Hi309a

DPDK 应用手册

文档版本　04

发布日期　2026-02-27

版权所有 © 海思技术有限公司2026。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明

、HISILICON、海思和其他海思商标均为海思技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

海思技术有限公司

地址：　上海市青浦区虹桥港路2号101室　邮编：201721

网址：　https://www.hisilicon.com/cn/

客户服务邮箱：　support@hisilicon.com

前 言

概述

本文档介绍了Hi309a DPDK支持的示例应用以及使用方法，并提供了相关性能指标参考数据。

读者对象

本文档主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
危险	表示如不避免则将会导致死亡或严重伤害的具有高等级风险的危害。
警告	表示如不避免则可能导致死亡或严重伤害的具有中等级风险的危害。
注意	表示如不避免则可能导致轻微或中度伤害的具有低等级风险的危害。
须知	用于传递设备或环境安全警示信息。如不避免则可能会导致设备损坏、数据丢失、设备性能降低或其他不可预知的结果。 “须知”不涉及人身伤害。
说明	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

版权所有 © 海思技术有限公司

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

修订日期	版本	修订说明
2025-11-26	01	第1次正式版本发布，新增DPDK应用示例说明、DPDK内存池应用说明。
2025-12-12	02	第2次正式版本发布。 更新**表4-1**摘要长度范围；更新**对称加解密性能测试参考结果**章节表格中算法名称大小写。
2026-01-21	03	第3次正式版本发布。 新增simple-mp、symmetric-mp多进程示例使用说明； 新增非对称加解密test/test_cryptodev_asym示例使用说明； 新增testpmd转发示例使用说明； 新增加解密设备使用说明； 调整目录结构。
2026-02-27	04	第4次正式版本发布。 testpmd新增支持port调速命令。

版权所有 © 海思技术有限公司

目 录

前 言 i
1 概述 1
2 产品版本 3
3 内存池使用说明 4
3.1 概述 4
3.2 基于 BMU 模块内存池管理 5
3.2.1 内存池命名 5
3.2.2 配置内存池 ops 钩子函数 5
3.2.3 注意事项 5
3.3 基于 DPDK RING 内存池管理 6
3.3.1 内存池命名 6
3.3.2 配置内存池 ops 钩子函数 6
3.3.3 注意事项 6
4 加解密设备使用说明 7
4.1 对称加解密设备 7
4.1.1 对称加解密算法支持 8
4.1.2 注意事项 9
4.2 非对称加解密设备 10
4.2.1 非对称加解密算法支持 10
4.2.2 硬件引擎加载 11
4.2.3 注意事项 12
5 示例说明 13
5.1 转发示例使用说明 13
5.1.1 DTS 配置 14
5.1.1.1 非 RSS 与 RSS 转发模式配置方式 14
5.1.2 DPDK 编译 15
5.1.3 镜像升级与运行环境配置 15
5.1.4 L2FWD 转发示例使用说明 16
5.1.4.1 L2FWD 示例启动 16
5.1.4.2 注意事项 17
5.1.5 L3FWD 转发示例使用说明 17

 版权所有 © 海思技术有限公司

5.1.5.1 L3FWD 示例启动 17
5.1.5.2 注意事项 18
5.1.6 Testpmd 示例使用说明 18
5.1.6.1 Testpmd 示例启动 18
5.1.6.1.1 启动命令说明 18
5.1.6.1.2 交互模式命令说明 20
5.1.6.2 注意事项 21
5.1.7 KNI 示例使用说明 22
5.1.7.1 KNI 示例启动 22
5.1.7.2 验证方法 23
5.1.7.3 注意事项 26
5.2 加解密示例使用说明 26
5.2.1 DTS 配置 26
5.2.2 DPDK 编译 26
5.2.3 镜像升级与运行环境 27
5.2.4 对称加解密示例使用说明 27
5.2.4.1 test-crypto-perf 示例使用说明 27
5.2.4.1.1 示例支持算法 27
5.2.4.1.2 自定义测试向量编写模板 27
5.2.4.1.3 示例启动 28
5.2.4.1.4 注意事项 31
5.2.5 非对称加解密示例使用说明 31
5.2.5.1 test/test_cryptodev_asym 示例使用说明 31
5.2.5.1.1 示例启动 31
5.2.5.1.2 注意事项 31
5.3 多进程示例使用说明 31
5.3.1 DTS 配置 32
5.3.2 DPDK 编译 32
5.3.3 镜像升级与运行环境 32
5.3.4 simple-mp 32
5.3.4.1 simple-mp 示例启动 32
5.3.4.2 simple-mp 示例使用 33
5.3.5 symmetric-mp 34
5.3.5.1 symmetric-mp 示例启动 34
5.3.5.2 symmetric-mp 示例使用 35
6 性能测试结果 37
6.1 转发性能测试结果 37
6.1.1 测试环境与拓扑 37
6.1.2 打流仪配置 38
6.1.3 测试结果 42
6.2 对称加解密性能测试参考结果 44
6.2.1 测试环境 44

 版权所有 © 海思技术有限公司

6.2.2 测试结果 44
6.3 非对称加解密性能测试参考结果 48
6.3.1 测试环境 48
6.3.2 测试结果 48

 版权所有 © 海思技术有限公司

插图目录

图 3-1 DPDK 内存层级结构框图 4
图 4-1 Hi309a DPDK 支持加解密设备软件结构 7
图 5-1 Hi309a DPDK 非 RSS 转发方案核心与端口对应关系 13
图 5-2 Hi309a DPDK RSS 转发方案核心与端口对应关系 14
图 5-3 轮询周期性打印网口链路状态 log 23
图 5-4 KNI 虚拟设备 24
图 5-5 KNI 单板 ping 测试 24
图 5-6 对端单板抓包测试 25
图 5-7 KNI 应用内部数据包统计信息 25
图 5-8 主进程与从进程启动 33
图 5-9 支持指令查看操作 33
图 5-10 主从进程交互消息 34
图 6-1 309a DPDK L2、L3 层 RSS 转发方案硬件拓扑示意 37
图 6-2 配置测试使用流模式 38
图 6-3 配置报文为 UDP 格式 39
图 6-4 配置报文的源 ip 和目的 ip 39
图 6-5 配置流支持散列到多队列 40
图 6-6 使用吞吐量测试模式 41
图 6-7 测试套使用步骤 1 配置的流进行测试 41
图 6-8 配置检测乱序（Out of Sequence）、初始速度（Initial Rate），启动性能测试 42

版权所有 © 海思技术有限公司

表格目录

表 1-1 Hi309a 支持 DPDK 应用列表 1
表 4-1 Hi309a DPDK 支持的加解密算法与数据摘要算法 8
表 4-2 309a 支持硬件加速的 OpenSSL 非对称加解密算法接口 10
表 5-1 设备树配置文件 dts_config.h 参数 14
表 5-2 启动参数解析 18
表 5-3 控制命令 20
表 5-4 配置命令 20
表 5-5 查看命令 21
表 5-6 启动参数解析 29
表 6-1 XGE 网口双向打流性能测试结果 42
表 6-2 DPDK 下 Hi309a SEC 模块加密性能参考结果 44
表 6-3 DPDK 下 Hi309a SEC 模块认证性能参考结果 45
表 6-4 DPDK 下 Hi309a SEC 模块先加密后认证性能参考结果 46
表 6-5 非对称加解密性能测试参考结果 48

版权所有 © 海思技术有限公司

1 概述

本文档介绍了Hi309a DPDK（Data Plane Development Kit）的使用方法及支持的示例，并提供了相关业务的性能指标参考数据。

表 1-1 Hi309a 支持 DPDK 应用列表

业务类型	示例名称	示例描述
网络转发	l2fwd	此示例演示了每个RX_PORT上接收的每个数据包执行L2转发，支持非RSS（Receive Side Scaling，网卡的多队列负载分流）转发方案和RSS转发方案，RSS转发方案可用于测试Hi309a DPDK应用场景中L2层最大线速转发性能。
	l3fwd	此示例演示了每个RX_PORT上接收的每个数据包执行L3转发，仅支持RSS转发方案，RSS转发方案可用于测试Hi309a DPDK应用中L3层最大线速转发性能。
	testpmd	此示例是DPDK提供的一个能利用PMD转发数据包的程序，可以通过此示例快速测试硬件的包转发功能和能力。
加解密	test-crypto-perf	此示例演示了测试Hi309a SEC模块执行对称加解密硬件加速功能，支持吞吐量和延迟的性能测试。
	test/test_cryptodev_asym	此示例演示了测试Hi309a HPRE模块执行非对称加解密硬件加速功能，支持吞吐量性能测试。
多进程	simple-mp	此示例演示了两个DPDK进程如何使用队列和内存池来共同工作以共享信息。
	symmetric-mp	此示例演示了一组进程如何并行运行，每个进程执行相同的一组数据包处理操作。

版权所有 © 海思技术有限公司

须知

Hi309a SDK支持多版本DPDK（21.11.0、21.11.1），**表1-1**中的应用在DPDK（21.11.0、21.11.1）中均支持。

版权所有 © 海思技术有限公司

2 产品版本

本文档相对应的产品版本如下。

产品名称	产品版本
HiSolarA2RT	HiSolarA2RT_V100R002C10

版权所有 © 海思技术有限公司

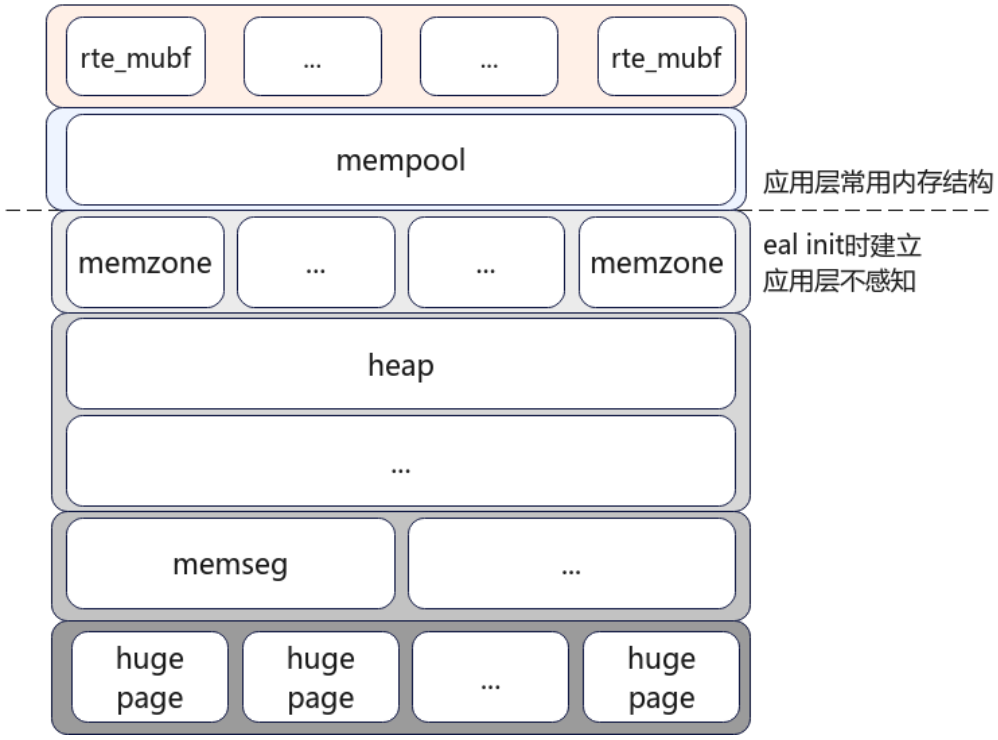
3 内存池使用说明

3.1 概述

DPDK对内存的管理进行了分层处理，如图3-1所示。mempool是应用层主要操作的内存对象，在Hi309a DPDK中支持BMU、RING两种mempool的管理方式，此两种内存池管理方式支持共存且共同使用。

- BMU内存池管理方式是通过Hi309a BMU模块实现内存池硬件管理，此种方式可以极大的简化缓存的申请与释放过程，减少软件开销，通过BMU申请的内存池对象支持自动与芯片其他模块进行交互。
- RING内存池管理方式是通过无锁环形队列实现内存池软件管理，相比于BMU方式内存池管理在缓存申请和释放时会有更大的软件开销。

图 3-1 DPDK 内存层级结构框图



版权所有 © 海思技术有限公司

3.2 基于 BMU 模块内存池管理

Hi309a BMU最多支持16个内存池的管理，内存池的命名需要遵循一定约束规则，才可保证通过BMU获得内存池。

通过BMU申请内存池有两个关键点：

1. 内存池命名正确。
2. 配置正确的内存池ops钩子函数。

3.2.1 内存池命名

通过BMU申请内存池，内存池命名必须以“BMUx”为结尾，x取值为[0,15]，对内存池的前缀命名无限制。

正确内存池命名示例：

```
BMU0
BMU1
BMU15
test_mp_BMU0
```

错误内存池命名示例：

```
BMU16              ----内存池id超过15
test_mp_BMU        ----缺少内存池id
BMU0_test_mp       ----内存池名称后缀不符要求
```

3.2.2 配置内存池 ops 钩子函数

DPDK申请内存池一般是由rte_mempool_create_empty()、rte_mempool_set_ops_byname()、rte_mempool_populate_default()这三个函数接口或对这三个接口的封装接口实现。内存池是通过何种方式申请和管理是由rte_mempool_set_ops_byname()接口决定的。必须保证rte_mempool_set_ops_byname()接口的第二个入参为“udrv_bmu”，才可确保内存池的ops函数使用udr_bmu驱动支持的函数，此种方式实现内存池通过BMU申请和管理。

```
struct rte_mempool **pool;

const char *mp_ops_name = "udrv_bmu";
int ret = rte_mempool_set_ops_byname(*pool, mp_ops_name, NULL);
if (ret != 0) {
        RTE_LOG(ERR, USER1, "Error setting mempool handler\n");
        return -1;
}
```

3.2.3 注意事项

版权所有 © 海思技术有限公司

须知

1. 转发业务相关的内存必须使用BMU申请的内存池、用于SEC模块处理的内存池必须使用BMU申请的内存池。其他业务相关的内存池、不涉及硬件交互的内存池可以使用BMU方式申请或RING方式申请。
2. BMU申请的内存池最多支持16个，当转发业务使用的内存池超过16个时，可以通过共用内存池方式解决。在申请内存池需要注意内存池buffer size、buffer数量可以同时满足业务需求。此外，可以适当增大内存池中资源数量，避免内存池空闲buffer资源耗尽的情况。
3. 使用BMU申请内存池命名推荐使用“BMUx”，不使用前缀，x取值为[0,15]。
4. 凡是通过“rte_mbuf_best_mempool_ops()”获取内存池操作名称，均是通过BMU方式申请和管理内存池。

