实验四 利用DPDK加速DNS查询(Part 2)

1. **实验目的**
2. 通过学习基于DPDK的多线程编程，实现多线程版本的SimpleDNS(using DPDK)。
3. 探索网卡的硬件特性，并在多线程程序中加以利用。
4. **实验内容**
5. 实验三中的SimpleDNS程序，在结合DPDK之后仍无法避免丢包，简单分析其原因。
6. 传统的包转发处理模型分为run to completion模型和pipeline模型(参考书第5章)，可任选其中一种，在实验三得到的程序基础上，拓展实现一个多线程的SimpleDNS(using DPDK)。以下给出两种模型的参考实现过程：

* Run to completion模型

结合网卡的多队列特性，并应用流分类(参考书第8章)，将收到的查询包均匀分散到不同的队列上处理，多线程程序运行时各个处理核心互相独立。参考[/examples/l2fwd](https://doc.dpdk.org/guides-18.11/sample_app_ug/l2_forward_real_virtual.html)，实现run to completion模型的多线程程序。

(注意：若虚拟环境下，虚拟网卡不能很好地支持多队列特性，可直接到服务器上调试程序。)

* Pipeline模型：

对SimpleDNS的处理流程合理分段，多线程程序运行时各个处理核心对应不同功能模块，利用DPDK无锁环形缓冲进行同步(参考书第4章)，协同工作。参考[/examples/distributor](https://doc.dpdk.org/guides-18.11/sample_app_ug/dist_app.html)，实现pipeline模型的多线程程序。

Optional：上述参考的examples中都使用了signal信号处理，使用信号可以更合理地结束程序，并便于在退出前输出统计信息。

1. **回答问题**

官方文档：[DPDK documentation](http://doc.dpdk.org/guides-18.11/), [DPDK API](http://doc.dpdk.org/api-18.11/)

注意：不同版本DPDK对应的文档内容也不完全一样，应对应查看

1. 给出选择run to completion(或pipeline)模型的理由。
2. 参考实验三的测试方法，在服务器上测试程序性能，给出具体数据。
3. **思考题**

无。

1. **报告提交**

本实验的完成情况以及对以上1个问题的回答作为第四阶段的进展报告，于12 月 3日之前提交给助教。12月5日课堂讨论。

进展实验报告需注明小组成员（组长排在第一个），以及每位成员对该报告的贡献比例。上课前各小组做好PPT，每个小组派一位同学上台主讲，其余同学可以补充。文件命名及格式：第四阶段\_组长姓名.pdf。

实验过程中遇到任何问题，请及时发邮件给助教（mig@mail.ustc.edu.cn），并抄送一份给主讲老师。