据包捕获。每次调用这些API时，库创建一个单独的客户端套接字，生成“pdump enable”请求，并将请求发送到服务器。在套接字上监听的服务器将通过对给定的端口或设备ID和队列组合的以太网Rx/TX注册回调函数来接收请求并启用数据包捕获。然后，服务器将镜像数据包到新的mempool并将它们入队到客户端传递给这些API的rte\_ring。服务器还将响应发送回客户端，以了解处理过的请求的状态。从服务器收到响应后，客户端套接字关闭。

库API rte\_pdump\_disable()和rte\_pdump\_disable\_by\_deviceid()禁用数据包捕获。每次调用这些API时，库会创建一个单独的客户端套接字，生成“pdump disable”请求，并将请求发送到服务器。正在监听套接字的服务器将通过对给定端口或设备ID和队列组合的以太网RX和TX删除回调函数来执行请求并禁用数据包捕获。服务器还将响应发送回客户端，以了解处理过的请求的状态。从服务器收到响应后，客户端套接字关闭。

库API rte\_pdump\_uninit()通过关闭pthread和服务器套接字来初始化数据包捕获框架。

库API rte\_pdump\_set\_socket\_dir()根据API的类型参数将给定路径设置为服务器套接字路径或客户端套接字路径。如果给定路径为NULL，则将选择默认路径（即root用户的/var/run/.dpdk或非root用户的～/.dpdk）。如果服务器套接字路径与默认路径不同，客户端还需要调用此API来设置其服务器套接字路径。

## 19.3.用例:抓包

DPDK应用程序/pdump工具是基于此库开发的，用于捕获DPDK中的数据包。用户可以用它来开发自己的数据包捕获工具。