3.3 基于 DPDK RING 内存池管理

通过DPDK RING申请的内存池在命名、数量均无相关约束。正确配置的内存池ops钩子函数即可通过RING申请内存池。

3.3.1 内存池命名

通过RING申请内存池，内存池命名通常无约束，支持任意命名。

3.3.2 配置内存池 ops 钩子函数

DPDK申请内存池一般是由rte_mempool_create_empty()、rte_mempool_set_ops_byname()、rte_mempool_populate_default()这三个函数接口或对这三个接口的封装接口实现。内存池是通过何种方式申请和管理是由rte_mempool_set_ops_byname()接口决定的。控制rte_mempool_set_ops_byname()接口的第二个入参，可保证内存池的ops函数使用RING驱动支持的函数，此种方式实现内存池通过RING申请和管理。

```
struct rte_mempool **pool;

const char *mp_ops_name = "ring_mp_mc";
int ret = rte_mempool_set_ops_byname(*pool, mp_ops_name, NULL);
if (ret != 0) {
        RTE_LOG(ERR, USER1, "Error setting mempool handler\n");
        return -1;
}
```

3.3.3 注意事项

须知

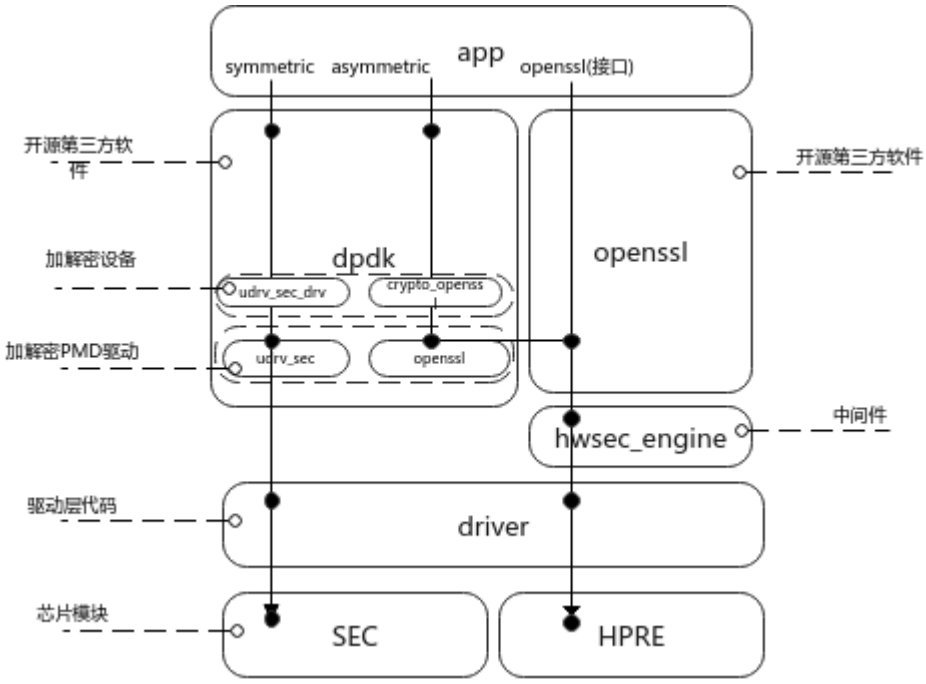
1. 通过RING申请内存池的ops函数有多种类型，包括“ring_mp_mc”、“ring_sp_sc”、“ring_mp_sc”、“ring_sp_mc”等，通常使用多生产者多消费者类型（“ring_mp_mc”）ops，用户可根据业务进行调整。
2. 非转发业务相关的内存池可以使用RING方式申请，并且数量无限制。

版权所有 © 海思技术有限公司

4 加解密设备使用说明

Hi309a加速器包含SEC和HPRE模块，分别为对称加解密和非对称加解密提供硬件加速。在Hi309a DPDK中，对称加解密通过加解密设备udrv_sec_drv直接调用SEC模块硬件加速能力，非对称加解密通过虚拟加解密设备crypto_openssl调用HPRE模块硬件加速能力，Hi309a DPDK下加解密设备软件结构如图4-1所示。

图 4-1 Hi309a DPDK 支持加解密设备软件结构



4.1 对称加解密设备

Hi309a DPDK中对称加解密设备udrv_sec_drv挂载在platform总线，使用时直接调用加密设备udrv_sec_drv执行加解密任务即可。

版权所有 © 海思技术有限公司

4.1.1 对称加解密算法支持

表 4-1 Hi309a DPDK 支持的加解密算法与数据摘要算法

算法类型	算法名称	key_size	digest_size	iv_size	block_size	备注
加解密算法	AES_CBC	{16,32}, increment = 8	-	{16,16}, increment = 0	16	-
	AES_CTR	{16,32}, increment = 8	-	{16,16}, increment = 0	16	-
	AES_ECB	{16,32}, increment = 8	-	{0,0}, increment = 0	16	-
	3DES_CBC	{16,24}, increment = 8	-	{8,8}, increment = 0	8	-
	3DES_ECB	{16,24}, increment = 8	-	{0,0}, increment = 0	8	-
	DES_CBC	{16,16}, increment = 0	-	{8,8}, increment = 0	8	-
	SM4_CBC	{16,16}, increment = 0	-	{16,16}, increment = 0	16	-
数据摘要算法	AES_CMAC	{16,16}, increment = 0	{4,16}, increment = 4	-	64	-
	AES_XCBC_MAC	{16,16}, increment = 0	{12,12}, increment = 0	-	64	-
	MD5	{0,0}, increment = 0	{4,16}, increment = 4	-	64	-
	MD5_HMAC	{64,64}, increment = 0	{4,16}, increment = 4	-	64	-
	SHA1	{0,0}, increment = 0	{4,20}, increment = 0	-	64	-

版权所有 © 海思技术有限公司

算法类型	算法名称	key_size	digest_size	iv_size	block_size	备注
	SHA1_HMAC	{64,64}, increment = 0	{4,20}, increment = 0	-	64	-
	SHA224	{0,0}, increment = 0	{4,28}, increment = 4	-	64	-
	SHA224_HMAC	{64,64}, increment = 0	{4,28}, increment = 4	-	64	-
	SHA256	{0,0}, increment = 0	{4,32}, increment = 4	-	64	-
	SHA256_HMAC	{1,64}, increment = 1	{4,32}, increment = 1	-	64	-
	SHA384	{0,0}, increment = 0	{4,48}, increment = 4	-	64	-
	SHA384_HMAC	{64,64}, increment = 0	{4,48}, increment = 4	-	64	-
	SHA512	{0,0}, increment = 0	{4,64}, increment = 4	-	64	-
	SHA512_HMAC	{64,64}, increment = 0	{4,64}, increment = 4	-	64	-
	SM3_HMAC	{0,64}, increment = 4	{4,32}, increment = 4	-	64	-

4.1.2 注意事项

须知

1. Hi309a DPDK加解密仅支持表格中的算法，非表格中算法不支持。
2. DPDK通过Hi309a SEC模块支持多种加解密算法和数据摘要算法，为保证系统安全性，产品开发人员在选择密码算法时尽量选择成熟的、具有足够安全强度的算法。表4-1中描述算法的安全性请参考《 HiSolarA2RT V100R002C10 二次开发网络安全注意事项 》“2.4 密码算法安全使用注意事项章节”。

版权所有 © 海思技术有限公司

4.2 非对称加解密设备

Hi309a DPDK中非对称加解密设备crypto_openssl挂载在vdev总线，使用时需要先创建虚拟加解密设备crypto_openssl，才可调用加解密设备crypto_openssl执行加解密任务即可。

4.2.1 非对称加解密算法支持

表 4-2 309a 支持硬件加速的 OpenSSL 非对称加解密算法接口

算法	OpenSSL非对称加解密309a硬件加速支持接口	算法模式	秘钥位宽	说明
DH	DH_generate_key()	标准DH计算	1024bit/2048bit/3072bit/4096bit	-
	DH_compute_key()	标准DH计算	1024bit/2048bit/3072bit/4096bit	-
RSA	RSA_public_encrypt()	STD模式	1024bit/2048bit/3072bit/4096bit	-
	RSA_private_decrypt()	CRT模式	1024bit/2048bit/3072bit/4096bit	-
	RSA_private_encrypt()	CRT模式	1024bit/2048bit/3072bit/4096bit	-
	RSA_public_decrypt()	STD模式	1024bit/2048bit/3072bit/4096bit	-
ECDH	EC_KEY_generate_key()	-	128bit/192bit/256bit/320bit/384bit/521bit	当前版本dpdk OpenSSL-pmd驱动不支持，用户可直接调用OpenSSL接口。
	ECDH_compute_key()	-	128bit/192bit/256bit/320bit/384bit/521bit	

版权所有 © 海思技术有限公司

算法	OpenSSL非对称加解密309a硬件加速支持接口	算法模式	秘钥位宽	说明
ECDSA	ECDSA_sign()	-	128bit/192bit/256bit/320bit/384bit/521bit	
	ECDSA_verify()	-	128bit/192bit/256bit/320bit/384bit/521bit	

4.2.2 硬件引擎加载

基于OpenSSL硬件引擎（Engine）机制实现了Hi309a对OpenSSL非对称加解密部分接口的硬件加速。在通过OpenSSL调用Hi309a硬件加速能力前，必须先执行动态引擎加载。动态引擎的加载代码示例如下。

```
#include <openssl/engine.h> //必须包含头文件
#define HW_ENGINE_PARAM_NUM 5

struct egn_ctr {
    const char *cmd;
    const char *arg;
};

static int hardware_engine_load(void)
{
    printf("******************start hardware_engine_load*****************\r\n");

    /* Check whether the engine has been loaded. */
    ENGINE *existing_engine = ENGINE_by_id("hpre_engine");
    if (existing_engine != NULL) {
        printf("Hardware engine already loaded, skipping...\r\n");
        return 0;
    }

    /* Support for hardware-accelerated algorithm configuration */
    uint32_t algs = ENGINE_METHOD_RSA | ENGINE_METHOD_EC |
                    ENGINE_METHOD_PKEY_METHS | ENGINE_METHOD_DH;
    char egn_name[16] = "hpre_engine";
    struct egn_ctr eng_ctr_cmds[HW_ENGINE_PARAM_NUM] = {
        {.cmd = "SO_PATH",          .arg = "/usr/lib64/engines-3/libhwsec.so"},
        {.cmd = "ID",               .arg = egn_name},
        {.cmd = "LOAD",             .arg = NULL},
        {.cmd = "alg_select",       .arg = (const char *)&algs},
        {.cmd = "single_thread",    .arg = "1"}
    };
    /* Obtain a pointer to a dynamic engine structure,
       and subsequently use this structure pointer to dynamically load external engines without
needing to recompile. */
    ENGINE_load_dynamic();
    ENGINE *e = ENGINE_by_id("dynamic");
    if (e == NULL) {
        return -1;
    }
    for (int i = 0; i < HW_ENGINE_PARAM_NUM; i++) {
        struct egn_ctr *ctrl = &eng_ctr_cmds[i];
        int ret = ENGINE_ctrl_cmd_string(e, ctrl->cmd, ctrl->arg, 0);
```


版权所有 © 海思技术有限公司

```
        if (ret == 0) {
            printf("failed to op sec egn cmd(%s) arg(%s)\r\n", ctrl->cmd, ctrl->arg);
            goto L_err;
        }
    }
    if (ENGINE_init(e) == 0) {
        printf("ENGINE_init error\r\n");
        goto L_err;
    }
    /* Configure the hardware acceleration engine as the default engine. */
    if ((ENGINE_set_default(e, algs) == 0)) {
        printf("ENGINE_set_default error\r\n");
        goto L_err;
    }
    /* Register the engine to the global table. */
    if (!ENGINE_add(e)) {
        printf("ENGINE_add to the global table error\r\n");
        goto L_err;
    }
    ENGINE_up_ref(e);
    printf("**********hardware_engine_load success**********\r\n\r\n\r\n");
    return 0;
L_err:
    printf( "failed to load hwsec engine, clean resource\r\n");
    ENGINE_remove(e);
    ENGINE_free(e);
    return -1;
}
```

4.2.3 注意事项

须知

1. Hi309a SDK提供的OpenSSL硬件引擎内部使用系统大页内存进行数据存储，在通过加解密设备crypto_openssl调用Hi309a非对称加解密硬件加速能力时，必须预留大页内存给硬件引擎管理。可通过dpdk eal参数“--socket-mem”控制DPDK管理的大页内存空间，比如：

   “--socket-mem 1024”，参数单位：MByte。

2. 在通过加解密设备crypto_openssl调用Hi309a非对称加解密硬件加速能力时，必须在DPDK应用程序“rte_eal_init()”之后执行OpenSSL硬件引擎的动态加载，否则执行非对称加解密任务时使用的是OpenSSL软件计算，不是Hi309a的硬件加速能力。
3. 在通过加解密设备crypto_openssl调用Hi309a非对称加解密硬件加速能力时，必须通过eal参数“--vdev”创建虚拟加解密设备，“--vdev crypto_openssl”。
4. 当前版本dpdk（21.11.0）OpenSSL-pmd驱动不支持椭圆曲线密码算法，无法通过dpdk接口调用调用椭圆曲线密算法相关接口，用户可以直接调用OpenSSL接口，实现椭圆曲线密码算法。
5. DPDK通过Hi309a HPRE模块支持多种非对称加解密算法，为保证系统安全性，产品开发人员在选择密码算法规格时应选择具有足够安全强度的算法。表4-2中描述算法的安全性请参考《HiSolarA2RT V100R002C10 二次开发网络安全注意事项》“2.4 密码算法安全使用注意事项”章节。

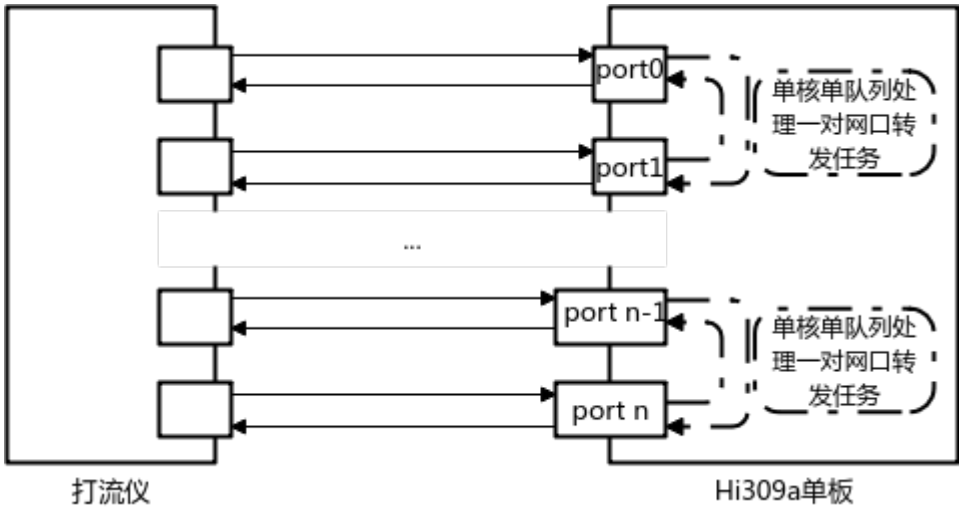
版权所有 © 海思技术有限公司

5 示例说明

5.1 转发示例使用说明

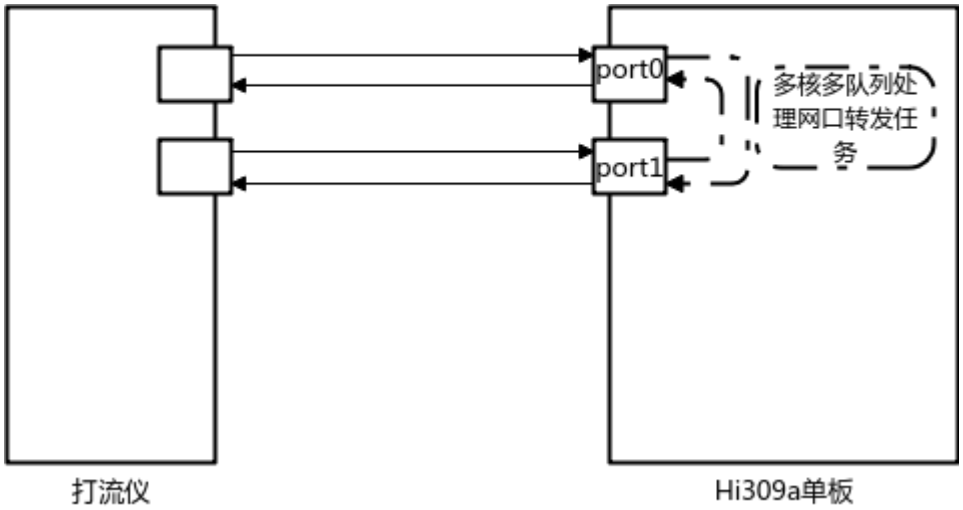
Hi309a DPDK 转发示例支持非RSS转发方案和RSS转发方案，非RSS转发方案中单核心单队列处理一对网口的转发任务，如图5-1所示，RSS转发方案中多核心多队列共同处理一对网口的转发任务，以提升转发性能，如图5-2所示。使用不同的转发方案还需要同步修改相关配置，主要为设备树配置文件dts_config.h文件的修改，详细修改内容请参考DTS配置和DPDK编译章节。

图 5-1 Hi309a DPDK 非 RSS 转发方案核心与端口对应关系



版权所有 © 海思技术有限公司

图 5-2 Hi309a DPDK RSS 转发方案核心与端口对应关系



5.1.1 DTS 配置

Hi309a DPDK支持的 L2层、L3层转发，在不同单板（demo板、m900板、m901板、m902板）、不同转发方案（非RSS转发方案和RSS转发方案）、不同网口类型（GE、XGE）应用场景中存在配置差异，需要修改设备树配置文件dts_config.h、示例应用层代码（宏开关）才可实现目标转发方式。

5.1.1.1 非 RSS 与 RSS 转发模式配置方式

Hi309a DPDK转发非RSS和RSS转发配置通过设备树配置文件（dts_config.h）控制，设备树配置文件详细说明参考表5-1。

Hi309a DPDK L2转发示例支持RSS转发方案和非RSS转发方案，默认使用非RSS转发方案。测试L2转发示例非RSS转发方案时，无需修改代码，测试L2转发示例RSS转发方案时需要修改代码和设备树配置文件（dts_config.h）。Hi309a DPDK L3转发仅支持RSS转发方案，不支持非RSS转发方案。Hi309a DPDK Testpmd示例支持RSS和非RSS转发方案，RSS和非RSS转发方式由启动参数控制，执行目标转发方式时必须确保与设备树配置文件dts_config.h参数适配。

表 5-1 设备树配置文件 dts_config.h 参数

参数宏	默认值	取值	说明
RSS_MODE_KERNAL	2	2	固定值，用户不可更改。内核转发业务模式下RSS模式。
RSS_MODE_DPDK	1	1	固定值，用户不可更改。DPDK转发业务模式下RSS模式。
RSS_MODE_DISABLE	0	0	固定值，用户不可更改。关闭RSS模式。

版权所有 © 海思技术有限公司

参数宏	默认值	取值	说明
RSS_MODE_SELECT	RSS_MODE_DISABLE	RSS_MODE_DISABLE、RSS_MODE_DPDK、RSS_MODE_KERNAL	RSS模式选择，用户配置。
RSS_RXQ_NUM	0	[0, 8]	RSS模式下接收队列数量，用户配置。

须知

1. 配置关闭RSS模式时，“RSS_RXQ_NUM”必须配置为0。
2. 使能RSS模式时，“RSS_RXQ_NUM”配置8、4、2、1才可保证每个队列拥有的分发通道数量是相等的，其他值7、6、5、3会造成队列拥有的分发通道数量存在1的数量差异。比如：配置接收队列数量为3，则3个接收队列拥有的分发通道数量分别为86、85、85。
3. Testpmd转发示例使用接收队列数量4，l2fwd、l3fwd转发示例使用接收队列为8。

5.1.2 DPDK 编译

版本编译请参考《Hi309a 系统编译运行指南》“统一镜像构建”章节。

须知

1. 编译参数-n，请选择包含DPDK的设备树，可选择（dpdk_21.11.0、dpdk_21.11.1、kernel_5.10.0_with_dpdk_21.11.0、kernel_5.10.0_with_dpdk_21.11.1、kernel_5.4.18_with_dpdk_21.11.0、kernel_5.4.18_with_dpdk_21.11.1），编译名为hi309a_xxx_xxx_dpdk.dts的设备树。
2. 使用RSS方案执行转发任务时，l2fwd示例中“RSS_ENABLE”宏定义配置为1。宏定义位置“mpu_hi309a_solution/open_source/dpdk/examples/l2fwd/main.c”。

5.1.3 镜像升级与运行环境配置

前提条件：根据所需按照**DTS配置**、**DPDK编译**说明编译对应设备树、系统镜像、文件系统。

步骤1 openEuler系统升级完成。

步骤2 进入openEuler系统后，执行如下命令：

```
cd /lib/udrivers
insmod motorcomm.ko
```


版权所有 © 海思技术有限公司

```
insmod hns_mdio.ko
insmod hi309a_urm.ko
insmod hi309a_urm_adapt.ko
insmod hi309a_poe.ko
insmod hi309a_ppe.ko
insmod hi309a_mag.ko
insmod igb_uio.ko
insmod rte_kni.ko
insmod hi309a_kdrv_dfx.ko
insmod dpdk_phy.ko

mkdir -p /dev/hugepages
mountpoint -q /dev/hugepages || mount -t hugetlbfs nodev /dev/hugepages
echo 1024 > /proc/sys/vm/nr_hugepages
```

----结束

须知

motorcomm.ko为phy相关的驱动。客户需要提供phy相关驱动源码。在文件制作过程中，客户需要把phy的源码编译成ko放入文件系统里面。

5.1.4 L2FWD 转发示例使用说明

Hi309a DPDK支持L2FWD示例，示例支持非RSS转发方案和RSS转发方案。参考**DTS配置**、**DPDK编译**、**镜像升级与运行环境配置**章节修改描述，获取示例的目标转发方式。

5.1.4.1 L2FWD 示例启动

步骤1 单板与网络相关测试仪器对接后，根据系统中网口的数量，动态修改执行的参数。

非RSS转发方案网口为8个时，可以执行如下命令：

```
非RSS方案
dpdk-l2fwd -l 0,1,2,3,4,5,6,7 -n 1 -- -q 1 -p 0xff -T 2000
```

RSS转发方案仅支持2个网口（port0、port1），可以执行如下命令：

```
RSS方案
dpdk-l2fwd -l 0,1,2,3,4,5,6,7 -n 1 -- -q 1 -p 0x3 -T 2000
```

参数说明：

```
-l  2,3 绑定在2 和 3号核上。
-n  每个CPU的内存通道数。
-q  每个CPU管理的队列数（等于端口数），默认配置一个队列。
-p  使用的端口的掩码: 0x1 表示2的0次方使用的是port 0 。
    使用的是port 1是2的一次方，也就是2  即0x2。
    使用的是port 2 ，就是2 的2次方，也就是4，即0x4。
    使用的是port 3 ，就是2 的3次方，也就是8，即0x8。
    则此时-p 的值是，2的0次方+ 2的1次方，也就是3，即此时参数为：  -p 0x3
-T 100 每隔100s显示一下转发的统计计数，此处可以设置大一些，防止频繁打印导致性能下降。
--portmap 指定端口映射关系。比如--portmap="(1,2),(0,3)"  第一个括号（1，2）表示端口1，2作为一个收发对，在第1个核心运行（核1）；第二个括号(0,3）表示端口0，3作为一个收发对，绑定到第2个核心（核2）
```

如果需要结束运行，则按CRTL+C结束DPDK的运行。

----结束

版权所有 © 海思技术有限公司

5.1.4.2 注意事项

须知

1. 在RSS转发方案中仅可使能一对转发口进行测试，否则可能会转发异常问题。比如：测试port0-port1配置“-p 0x03”，测试port2-port3配置“-p 0xc”，测试port0-port3配置0x9。
2. 在RSS转发方案中l2fwd转发示例使用接收队列数量为8，设备树配置文件（dts_config.h）中宏参数“RSS_RXQ_NUM”需配置为8，“RSS_MODE_SELECT”配置为“RSS_MODE_DPDK”。
3. 在RSS转发方案中l2fwd转发示例代码需要修改“RSS_ENABLE”数值为1，数值含义0：关闭RSS转发方案，1：使能RSS转发方案。宏定义路径：“open_source/dpdk/examples/l2fwd/main.c”。
4. 在非RSS转发方案中l2fwd转发示需将设备树配置文件（dts_config.h）中宏参数"RSS_RXQ_NUM"需配置为0，"RSS_MODE_SELECT"配置为"RSS_MODE_DISABLE"。
5. 在进行性能测试过程中不可有终端打印log信息，会影响转发性能，可将-T参数配置为超长时间避免打印，如：“-T 20000”。
6. 当错误命令或其他原因导致启动失败时，再次启动需要先执行重启命令，再执行测试命令。重启命令如下：

```
umount /dev/hugepages
mountpoint -q /dev/hugepages || mount -t hugetlbfs nodev /dev/hugepages
rm -rf /var/run/dpdk
```

5.1.5 L3FWD 转发示例使用说明

Hi309a DPDK支持DPDK的L3FWD示例，示例仅支持RSS转发方案。参考**DTS配置**、**DPDK编译**和**镜像升级与运行环境配置**章节修改描述，获取示例的目标转发方式。

5.1.5.1 L3FWD 示例启动

步骤1 单板与网络相关测试仪器对接后，执行l3fwd启动参数。

```
dpdk-l3fwd -l 0,1,2,3,4,5,6,7 -n 1 -- -p 0x3 --config="(0,0,0),(0,1,1),(0,2,2),(0,3,3),(0,4,4),(0,5,5),(0,6,6),
(0,7,7)" --parse-ptype
```

参数说明：

```
-l  2,3 绑定在2 和 3号核上。
-n  每个CPU管理的队列数，这里设置为一个队列。
-p  使用的端口的掩码: 0x1 表示2的0次方使用的是port 0 。
    使用的是port 1是2的一次方，也就是2  即0x2。
    使用的是port 2 ，就是2 的2次方，也就是4，即0x4。
    使用的是port 3 ，就是2 的3次方，也就是8，即0x8。
    则此时-p 的值是，2的0次方+ 2的1次方，也就是3，即此时参数为：  -p 0x3
-config 三元组（端口id,核id，队列id）,队列id必须从0开始。
 --parse-ptype 使能包类型解析功能。
```

如果需要结束运行，则按CRTL+C结束DPDK的运行。

----结束

版权所有 © 海思技术有限公司

5.1.5.2 注意事项

须知

1. RSS转发方案中port0、port1对应为DTS中dpdketh@0、dpdketh@1中对应的物理端口。prot0、port1与硬件单板网口的映射关系需查看原理图确认。
2. 示例中IP与端口绑定关系是IP为198.18.0.xx的报文从port0发送，IP为198.18.1.xx的报文从port1发送。
3. 在RSS转发方案中l3fwd转发示例使用接收队列数量为8，设备树配置文件（dts_config.h）中宏参数“RSS_RXQ_NUM”需配置为8，“RSS_MODE_SELECT”配置为“RSS_MODE_DPDK”。
4. 示例中路由算法使用LPM算法。路由表使用DPDK示例中默认路由表：

   三元组{IP地址，匹配位数，Port}

   {RTE_IPV4(198, 18, 0, 0), 24, 0},

   {RTE_IPV4(198, 18, 1, 0), 24, 1}。
5. 当错误命令或其他原因导致启动失败时，再次启动需要先执行重启命令，再执行测试命令。重启命令如下：

```
umount /dev/hugepages
mountpoint -q /dev/hugepages || mount -t hugetlbfs nodev /dev/hugepages
rm -rf /var/run/dpdk
```

5.1.6 Testpmd 示例使用说明

5.1.6.1 Testpmd 示例启动

5.1.6.1.1 启动命令说明

启动命令：非RSS转发方案启动命令示例

```
dpdk-testpmd -l 0-7 -n 1 -- -i --portmask=0xf --nb-cores=7 --disable-rss --no-lsc-interrupt
```

启动命令：RSS转发方案启动命令示例

```
dpdk-testpmd -l 0-7 -n 1 -- -i --portmask=0x3 --nb-cores=4 --coremask=0xf0 --rxq=4 --txq=4 --no-lsc-interrupt
```

表 5-2 启动参数解析

配置项	含义	取值范围	备注
-l	0,1,2,3,4,5,6,7	示例使用的核心数。	默认0核作为主核，不参与转发测试，作为命令解析使用。
-n	1	每个CPU的内存通道数。	-

版权所有 © 海思技术有限公司

配置项	含义	取值范围	备注
-i, --interactive	NA	以交互模式运行testpmd。	不带此参数，使用自动模式运行testpmd，转发模式使用配置模式，如果未配置，默认使用io转发。
--tx-first	NA	首先发送brust数量数据包后开始转发，默认brust=32。	所有端口都会发送数据brust数量的数据包。 仅在自动模式下生效，交互模式下无效。
-a, --auto-start	NA	自动执行默认转发模式。	相比于交互模式，不用执行“start”命令，testpmd会自动执行默认转发模式。
--nb-cores	1≤ N ≤ -l参数的数量	设置转发核数。	testpmd会自动规划核心转发任务。
--nb-ports	2 ≤ N ≤ 单板上的“端口数”	设置转发端口数。	-
--coremask	[0x2, 0xFE]	转发核心的十六进制掩码，使用核心掩码：0x2 ，表示使用core 1 。	此掩码不能包含主核。
--portmask	[0x1, NA]	转发端口的十六进制位掩码，使用核心掩码：0x1 ，表示使用port1。	上限值根据单板实际使用端口确定。
--forward-mode	io、mac、rxonly、txonly	转发模式选择。	默认为io转发模式。
--disable-rss	-	关闭RSS模式，取消此配置命令则默认使能RSS模式。	-
--rxq	[1, 8]	设置每个端口的接收队列数为N，默认值为1。	非RSS转发方案缺省配置为1。
--txq	[1, 8]	设置每个端口的发送队列数为N，默认值为1。	非RSS转发方案缺省配置为1。
--no-lsc-interrupt	-	禁用所有端口的LSC中断	-

版权所有 © 海思技术有限公司

5.1.6.1.2 交互模式命令说明

在执行完testpmd启动命令后，如果是交互模式运行，可继续输入交互命令实现启动转发、查看信息等。

表 5-3 控制命令

命令示例	取值范围	功能含义
start	-	启动数据包转发功能。
stop	-	停止数据包转发功能。
quit	-	退出testpmd应用。
start tx_first	-	先发送brust数量的数据包后再启动数据包转发功能。

表 5-4 配置命令

命令示例	取值范围	功能含义	备注
set fwd io	io、mac、rxonly、txonly	配置转发模式示例	必须先执行“stop”命令才可配置。
set portmask 0xc	[0x1, NA]	配置转发网口的十六进制掩码	可使用此命令实现端口屏蔽。必须先执行“stop”命令才可配置。
set nbport 2	2 ≤ N ≤ 单板上的“端口数	配置转发网口数量	必须先执行“stop”命令才可配置。
set coremask 0xf0	[0x2, 0xFE]	配置使用核心的十六进制掩码	使用核心掩码：0x2，表示使用core 1，不能包含主核。必须先执行“stop”命令才可配置。
set nbcore 4	[1, 7]	配置使用转发核心数量	必须先执行“stop”命令才可配置。
port start (port_id|all)	-	启动转发网口的收发功能	-
port stop (port_id|all)	-	停止转发网口的收发功能	-

版权所有 © 海思技术有限公司

命令示例	取值范围	功能含义	备注
port config (port_id) speed (10|100|1000|10000) duplex (full)	-	转发网口速率配置	必须先执行“port stop (port_id)”命令，才可执行速率配置。 GE网口速率切换支持10Mbps、100Mbps、1000Mbps规格，XGE网口速率切换支持1000Mbps(1Gbps)、10000Mbps(10Gbps)规格。 速率配置参数存储在结构体中，在执行“port start (port_id)”时生效。

表 5-5 查看命令

命令示例	取值范围	功能含义	备注
show fwd stats all	-	显示转发引擎在它们启动时的转发统计信息。	仅支持非RSS转发方式下。
show port summary all	summary、info、xstats	显示所有端口简要摘要信息，如设备名称、驱动程序名称等。	-
show config fwd	fwd、rxtx、cores	显示应用程序的配置。配置来自命令行、运行时或应用程序默认值。	-
clear fwd stats all	-	清空转发统计信息	-

5.1.6.2 注意事项

版权所有 © 海思技术有限公司

须知

1. 在RSS转发方案中testpmd示例使用接收队列数量为4，设备树配置文件（dts_config.h）中宏参数“RSS_RXQ_NUM”需配置为4，“RSS_MODE_SELECT”配置为“RSS_MODE_DPDK”。
2. 在RSS转发方案中仅可使能一对转发口进行测试，否则可能会转发异常问题。比如：测试port0-port1配置“-p 0x03”，测试port2-port3配置“-p 0xc”，测试port0-port3配置“0x9”。
3. 在非RSS转发方案中testpmd转发示需将设备树配置文件（dts_config.h）中宏参数“RSS_RXQ_NUM”需配置为0，“RSS_MODE_SELECT”配置为“RSS_MODE_DISABLE”。
4. 测试txonly模式时，仅允许使能1个端口验证发送功能，同时使能多个端口会导致内存池耗尽报错。
5. 测试端口调速配置命令“port config (port_id) speed (10|100|1000) duplex (full)”时，在程序启动命令时增加“--no-lsc-interrupt”，用于检测phy芯片速率协商成功更新状态。
6. 其余未在手册中说明的testpmd命令不支持。
7. 当错误命令或其他原因导致启动失败时，再次启动需要先执行重启命令，再执行测试命令。重启命令如下：

```
umount /dev/hugepages
mountpoint -q /dev/hugepages || mount -t hugetlbfs nodev /dev/hugepages
rm -rf /var/run/dpdk
```

5.1.7 KNI 示例使用说明

Hi309a DPDK支持内核网卡接口（Kernel NIC Interface，KNI）示例，支持使用ethtool、ifconfig、tcpdump等标准Linux网络工具对DPDK端口进行管理，并实现DPDK应用程序与Linux内核间的数据包交换。参考**DTS配置**、**DPDK编译**、**镜像升级与运行环境配置**章节修改描述，获取KNI示例。

说明

insmod rte_kni.ko加载rte_kni模块时，会创建设备“/dev/kni”，KNI通过该设备控制内核模块并与之通信。rte_kni内核模块包含多个可选参数，可在模块加载时指定参数以控制其行为。部分参数说明：

- lo_mode：环回模式，默认=lo_mode_none。
  lo_mode_none：禁用内核环回。
  lo_mode_fifo：启用基于 FIFO 的内核环回。
  lo_mode_fifo_skb：启用基于 FIFO 和 skb 缓冲区的内核环回。
- kthread_mode：内核线程模式，默认=single。
  single：启用单内核线程模式。
  multiple：启用多内核线程模式。
- carrier：KNI 接口的默认载波状态，默认=off。
  off接口创建时载波状态设置为关闭。
  on接口创建时载波状态设置为开启。

5.1.7.1 KNI 示例启动

步骤1 单板与网络相关测试仪器对接后，根据系统中实际使用网口的数量和编号，动态修改执行的参数。

版权所有 © 海思技术有限公司

支持2个网口（port0、port1）时，可以执行如下命令：

```
./dpdk-kni -l 4-7 -n 1 -- -p 0x3 -m --config="(0,4,6),(1,5,7)"
```

参数说明：

```
-l  4-7 绑定在4、5、6和7号核上。
-n  每个CPU的内存通道数。
-P  可选，用于将所有端口设置为混杂模式。
-p  使用的端口的掩码: 0x1 表示2的0次方使用的是port0。
    使用的是port1，就是2的1次方，即0x2。
    使用的是port2，就是2的2次方，即0x4。
    使用的是port3，就是2的3次方，即0x8。
    则此时-p的值是，2的0次方+ 2的1次方，也就是3，即此时参数为： -p 0x3
-m  可选，用于启用以太网载波状态的监控与更新。设置此选项后，将启动一个线程定期检查物理以太网端口的以太网链路状态，并相应设置对应KNI网络接口的载波状态以匹配物理端口状态。
--config 指定端口与核心的映射关系。比如--config="(0,4,6),(1,5,7)，第一个括号（0,4,6）表示核心4作为端口0的接收核心，核心6作为端口0的发送核心；第二个括号(1,5,7）表示核心5作为端口1的接收核心，核心7作为端口1的发送核心。
```

如果需要结束运行，则按“CRTL+C”结束KNI示例的运行。

----结束

5.1.7.2 验证方法

步骤1 系统启动完成后，加载DPDK相关内核驱动模块。

```
cd /lib/udrivers
insmod motorcomm.ko
insmod hns_mdio.ko
insmod hi309a_urm.ko
insmod hi309a_urm_adapt.ko
insmod hi309a_poe.ko
insmod hi309a_ppe.ko
insmod hi309a_mag.ko
insmod igb_uio.ko
insmod rte_kni.ko
insmod hi309a_kdrv_dfx.ko
insmod dpdk_phy.ko
```

步骤2 配置大页内存并挂载大页文件系统。

```
mkdir -p /dev/hugepages
mountpoint -q /dev/hugepages || mount -t hugetlbfs nodev /dev/hugepages
echo 1024 > /proc/sys/vm/nr_hugepages
```

步骤3 执行“dpdk-kni”示例启动命令：

```
./dpdk-kni -l 4-7 -n 1 -- -p 0x3 -m --config="(0,4,6),(1,5,7)"
```

观察启动时的输出信息，启动后会首次检查打印界面如图5-3所示。

图 5-3 轮询周期性打印网口链路状态 log

```
update_phy_link_status: link_speed = 0, link_duplex = 1, link_autoneg = 1, link_status = 0
Port 0 Link down
update_phy_link_status: link_speed = 1000, link_duplex = 1, link_autoneg = 1, link_status = 1
Port 1 Link up at 1 Gbps FDX Autoneg
APP: ========================
APP: KNI Running
APP: kill -SIGUSR1 672
APP:     Show KNI Statistics.
APP: kill -SIGUSR2 672
APP:     Zero KNI Statistics.
APP: ========================
APP: Lcore 5 is reading from port 1
APP: Lcore 7 is writing to port 1
APP: Lcore 6 is writing to port 0
APP: Lcore 4 is reading from port 0
update_phy_link_status: link_speed = 0, link_duplex = 1, link_autoneg = 1, link_status = 0
APP: vEth0 NIC Link downupdate_phy_link_status: link_speed = 1000, link_duplex = 1, link_autoneg = 1, link_status = 1
APP: vEth1 NIC Link up at 1 Gbps FDX Autonegupdate_phy_link_status: link_speed = 0, link_duplex = 1, link_autoneg = 1, link_status = 0
APP: vEth0 NIC Link downupdate_phy_link_status: link_speed = 1000, link_duplex = 1, link_autoneg = 1, link_status = 1
APP: vEth1 NIC Link up at 1 Gbps FDX Autonegupdate_phy_link_status: link_speed = 0, link_duplex = 1, link_autoneg = 1, link_status = 0
```


版权所有 © 海思技术有限公司

步骤4 执行ifconfig -a命令，检查KNI虚拟设备vEth0和vEth1（对应启动命令参数-p 0x3，表示启用两个端口）是否已成功创建。

图 5-4 KNI 虚拟设备

```
hi309a ~ # ifconfig -a
ethmii    Link encap:Ethernet  HWaddr 5E:D0:B3:00:A0:B7
          inet addr:192.168.0.100  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::5cd0:b3ff:fe00:a0b7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:12644 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1050 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:18100549 (17.2 MiB)  TX bytes:190205 (185.7 KiB)
          Interrupt:13

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:12 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:896 (896.0 B)  TX bytes:896 (896.0 B)

sit0      Link encap:IPv6-in-IPv4
          NOARP  MTU:1480  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

vEth0     Link encap:Ethernet  HWaddr 3E:D9:95:A4:5A:11
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

vEth1     Link encap:Ethernet  HWaddr BA:5A:67:FC:A5:DE
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

步骤5 以vEth1为例，为其配置IP地址并启用接口：

```
ifconfig vEth1 192.168.1.112
ifconfig vEth1 up
```

步骤6 将vEth1对应的物理网口与外部测试设备的网口用网线连接，确保对端网口为up状态，并为其配置与vEth1同一子网的IP地址。

步骤7 在KNI单板上执行ping命令，向对端单板的IP地址发送ICMP请求，验证网络连通性是否正常。

图 5-5 KNI 单板 ping 测试

```
hi309a ~ # ping -I vEth1 192.168.1.110
PING 192.168.1.110 (192.168.1.110): 56 data bytes
64 bytes from 192.168.1.110: seq=0 ttl=64 time=0.118 ms
64 bytes from 192.168.1.110: seq=1 ttl=64 time=0.073 ms
64 bytes from 192.168.1.110: seq=2 ttl=64 time=0.070 ms
64 bytes from 192.168.1.110: seq=3 ttl=64 time=0.069 ms
```


版权所有 © 海思技术有限公司

图 5-6 对端单板抓包测试

```
hi309a /lib/udrivers # tcpdump -i eth7 icmp
[ 2340.437995] device eth7 entered promiscuous mode
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth7, link-type EN10MB (Ethernet), snapshot length 262144 bytes
13:13:59.849950 IP 192.168.1.112 > 192.168.1.110: ICMP echo request, id 744, seq 0, length 64
13:13:59.849981 IP 192.168.1.110 > 192.168.1.112: ICMP echo reply, id 744, seq 0, length 64
13:14:00.849976 IP 192.168.1.112 > 192.168.1.110: ICMP echo request, id 744, seq 1, length 64
13:14:00.849989 IP 192.168.1.110 > 192.168.1.112: ICMP echo reply, id 744, seq 1, length 64
13:14:01.850007 IP 192.168.1.112 > 192.168.1.110: ICMP echo request, id 744, seq 2, length 64
13:14:01.850018 IP 192.168.1.110 > 192.168.1.112: ICMP echo reply, id 744, seq 2, length 64
```

步骤8 向KNI应用程序发送SIGUSR1信号，查看应用输出的数据包统计信息，确认收发包计数符合预期。

```
kill -SIGUSR1 <pid>
kill -SIGUSR2 <pid>
```

图 5-7 KNI 应用内部数据包统计信息

```
**KNI example application statistics**
======  ==============  ============  ============  ============  ============
 Port    Lcore(RX/TX)   rx_packets    rx_dropped    tx_packets    tx_dropped
------  --------------  ------------  ------------  ------------  ------------
      0          4/ 6             0             0             0             0
      1          5/ 7            46             0            57             0
======  ==============  ============  ============  ============  ============
```

步骤9 键盘按“Ctrl+C”结束应用，然后重复**步骤3**启动应用，可以正常运行不报错。

步骤10 执行卸载驱动指令后，重复**步骤1**加载驱动，反复执行无异常。

```
cd /lib/udrivers
rmmod dpdk_phy.ko
rmmod hi309a_kdrv_dfx.ko
rmmod rte_kni.ko
rmmod igb_uio.ko
rmmod hi309a_mag.ko
rmmod hi309a_ppe.ko
rmmod hi309a_poe.ko
rmmod hi309a_urm_adapt.ko
rmmod hi309a_urm.ko
rmmod hns_mdio.ko
rmmod motorcomm.ko
```

----结束

版权所有 © 海思技术有限公司

5.1.7.3 注意事项

须知

1. 在KNI示例中，如果加载rte_kni.ko内核模块时未配置carrier=on，则运行示例时需要添加-m参数开启载波，否则链路状态始终为DOWN，无法进行数据通信。
2. 在KNI示例中，每个KNI设备通常需要占用多个CPU核心：两个核心用于运行DPDK轮询线程；一个核心（单内核线程模式）或多个核心（多内核线程模式）用于运行KNI内核线程；此外，若启用-m参数，还需一个额外核心用于以太网链路状态定期检查线程。因此，需要合理分配CPU核心（推荐启用端口数量不大于2个），避免DPDK轮询线程与KNI内核线程争用同一核心，必要时进行核心隔离。
3. 当KNI应用启动后，用户可像管理常规Linux网络接口一样，使用标准Linux命令来管理KNI接口，还可通过向kni应用发送SIGUSR1和SIGUSR2信号来查看及重置其内部数据包统计信息。
4. 若对端单板抓包正常，但本端KNI单板无法ping通，通常是由于反向路径过滤（Reverse Path Filtering，rp_filter）机制所致，可尝试关闭对应接口的rp_filter配置（设置为0）以解决该问题。
5. 当KNI应用关闭时，所有KNI接口都将从Linux内核中删除。
6. 当错误命令或其他原因导致启动失败时，再次启动需要先执行重启命令，再执行测试命令。重启命令如下：

```
umount /dev/hugepages
mountpoint -q /dev/hugepages || mount -t hugetlbfs nodev /dev/hugepages
rm -rf /var/run/dpdk
```

5.2 加解密示例使用说明

对称加解密的功能和性能测试，Hi309a DPDK支持test-crypto-perf应用，可通过此应用进行Hi309a SEC模块硬件加解密性能测试，支持7种加解密算法和15种数据摘要算法的吞吐量和延迟性能测试，支持算法请参考**表4-1**。

非对称加解密的功能和性能测试，Hi309a DPDK支持test/test_cryptodev_asym自动化测试程序，可通过此应用进行Hi309a HPRE模块硬件加解密性能测试，支持10种非对称加解密OpenSSL接口硬件加速。支持详细算法接口参考**非对称加解密算法支持**。

5.2.1 DTS 配置

Hi309a SDK包中DPDK的DTS文件默认已支持加解密功能，均无需要进行二次修改。

5.2.2 DPDK 编译

版本编译请参考《Hi309a 系统编译运行指南》“统一镜像构建”章节。

须知

编译参数-n，请选择包含DPDK的设备树，可选择（dpdk_21.11.0、dpdk_21.11.1、kernel_5.10.0_with_dpdk_21.11.0、kernel_5.10.0_with_dpdk_21.11.1、kernel_5.4.18_with_dpdk_21.11.0、kernel_5.4.18_with_dpdk_21.11.1），编译名为hi309a_xxx_xxx_dpdk.dts的设备树。

版权所有 © 海思技术有限公司

5.2.3 镜像升级与运行环境

前提条件：根据所需按照**DTS配置**、**DPDK编译**说明编译对应设备树、系统镜像、文件系统。

步骤1 openEuler系统升级完成。

步骤2 进入openEuler系统后，执行如下命令：

```
cd /lib/udrivers
insmod motorcomm.ko
insmod hns_mdio.ko
insmod hi309a_urm.ko
insmod hi309a_urm_adapt.ko
insmod hi309a_poe.ko
insmod hi309a_ppe.ko
insmod hi309a_mag.ko
insmod igb_uio.ko
insmod rte_kni.ko
insmod hi309a_kdrv_dfx.ko
insmod dpdk_phy.ko

mkdir -p /dev/hugepages
mountpoint -q /dev/hugepages || mount -t hugetlbfs nodev /dev/hugepages
echo 1024 > /proc/sys/vm/nr_hugepages
```

----结束

须知

motorcomm.ko为phy相关的驱动。客户需要提供phy相关驱动源码。在文件制作过程中，客户需要把phy的源码编译成ko放入文件系统里面。

5.2.4 对称加解密示例使用说明

5.2.4.1 test-crypto-perf 示例使用说明

5.2.4.1.1 示例支持算法

示例支持算法与规格特性，请参考**表4-1**。

5.2.4.1.2 自定义测试向量编写模板

测试程序含有默认的测试向量，用户也可使用自己测试向量。测试向量需按照模板编写，可参照如下模板进行编写。

```
# Global Section
plaintext =
//填充明文数据，实际使用时删掉此行

ciphertext =
//填充密文数据，实际使用时删掉此行

cipher_key =
//填充加密算法秘钥，实际使用时删掉此行

auth_key =
//填充认证算法秘钥，实际使用时删掉此行
```


版权所有 © 海思技术有限公司

```
cipher_iv =
//填充加密算法向量，实际使用时删掉此行

# sha_hmac_buff_32
digest =
//填充数据摘要结果，实际使用时删掉此行
```

须知

1. 自定义测试向量文件格式必须为：xxx.data。
2. 自定义测试向量文件与示例启动命令放置在同一文件夹下。
3. 自定义测试向量的明文（“plaintext”）长度、加密密钥（“cipher_key”）、加密初始化向量（“cipher_iv”）、数据摘要算法密钥（“auth_key”）、摘要（“digest”）长度，均不可小于启动命令对应的长度值。

自定义测试向量参考示例：基于上述模板使用aes-cbc加密算法，sha-hmac hash算法进行认证。编写自定义测试向量如下：

```
# Global Section
plaintext =
0xff, 0xca, 0xfb, 0xf1, 0x38, 0x20, 0x2f, 0x7b, 0x24, 0x98, 0x26, 0x7d, 0x1d, 0x9f, 0xb3, 0x93,
0xd9, 0xef, 0xbd, 0xad, 0x4e, 0x40, 0xbd, 0x60, 0xe9, 0x48, 0x59, 0x90, 0x67, 0xd7, 0x2b, 0x7b,
0x8a, 0xe0, 0x4d, 0xb0, 0x70, 0x38, 0xcc, 0x48, 0x61, 0x7d, 0xee, 0xd6, 0x35, 0x49, 0xae, 0xb4,
0xaf, 0x6b, 0xdd, 0xe6, 0x21, 0xc0, 0x60, 0xce, 0x0a, 0xf4, 0x1c, 0x2e, 0x1c, 0x8d, 0xe8, 0x7b

ciphertext =
0x75, 0x95, 0xb3, 0x48, 0x38, 0xf9, 0xe4, 0x88, 0xec, 0xf8, 0x3b, 0x09, 0x40, 0xd4, 0xd6, 0xea,
0xf1, 0x80, 0x6d, 0xfb, 0xba, 0x9e, 0xee, 0xac, 0x6a, 0xf9, 0x8f, 0xb6, 0xe1, 0xff, 0xea, 0x19,
0x17, 0xc2, 0x77, 0x8d, 0xc2, 0x8d, 0x6c, 0x89, 0xd1, 0x5f, 0xa6, 0xf3, 0x2c, 0xa7, 0x6a, 0x7f,
0x50, 0x1b, 0xc9, 0x4d, 0xb4, 0x36, 0x64, 0x6e, 0xa6, 0xd9, 0x39, 0x8b, 0xcf, 0x8e, 0x0c, 0x55

cipher_key =
0xE4, 0x23, 0x33, 0x8A, 0x35, 0x64, 0x61, 0xE2, 0x49, 0x03, 0xDD, 0xC6, 0xB8, 0xCA, 0x55, 0x7A

auth_key =
0xaf, 0x96, 0x42, 0xf1, 0x8c, 0x50, 0xdc, 0x67, 0x1a, 0x43, 0x47, 0x62, 0xc7, 0x04, 0xab, 0x05,
0xf5, 0x0c, 0xe7, 0xa2, 0xa6, 0x23, 0xd5, 0x3d, 0x95, 0xd8, 0xcd, 0x86, 0x79, 0xf5, 0x01, 0x47,
0x4f, 0xf9, 0x1d, 0x9d, 0x36, 0xf7, 0x68, 0x1a, 0x64, 0x44, 0x58, 0x5d, 0xe5, 0x81, 0x15, 0x2a,
0x41, 0xe4, 0x0e, 0xaa, 0x1f, 0x04, 0x21, 0xff, 0x2c, 0xf3, 0x73, 0x2b, 0x48, 0x1e, 0xd2, 0xf7

cipher_iv =
0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F

# sha_hmac_buff_32
digest =
0x4d, 0xb9, 0xf3, 0xc6, 0x0a, 0xd8, 0x13, 0xf9, 0x22, 0xc7, 0x61, 0x53, 0x72, 0xa6, 0xee, 0xda,
0x69, 0x11, 0x90, 0x7f
```

须知

自定义测试向量参考示例中摘要结果为对明文（“plaintext”）的数据摘要计算结果。

5.2.4.1.3 示例启动

加密示例启动命令样例（加密算法使用SM4-CBC算法）：

版权所有 © 海思技术有限公司

```
dpdk-test-crypto-perf -l 6-7 -- --ptest throughput --devtype udrv_sec_drv --optype cipher-only --cipher-algo sm4-cbc --cipher-op encrypt --cipher-key-sz 16 --cipher-iv-sz 16 --total-ops 16000 --burst-sz 32 --buffer-sz 512
```

解密示例启动命令样例（解密算法使用SM4-CBC算法）：

```
dpdk-test-crypto-perf -l 6-7 -- --ptest throughput --devtype udrv_sec_drv --optype cipher-only --cipher-algo sm4-cbc --cipher-op decrypt --cipher-key-sz 16 --cipher-iv-sz 16 --total-ops 16000 --burst-sz 32 --buffer-sz 512
```

认证示例启动命令样例（数据摘要算法使用SM3-HMAC算法）：

```
dpdk-test-crypto-perf -l 6-7 -- --ptest throughput --devtype udrv_sec_drv --optype auth-only --auth-algo sm3-hmac --auth-op generate --auth-key-sz 64 --digest 32 --total-ops 16000 --burst-sz 32 --buffer-sz 512
```

先加密后认证示例启动命令样例（加密算法使用SM4-CBC算法，数据摘要算法使用SM3-HMAC算法）：

```
dpdk-test-crypto-perf -l 6-7 -- --ptest throughput --devtype udrv_sec_drv --optype cipher-then-auth --cipher-algo sm4-cbc --cipher-op encrypt --cipher-key-sz 16 --cipher-iv-sz 16 --auth-algo sm3-hmac --auth-op generate --auth-key-sz 64 --digest 32 --total-ops 16000 --burst-sz 32 --buffer-sz 512
```

先认证后解密示例启动命令样例（数据摘要算法使用AES-CBC算法，解密算法使用SHA1-HMAC算法）：

```
dpdk-test-crypto-perf -l 6-7 -- --ptest throughput --devtype udrv_sec_drv --optype auth-then-cipher --cipher-algo aes-cbc --cipher-op decrypt --cipher-key-sz 16 --cipher-iv-sz 16 --auth-algo sha1-hmac --auth-op verify --auth-key-sz 64 --digest 20  --total-ops 16000 --burst-sz 32 --buffer-sz 64 --test-file aes_cbc_sha1_hmac.data
```

表 5-6 启动参数解析

配置项	含义	取值范围	备注
-l	测试使用核心数	0,1,2,3,4,5,6,7	核心数，默认参数的第一个核心作为主核，不参与加密性能测试。如：-l 6-7，仅7核参与加解密性能统计。
--ptest	性能测试类型	throughput，latency	-
--devtype	加解密设备名称	udrv_sec_drv	Hi309a SEC硬核加密设备名称。
--optype	操作类型	cipher-only，auth-only，cipher-then-auth，auth-then-cipher	-
--cipher-algo	加解密算法	sm4-cbc，aes-cbc，aes-ctr，aes-ecb，3des-cbc，3des-ecb，des-cbc	-
--cipher-op	加解密操作类型	encrypt，decrypt	-

版权所有 © 海思技术有限公司

配置项	含义	取值范围	备注
--cipher-key-sz	加解密秘钥长度	非任意值，受算法类型约束，不同算法存在差异。	非任意值，受算法类型约束。参考表4-1填写。
--cipher-iv-sz	加解密初始化向量	非任意值，受算法类型约束，不同算法存在差异。	非任意值，受算法类型约束。参考表4-1填写。
--auth-algo	认证算法	aes-cmac，aes-xcbc-mac，md5，md5-hmac， sha1，sha1-hmac，sha2-224，sha2-224-hmac，sha2-256，sha2-256-hmac，sha2-384，sha2-384-hmac，sha2-512，sha2-512-hmac，sm3-hmac	-
--auth-op	认证操作	generate，verify	-
--auth-key-sz	认证算法秘钥长度	非任意值，受算法类型约束，不同算法存在差异。	非任意值，受算法类型约束。参考表4-1填写。
--digest	认证算法摘要长度	非任意值，受算法类型约束，不同算法存在差异。	非任意值，受算法类型约束。参考表4-1填写。
--burst-sz	一个测试周期处理包的数量	16，24，32，推荐配置32。	影响测试过程中内存申请，配置过大会导致申请内存失败。影响测试性能。
--buffer-sz	单个数据包的大小	32，64，128，...，上限值不要超过1520。 支持以太网帧的所有长度测试。	影响测试过程中内存申请，配置过大会导致申请内存失败。 aes-cbc、aes-ecb算法加密是buffer-sz必须16Byte对齐； 3des-cbc、3des-ecb、des-cbc算法加密是buffer-sz必须8Byte对齐；
--total-ops	执行的总操作次数	推荐配置16000。	推荐按照burst-sz的倍数配置。较少的操作数会影响测试性能。
--test-file	自定义测试向量文件	根据模板编写	文件格式必须为：xxx.data，文件路径放置执行命令的文件中。

版权所有 © 海思技术有限公司

5.2.4.1.4 注意事项

须知

1. 进行加密操作时，初始化向量（IV）存储地址必须小于明文存储地址。
2. 进行先加密后认证操作时，初始化向量（IV）存储地址必须小于明文存储地址，明文存储地址必须小于摘要存储地址。
3. 进行时延测试时，将启动命令中“--ptest throughput”替换为“--ptest latency”。
4. “--total-ops”操作次数不可过大，会导致内存不够引发崩溃，崩溃后需重新升级固件。

5.2.5 非对称加解密示例使用说明

5.2.5.1 test/test_cryptodev_asym 示例使用说明

5.2.5.1.1 示例启动

DPDK测试自动化程序启动命令示例

```
dpdk-test --vdev crypto_openssl --socket-mem 1024
```

执行OpenSSL非对称加解密自动化测试程序

```
cryptodev_openssl_asym_autotest
```

退出应用程序

```
quit
```

5.2.5.1.2 注意事项

须知

1. Hi309a SDK提供的OpenSSL硬件引擎内部使用系统大页内存进行数据存储，通过dpdk eal参数“--socket-mem”控制DPDK管理的大页内存空间，最少给硬件引擎预留256MByte空闲大页内存。通常系统预留2GByte大页内存，通过“--socket-mem”控制DPDK管理的大页内存空间，推荐配置1024、1280、1536、1792，单位：MByte。
2. 硬件引擎加载失败可能依赖的动态库无法找到，可通过系统中配置动态库路径尝试解决。
   ```
   export LD_PRELOAD="/lib64/ld-linux-aarch64.so.1 /lib64/libc.so.6 /lib64/libm.so.6"
   ```
3. 进行性能测试时，需要使能测试程序中的性能测试宏开关，配置ASYM_PERF_TEST为1，路径：“open_source/dpdk/app/test/test_cryptodev_asym.c”。
4. 进行性能测试时，“--socket-mem”配置为1024。

5.3 多进程示例使用说明

版权所有 © 海思技术有限公司

5.3.1 DTS 配置

DPDK适配的DTS文件配置中默认关闭RSS功能，此时每个网口只有一个接收队列rx0与一个发送队列tx0。simple-mp示例可选择开启RSS或关闭RSS功能，symmetirc-mp需要打开RSS功能，否则除主进程外，其余进程无队列可用。

5.3.2 DPDK 编译

版本编译请参考《 Hi309a 系统编译运行指南 》“统一镜像构建”章节。simple-mp、symmetric-mp应用已集成到系统。

5.3.3 镜像升级与运行环境

前提条件：根据所需按照《 Hi309a 系统编译运行指南 》“统一镜像构建”章节说明编译对应设备树、系统镜像、文件系统。

步骤1 openEuler系统升级完成。

步骤2 进入openEuler系统后，执行如下命令：

```
cd /lib/udrivers
insmod motorcomm.ko
insmod hns_mdio.ko
insmod hi309a_urm.ko
insmod hi309a_urm_adapt.ko
insmod hi309a_poe.ko
insmod hi309a_ppe.ko
insmod hi309a_mag.ko
insmod igb_uio.ko
insmod rte_kni.ko
insmod hi309a_kdrv_dfx.ko
insmod dpdk_phy.ko

mkdir -p /dev/hugepages
mountpoint -q /dev/hugepages || mount -t hugetlbfs nodev /dev/hugepages
echo 1024 > /proc/sys/vm/nr_hugepages
```

----结束

5.3.4 simple-mp

5.3.4.1 simple-mp 示例启动

步骤1 启动主进程。

```
dpdk-simple_mp --proc-type=primary -l 0-1 -n 1 --legacy-mem --file-prefix=simple_mp
```

步骤2 启动从进程。

```
dpdk-simple_mp --proc-type=secondary -l 2-3 -n 1 --legacy-mem --file-prefix=simple_mp
```

----结束

版权所有 © 海思技术有限公司

说明

参数说明：

-l：2,3 绑定在2 和 3号核上。

-n：每个CPU的内存通道数。

--legacy-mem：固定内存池，启动时一次性分配，性能更可预测，但内存一旦分配则不可减少。

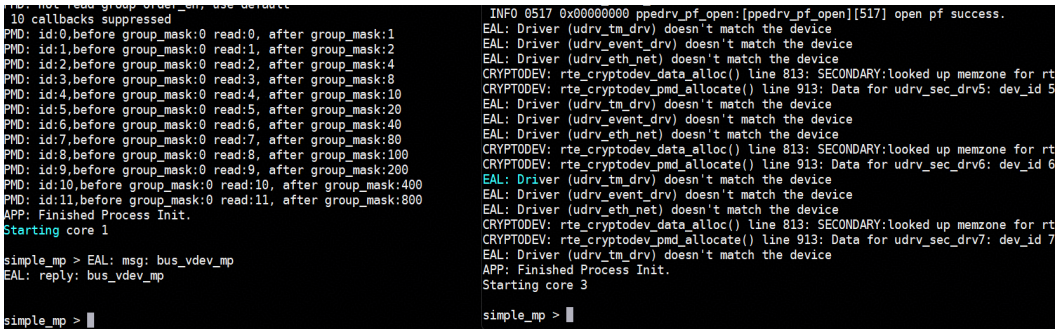
--proc-type：指定运行的进程类型，分为primary与secondary两种，如果传入auto，将有程序自动判断进程类型。

--file-prefix：用于指定当前 DPDK 进程的共享内存文件前缀。不使用此参数，共享内存文件前缀将使用代码给定的默认值。

5.3.4.2 simple-mp 示例使用

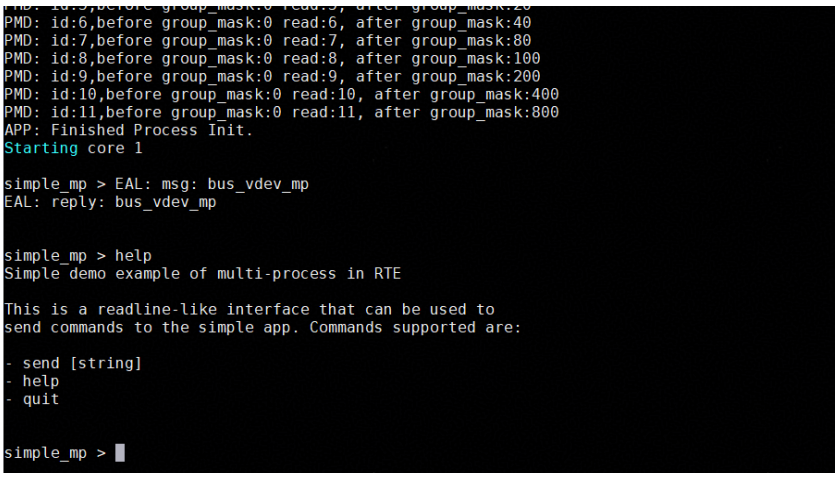
主从进程启动后示意图如图5-8所示，左边为主进程，右边为从进程。

图 5-8 主进程与从进程启动



启动后可通过 help 命令查看支持的操作，simple-mp支持 send 发送消息命令 与 quit 退出命令。

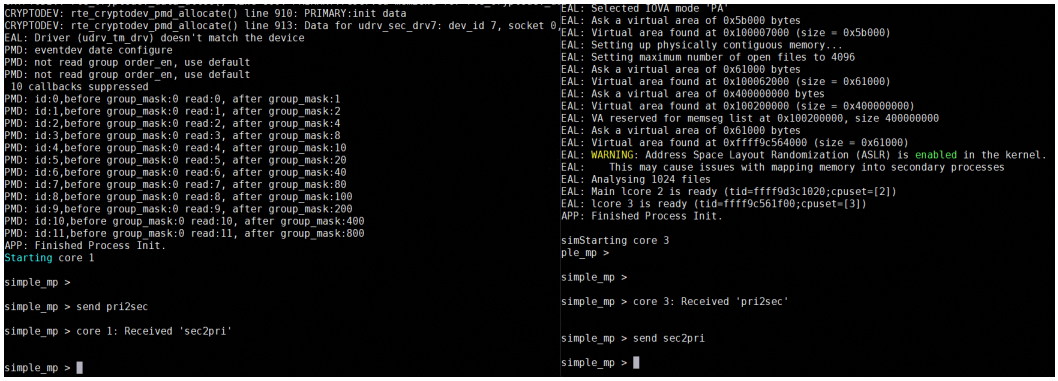
图 5-9 支持指令查看操作



使用send 命令发送消息，主进程发送消息，从进程接收消息，从进程发送消息，主进程接收消息。

版权所有 © 海思技术有限公司

图 5-10 主从进程交互消息



须知

1. 只能启动一个主进程，可以启动多个从进程，且多个从进程发送的消息主进程均能接收，但主进程发送的消息只会被一个从进程接收（最后启动的从进程）。
2. simple-mp示例的一个进程需要绑定两个核心，一个核心作为主核心执行命令行程序，另一个核心执行消息收发工作。如果只绑定一个核心，则仅有命令行，无法收发消息。

5.3.5 symmetric-mp

5.3.5.1 symmetric-mp 示例启动

步骤1 启动主进程。

```
dpdk-symmetric_mp -l 0-3 -n 1 --proc-type=primary --file-prefix=symmetric_mp --legacy-mem -- -p 3 --num-procs=8 --proc-id=0
```

步骤2 启动从进程。

```
dpdk-symmetric_mp -l 4-7  -n 1 --proc-type=secondary --file-prefix=symmetric_mp --legacy-mem -- -p 3 --num-procs=8 --proc-id=1
```

----结束

版权所有 © 海思技术有限公司

说明

参数说明：

-l：0-3 绑定在0，1，2，3号核上。

-n：每个CPU的内存通道数。

-p：使用的端口的掩码: 0x1 表示2的0次方使用的是port 0 。

使用的是port 1是2的一次方，也就是2 即0x2。

使用的是port 2 ，就是2的2次方，也就是4，即0x4。

使用的是port 3 ，就是2的3次方，也就是8，即0x8。

则此时-p 的值是，2的0次方+ 2的1次方，也就是3，即此时参数为： -p 0x3。

传入的端口需要成对，也即使用端口数量为2的倍数。

--legacy-mem：固定内存池，启动时一次性分配，性能更可预测，但内存一旦分配则不可减少。

--proc-type：指定运行的进程类型，分为primary与secondary两种，如果传入auto，将由程序自动判断进程类型。

--file-prefix：用于指定当前 DPDK 进程的共享内存文件前缀。不使用此参数，共享内存文件前缀将使用代码给定的默认值。

--num-procs：指定总进程数。RSS模式下固定为8。

--proc-id：指定当前进程 ID。RSS模式不实际生效。

须知

1. symmetric-mp可启动多个从进程，仅在开启RSS模式下从进程有效。非RSS模式下每个端口仅一个收发队列，因此主进程启动后，从进程将无队列可用。
2. symmetric-mp在RSS模式启动多个从进程，每一个核心绑定一个队列，绑定对应的核心即绑定对应的队列（如绑定0-3核心，也即绑定了0-3号队列）。
3. 所有进程的绑定核心数必须与RSS队列数相等。如启动一个主进程，一个从进程，则主进程绑定核心数+从进程绑定核心数 = 总的核心数 = 总的RSS队列数。
4. 能够开启的最大进程数据量 = 核心数量 = RSS队列数量。如：8个核心 <--> 8个队列 <-->最多8个进程（1主 + 7从，每一个进程绑定一个核心）。

5.3.5.2 symmetric-mp 示例使用

启动示例后，可通过打流测试来验证symmetric-mp示例数据包转发功能。此处使用Hi309a demo板，以打流仪测试为例，port0指定为单板的xge1口，port2指定为单板的xge2口。

步骤1 按**打流仪配置**章节配置打流仪，可按照L2FWD的转发数据流配置。

步骤2 将打流仪的10Gbps光口分别与单板的xge1和xge2口相连。

步骤3 按**打流仪配置**章节配置使用RFC2544测试套件进行网络吞吐量测试。

----结束

版权所有 © 海思技术有限公司

须知

1. symmetric-mp示例的默认功能是进行数据包转发，如进程1使用了0-3号队列，则进程1会接收0-3号队列的所有数据包，并将其通过发送队列转发至目的端口。
2. 源端口与目的端口指定为：0 <-> 1, 2 <-> 3 etc。在RSS模式下，只使用了port0与port1，此时port0与port1互为源端口与目的端口。可通过dts配置文件指定port0与port1的实际物理端口。
3. 由于symmetric-mp示例在退出时主进程会清理资源，故而退出程序时必须先关闭所有从进程，最后再关闭主进程，否则会造成系统崩溃。

版权所有 © 海思技术有限公司

6 性能测试结果

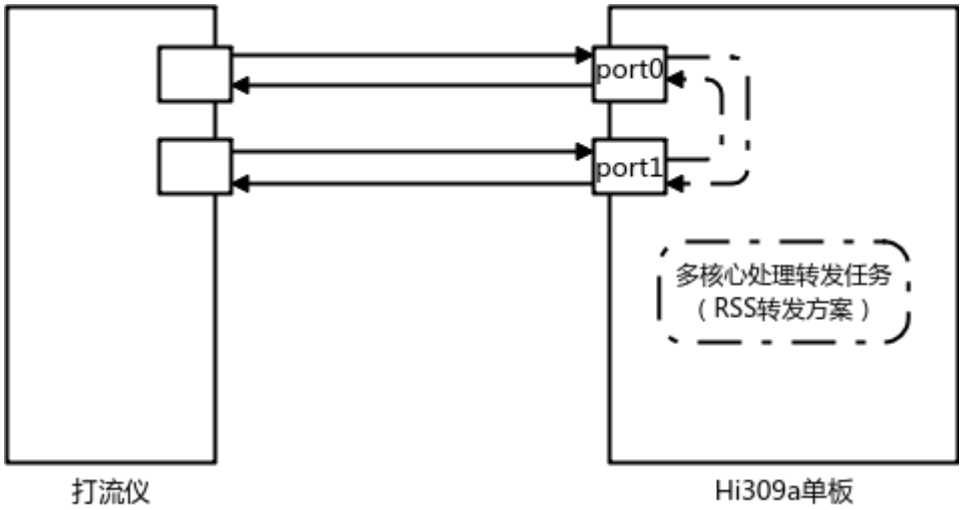
6.1 转发性能测试结果

在进行转发性能测试时，使用RSS方案才可获得Hi309a DPDK最大转发性能。转发性能与测试流的配置相关，测试流的原则：尽可能使测试流均匀分配到每个核上，以获取最大转发性能。

6.1.1 测试环境与拓扑

使用m900单板、demo单板与Tesigine对接测试，测试使用RFC2544测试套进行测试。

图 6-1 309a DPDK L2、L3 层 RSS 转发方案硬件拓扑示意



须知

RSS转发方案中port0、port1对应为DTS中dpdketh@0、dpdketh@1中对应的物理端口。prot0、port1与硬件单板网口的映射关系需查看原理图确认。

版权所有 © 海思技术有限公司

6.1.2 打流仪配置

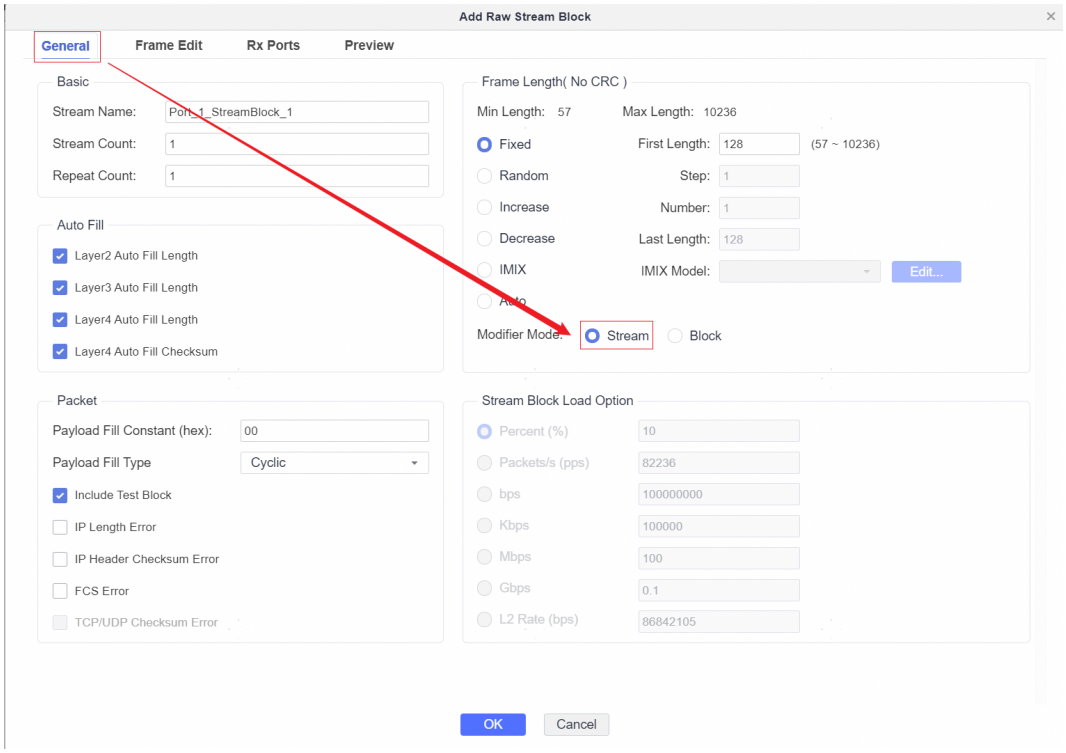
打流仪使用Tesgine，硬件版本BIOS_VER 1.55_7rate，软件版本：TesgineV700R003C30550。

步骤1 配置流信息。

流配置按照如下关键点进行配置：

- 报文采用UDP报文。
- 配置源IP和目的IP。示例中IP与端口绑定关系是IP为198.18.0.xx的报文从port0发送，IP为198.18.1.xx的报文从port1发送。port0对接的打流仪网口的流配置为源IP 198.18.0.xx，目的IP配置为198.18.1.xx；port1对接的打流仪网口的流配置为源IP 198.18.1.xx，目的IP配置为198.18.0.xx。
- 配置流支持散列到多队列。流与核心的对应关系与四元组{port_src, port_dest, ip_src, ip_dest}数值有关，可以通过变更port_src或ip_src，使得流均匀的散列到多个队列上，让多个核心参与转发处理。

图 6-2 配置测试使用流模式



版权所有 © 海思技术有限公司

图 6-3 配置报文为 UDP 格式

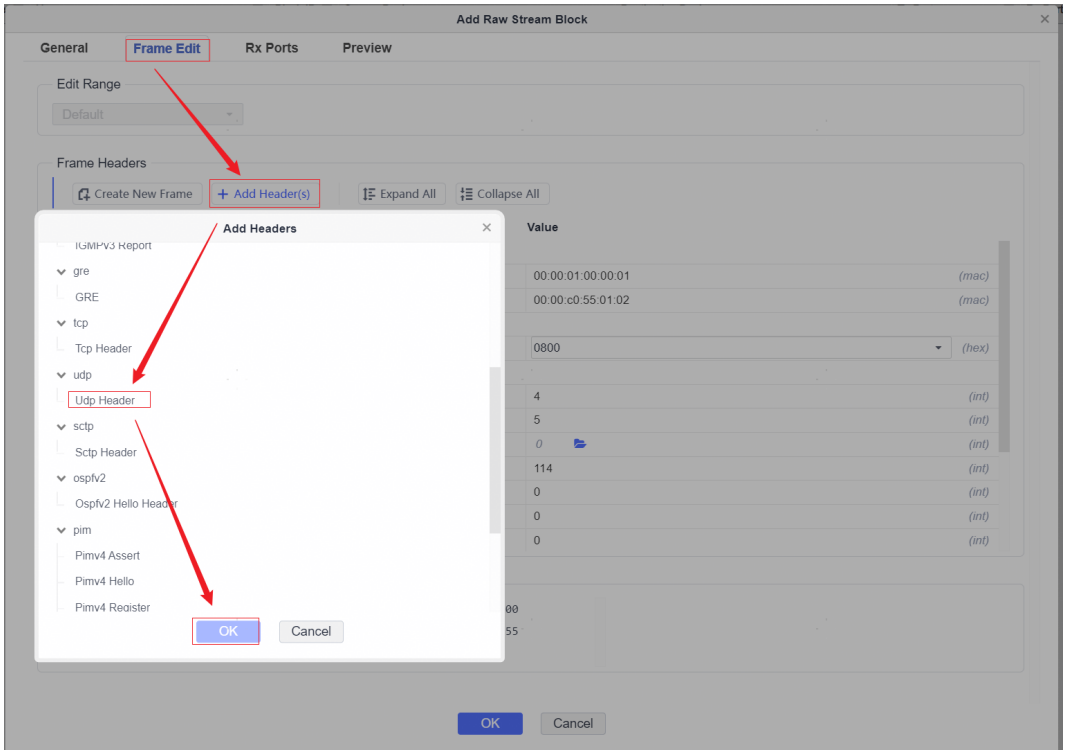
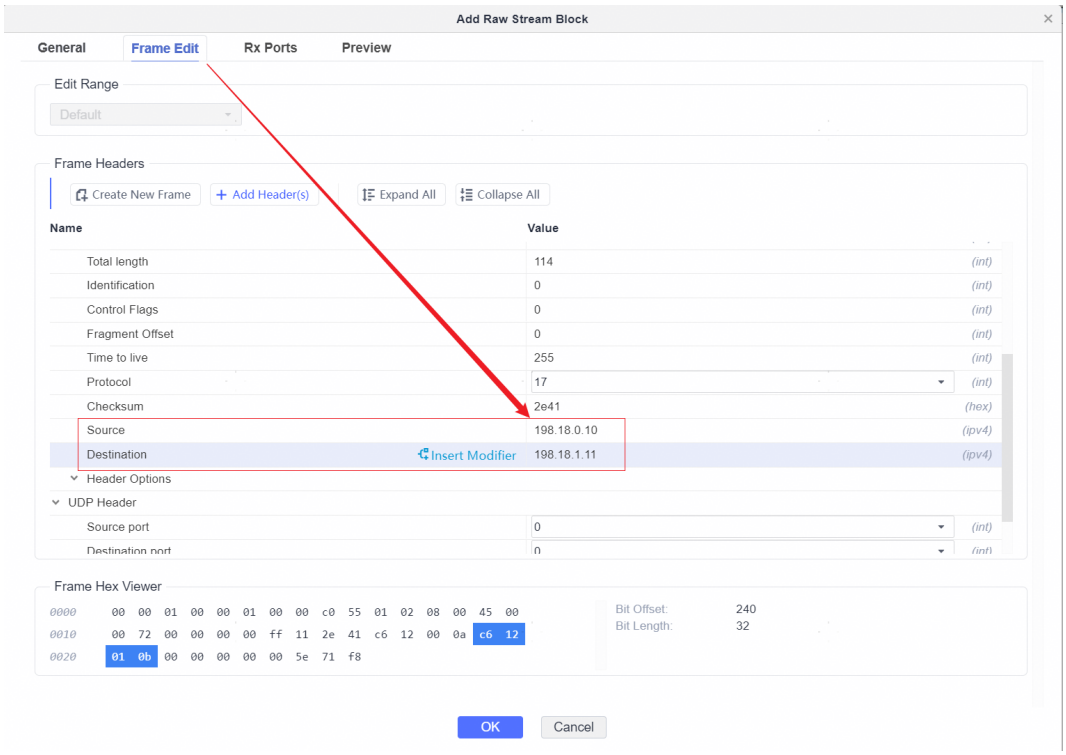
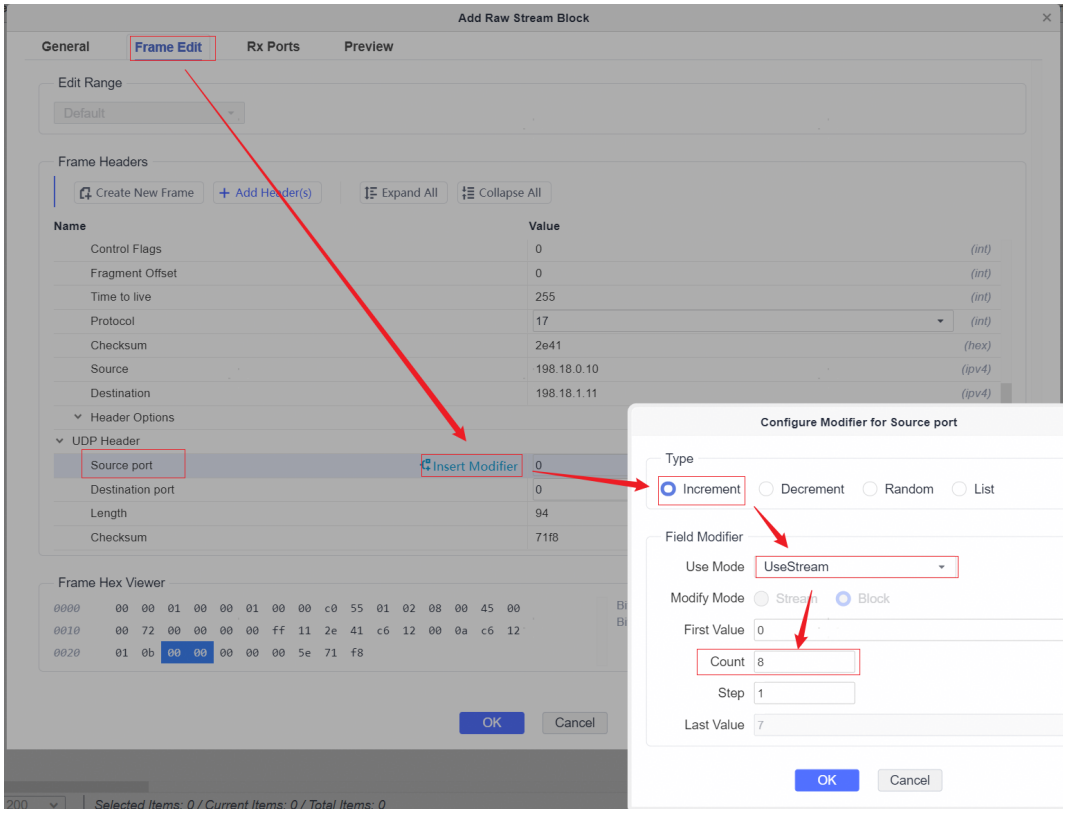


图 6-4 配置报文的源 ip 和目的 ip



版权所有 © 海思技术有限公司

图 6-5 配置流支持散列到多队列



须知

1. 流配置中目的IP和源IP仅dpdk L3层转发必须配置，L2层转发无需配置IP信息。
2. 示例中IP与端口绑定关系是IP为198.18.0.xx的报文从port0发送，IP为198.18.1.xx的报文从port1发送。port0对接的打流仪网口的流配置为源IP 198.18.0.xx，目的IP配置为198.18.1.xx；port1对接的打流仪网口的流配置为源IP 198.18.1.xx，目的IP配置为198.18.0.xx。
3. DPDK L3层转发路由算法使用LPM算法。路由表使用DPDK示例中默认路由表：

   三元组{IP地址，匹配位数，Port}

   {RTE_IPV4(198, 18, 0, 0), 24, 0},

   {RTE_IPV4(198, 18, 1, 0), 24, 1}。
4. DPDK L2层、L3层转发进行性能测试时均使用RSS方案，测试流需要均匀散列到不同队列上，让多核心进行转发处理，如果固定报文测试，性能与单核一致。

步骤2 配置RFC2544测试套。

版权所有 © 海思技术有限公司

图 6-6 使用吞吐量测试模式

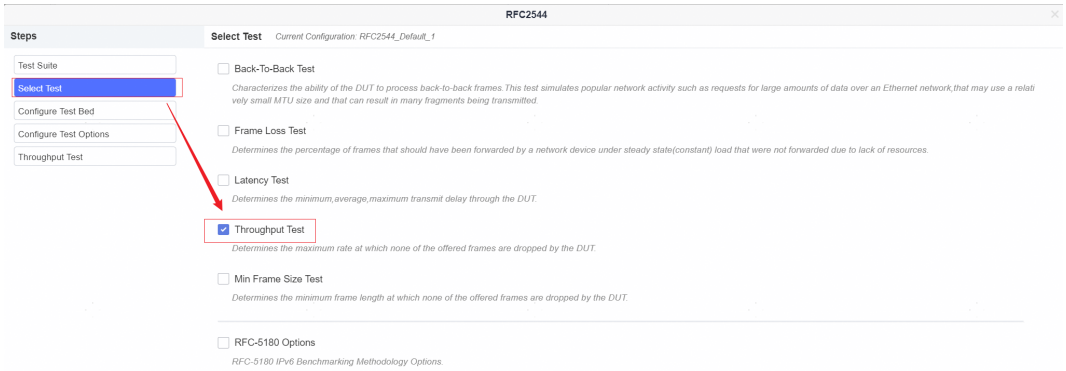
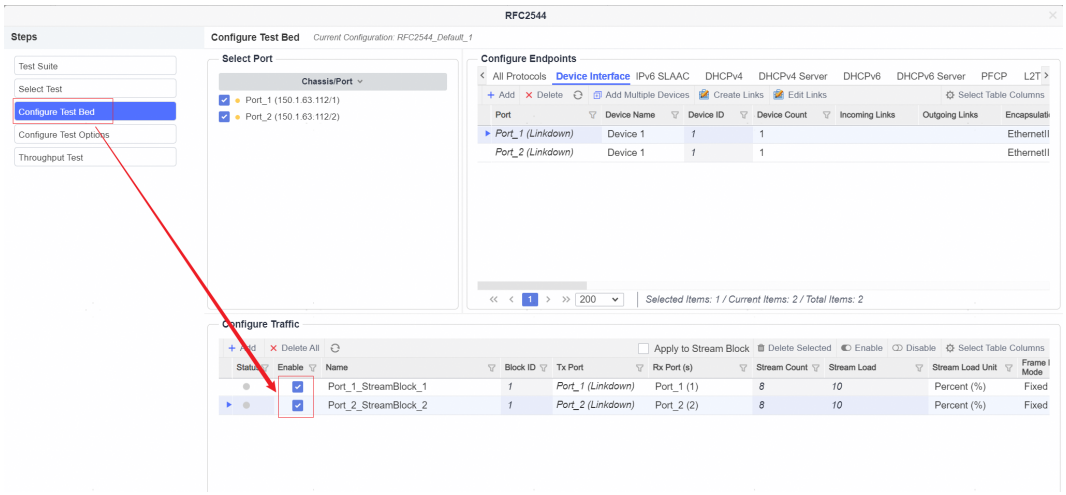
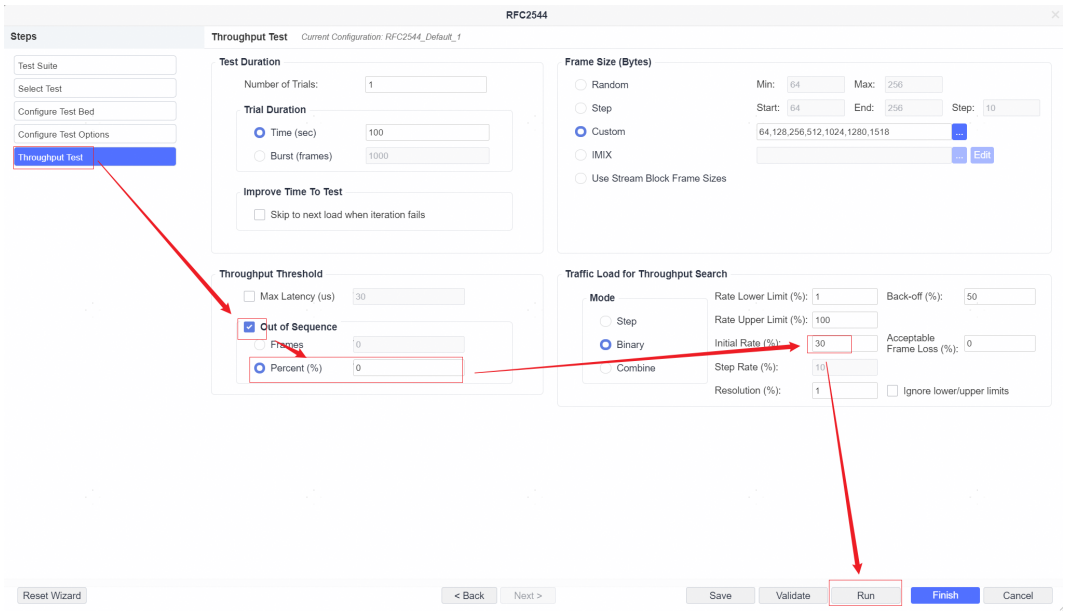


图 6-7 测试套使用步骤 1 配置的流进行测试



版权所有 © 海思技术有限公司

图 6-8 配置检测乱序（Out of Sequence）、初始速度（Initial Rate），启动性能测试



----结束

6.1.3 测试结果

表 6-1 XGE 网口双向打流性能测试结果

转发层次	核心数	网口类型	报文长度（单位：Byte）	udp报文最大不丢包吞吐量（单位：%）	udp报文最大不丢包吞吐量（单位：Mbps）	接收最大延时（单位：uSec）	备注
L2FWD	8	XGE	64	44.21875	8843.75	277.86	测试套输出的流在多核分布可能不均，导致结果存在差异。性能测试结果>=40%，44%为测试典型值。
			128	76.484375	15296.875	300.316	测试套输出的流在多核分布可能不均，导致结果存在差异。性能测试结果>=73%，76%为测试典型值。
			256	99.453125	19890.625	195.812	-
			512	99.453125	19890.625	193.344	-

版权所有 © 海思技术有限公司

转发层次	核心数	网口类型	报文长度（单位：Byte）	udp报文最大不丢包吞吐量（单位：%）	udp报文最大不丢包吞吐量（单位：Mbps）	接收最大延时（单位：uSec）	备注
			1024	99.453125	19890.625	192.264	-
			1280	99.453125	19890.625	194.488	-
			1518	99.453125	19890.625	189.28	-
L3FWD	8	XGE	64	44.765625	8953.125	355.692	测试套输出的流在多核分布可能不均，导致结果存在差异。性能测试结果>=40%，44%为测试典型值。
			128	78.671875	15734.375	310.412	测试套输出的流在多核分布可能不均，导致结果存在差异。性能测试结果>=73%，78%为测试典型值。
			256	99.453125	19890.625	348.192	-
			512	99.453125	19890.625	283.42	-
			1024	99.453125	19890.625	263.664	-
			1280 1518	99.453125 99.453125	19890.625 19890.625	268.776 243.232	-
symmetric-mp	8	XGE	64	43.109375	8621.875	1983.18	测试套输出的流在多核分布可能不均，导致结果存在差异。性能测试结果>=40%，44%为测试典型值。
			128	77.34375	15468.75	247.92	测试套输出的流在多核分布可能不均，导致结果存在差异。性能测试结果>=73%，78%为测试典型值。

版权所有 © 海思技术有限公司

转发层次	核心数	网口类型	报文长度（单位：Byte）	udp报文最大不丢包吞吐量（单位：%）	udp报文最大不丢包吞吐量（单位：Mbps）	接收最大延时（单位：uSec）	备注
			256	97.65625	19531.25	229.272	-
			512	97.65625	19531.25	1979.596	-
			1024	97.65625	19843.75	1975.868	-
			1280 1518	19843.75 19843.75	19890.625 19890.625	1975.776 1975.776	-

须知

1. 吞吐量性能参数demo、m900单板均能达到表中描述参数。
2. 接收最大延时每次测试可能存在波动。
3. 对于小包数据64Byte、128Byte，测试套输出的流在多核分布可能不均，导致性能结果存在差异。

6.2 对称加解密性能测试参考结果

6.2.1 测试环境

无需对接外部环境，仅单板即可执行。

6.2.2 测试结果

表 6-2 DPDK 下 Hi309a SEC 模块加密性能参考结果

算法名称	key_size（单位：Byte）	iv_size（单位：Byte）	测试包长			备注
			64Byte（单位：Gbps）	512Byte（单位：Gbps）	1518Byte（单位：Gbps）	
AES_CBC	16	16	1.4551	8.5465	19.4954	buffer-sz必须16Byte对齐，大包使用1520Byte代替1518Byte测试。
AES_CTR	16	16	1.4581	8.5475	19.3799	-
AES_ECB	16	0	1.4523	8.5429	19.5797	buffer-sz必须16Byte对齐，大包使用1520Byte代替1518Byte测试。

版权所有 © 海思技术有限公司

算法名称	key_size（单位：Byte）	iv_size（单位：Byte）	测试包长			备注
			64Byte（单位：Gbps）	512Byte（单位：Gbps）	1518Byte（单位：Gbps）	
DES_CBC	16	8	1.4511	8.5480	16.7771	buffer-sz必须16Byte对齐，大包使用1520Byte代替1518Byte测试。
3DES_CBC	16	8	1.4762	6.0360	6.1085	buffer-sz必须16Byte对齐，大包使用1520Byte代替1518Byte测试。
3DES_ECB	16	0	1.4604	6.0387	6.1089	buffer-sz必须16Byte对齐，大包使用1520Byte代替1518Byte测试。
SM4_CBC	16	16	1.4408	8.4315	9.2332	-

表 6-3 DPDK 下 Hi309a SEC 模块认证性能参考结果

算法名称	key_size（单位：Byte）	digest_size（单位：Byte）	测试包长			备注
			64Byte（单位：Gbps）	512Byte（单位：Gbps）	1518Byte（单位：Gbps）	
AES_CMAC	16	12	1.4989	8.8716	19.1429	-
AES_XCBC_MAC	16	12	1.5006	8.8869	19.0587	-
MD5	0	12	1.4989	8.8936	16.9962	-
MD5_HMAC	64	12	1.5016	8.6776	15.0089	-
SHA1	0	20	1.5006	8.8940	19.2189	-
SHA1_HMAC	64	20	1.4956	8.8349	19.1104	-
SHA224	0	12	1.5006	8.8811	19.1751	-
SHA224_HMAC	64	12	1.5022	8.8775	18.9913	-
SHA256	0	12	1.5001	8.8940	19.0964	-
SHA256_HMAC	64	32	1.5004	8.8847	19.0072	-
SHA384	0	12	1.5202	8.8805	19.1692	-
SHA384_HMAC	64	12	1.5025	8.8317	19.1815	-
SHA512	0	12	1.5022	8.8404	19.1192	-

版权所有 © 海思技术有限公司

算法名称	key_size（单位：Byte）	digest_size（单位：Byte）	测试包长			备注
			64Byte（单位：Gbps）	512Byte（单位：Gbps）	1518Byte（单位：Gbps）	
SHA512_HMAC	64	12	1.4883	8.8928	19.1373	-
SM3_HMAC	64	32	1.5037	8.8775	19.0134	-

表 6-4 DPDK 下 Hi309a SEC 模块先加密后认证性能参考结果

加密算法	认证算法	cipher_key_size（单位：Byte）	cipher_iv_size（单位：Byte）	auth_key_size（单位：Byte）	auth_digest_size（单位：Byte）	测试包长			备注
						64Byte（单位：Gbps）	512Byte（单位：Gbps）	1518Byte（单位：Gbps）	
AES_CBC	SHA1_HMAC	16	16	64	20	1.3933	8.3317	19.1236	buffer-sz必须8Byte对齐，大包使用1520Byte代替1518Byte测试。
AES_CBC	SHA256_HMAC	16	16	64	32	1.3945	8.3342	18.4079	buffer-sz必须8Byte对齐，大包使用1520Byte代替1518Byte测试。
DES_CBC	SHA1_HMAC	16	8	64	20	1.3933	8.3356	16.7217	buffer-sz必须8Byte对齐，大包使用1520Byte代替1518Byte测试。

版权所有 © 海思技术有限公司

加密算法	认证算法	cipher_key_size（单位：Byte）	cipher_iv_size（单位：Byte）	auth_key_size（单位：Byte）	auth_digest_size（单位：Byte）	测试包长			备注
						64Byte（单位：Gbps）	512Byte（单位：Gbps）	1518Byte（单位：Gbps）	
DES_CBC	SHA256_HMAC	16	8	64	32	1.3942	8.3026	16.7283	buffer-sz必须8Byte对齐，大包使用1520Byte代替1518Byte测试。
3DES_CBC	SHA1_HMAC	16	8	64	20	1.4017	6.0159	6.1062	buffer-sz必须16Byte对齐，大包使用1520Byte代替1518Byte测试。
3DES_CBC	SHA256_HMAC	16	8	64	32	1.3934	6.0380	6.1065	buffer-sz必须16Byte对齐，大包使用1520Byte代替1518Byte测试。
SM4_CBC	SM3_HMAC	16	16	64	32	1.3945	8.2994	9.2006	-

版权所有 © 海思技术有限公司

须知

1. 加解密性能、认证性能、先加密后认证性能是在“--total-ops 16000”、“ --burst-sz 32”参数下测试，调整这两个参数性能会有增强或衰减变化。
2. 相同命令下每次测试性能数据会有细微波动。
3. 测试大包时，操作次数不可过大，会导致内存不够引发崩溃，崩溃后需重新升级固件。
4. 解密方向性能与加密方向性能一致，认证（验证）方向性能与认证（生成）方向性能一致，先认证后解密方向性能与先加密后认证方向性能一致。

6.3 非对称加解密性能测试参考结果

6.3.1 测试环境

无需对接外部环境，仅单板即可执行。

6.3.2 测试结果

表 6-5 非对称加解密性能测试参考结果

算法	OpenSSL非对称加解密309a硬件加速支持接口	秘钥长度/椭圆曲线	调用DPDK接口性能测试（OpenSSL纯软性能）	调用DPDK接口性能测试（OpenSSL硬件加速）	调用OpenSSL接口性能测试（纯软性能）	调用OpenSSL接口性能测试（硬件加速）
DH	DH_generate_key()	1024 bit	0.42 Kops/s	2.65 Kops/s	0.4347 Kops/s	2.788 Kops/s
		2048 bit	-	-	0.4652 Kops/s	0.7223 Kops/s
	DH_compute_key()	1024 bit	0.43 Kops/s	2.72 Kops/s	0.4341 Kops/s	2.7823 Kops/s
		2048 bit	-	-	0.4641 Kops/s	0.7223 Kops/s
RSA	RSA_public_encrypt	1024 bit	21.26 Kops/	39.13 Kops/s	23.4392 Kops/s	50.3173 Kops/s
		2048 bit	-	-	7.3593 Kops/s	32.7243 Kops/s
	RSA_private_decrypt	1024 bit	1.37 Kops/s	8.48 Kops/s	1.37 Kops/s	8.5979 Kops/s
		2048 bit	-	-	0.2066 Kops/s	2.2500 Kops/s
	RSA_private_encrypt	1024 bit	1.38 Kops/s	8.80 Kops/s	1.3801 Kops/s	8.9516 Kops/s
		2048 bit	-	-	0.207 Kops/s	2.4469 Kops/s
	RSA_public_decrypt	1024 bit	17.21 Kops/s	38.53 Kops/s	27.3517 Kops/s	71.7462 Kops/s

版权所有 © 海思技术有限公司

算法	OpenSSL非对称加解密309a硬件加速支持接口	秘钥长度/椭圆曲线	调用DPDK接口性能测试（OpenSSL纯软性能）	调用DPDK接口性能测试（OpenSSL硬件加速）	调用OpenSSL接口性能测试（纯软性能）	调用OpenSSL接口性能测试（硬件加速）
		2048 bit	-	-	7.688 Kops/s	39.2528 Kops/s
ECDH	EC_KEY_generate_key	secp256k1	OpenSSL-pmd驱动不支持		0.6 Kops/s	3.6931 Kops/s
	ECDH_compute_key	secp256k1	OpenSSL-pmd驱动不支持		0.602 Kops/s	3.8331 Kops/s
ECDSA	ECDSA_sign	secp256k1	OpenSSL-pmd驱动不支持		0.5706 Kops/s	3.6327 Kops/s
	ECDSA_verify	secp256k1	OpenSSL-pmd驱动不支持		0.6805 Kops/s	1.8551 Kops/s

须知

1. 非对称加解密调用dpdk接口性能结果基于dpdk自动化测试程序“test/test_cryptodev_asym.c”获得。
2. 进行性能测试时，需要使能测试程序中的性能测试宏开关，配置“ASYM_PERF_TEST”为1，路径：“open_source/dpdk/app/test/test_cryptodev_asym.c”。
3. 进行性能测试时，“--socket-mem”配置为1024。
4. 相同命令下每次测试性能数据会有细微波动。

版权所有 © 海思技术有限公司