VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

IP2 – Technická zpráva

Diagnostický nástroj pro síťová zařízení

16.04.2018 Filip Kočica <xkocic01@stud.fit.vutbr.cz>

Abstrakt

Práce se zabývá návrhem, implementací a testováním algoritmu pro zpracování paketů na vysokorychlostní počítačové síti. Hlavním cílem bylo vytvoření funkční diagnostické aplikace sloužící pro přesnou rekonstrukci části síťové komunikace, jež umožní zpracování složených paketů a následné seřazení a ukládání zpracovaných paketů na plné rychlosti linek 40Gb/s platformy LS2088A, aniž by se nějaké pakety zahazovaly. Aplikace je založena na zpracování složených paketů, místo obyčejných, zejména kvůli zvýšení propustnosti. Procesor LS2088A obsahuje hardwareovou akcelerační jednotku DPAA2, která je velmi komplexní a celkově tak dosahuje řádově tisíce možných konfigurací. Součástí této práce bylo tedy i nalezení vhodné konfigurace platformy. Funkčních konfigurací je spousta, ale valná většina z nich má velice špatnou výkonnostní charakteristiku. Dosažené parametry vytvořené aplikace byly následně ověřeny na maximálně zatížených linkách při různých velikostech a strukturách složených paketů. Při správné konfiguraci aplikace a architektury dokáže aplikace na všech maximálně zatížených linkách platformy LS2088A zpracovávat složené pakety bez jakýchkoli ztrát. Procesor, na němž se aplikace vyvíjela, je první ze série předprodukčních vzorků a řada funkcí uvedených v dokumentaci je nepředvídatelných nebo dokonce nefunkčních.

Abstract

This work deals with design, implementation and testing of algorithm for packet processing on high–speed computer network. Main goal was to create fully functional diagnostic application intended for exact reconstruction of segment of the network communication, which allows parsing of package packets and then reordering and saving of parsed packets on fully loaded network links 40Gbps of LS2088A platform, without any missed packets. Aplication deals with package packets instead of normal ones, mainly because of increasement of perfomance. Processor LS2088A contains hardware accelaration unit DPAA2, which is higly complex subsystem and has more than thousand configurations. Finding of appropriate configuration of platform was also part of the work. There are many functional configurations, but most of them have poor performance characteristics. Parameters of application were verified on fully loaded network links with different sizes and structures of package packets. If architecture and application are correctly configured, application reach 100% permeability on fully loaded links of the LS2088A platform without any missed packets. The processor used for development of application is preproduction sample and many of advertised featured have unpredicable behaviour or they are even non–functional.

Obsah

1 Úvod

V dnešní době jsou situace, jako například odposlechy, kdy je potřeba přesně zachytit a uložit část síťové komunikace tak, aby se dala zpětně rekonstruovat beze ztrát a ve správném pořadí. Cílem této práce je vytvořit takovou diagnostickou aplikaci, která toto zvládne. Dlouhodobým trendem je zpracovávat síťový provoz na co nejvyšší rychlosti, tudíž i tato vytvořená diagnostická aplikace ukládající síťový provoz musí dokázat pracovat na vysoké rychlosti tak, aby nedocházelo ke ztrátám. K tomu napomáhají například komplexní hardwarové akcelerační jednotky, které pro dobrou výkonnostní charakteristiku potřebují správnou konfiguraci. Také zpracovávání velkého množství malých paketů je mnohem více náročné na síťové komponenty, než zpracování menšího počtu velkých paketů. Proto se v této práci provádí zabalení více paketů do jednoho velkého, tzv. složeného paketu, čímž se sníží rychlost mezi přijmutím dvou paketů a vstupní fronta paketů se tak zaplňuje pomaleji. Aplikace se vyvíjela na ARM architektuře, která je ze série předprodukčních vzorků. Byla to tedy jedna z prvních aplikací vyvíjených na této platformě, a proto bylo potřeba odladit vývoj na této platformě. Tato práce měla za úkol také otestovat naměřené hodnoty v předchozí práci Analýza výkonnosti síťového procesoru NXP, kdy se měřila pouze čistá propustnost, bez zpracování paketů.

2 DPDK

DPDK[1] je soubor knihoven a ovladačů pro rychlé zpracování paketů. Dnes podporuje nejpoužívanější typy procesorů, ale pouze v linuxovém prostředí (podmnožina funkcí DPDK funguje také na FreeBSD)[1]. PMD[2] je ovladač, jenž přistupuje k deskriptorům portů přímo bez jakýchkoli přerušení (s výjimkou přerušení změn stavu propojení), aby se pakety rychleji přijímaly, zpracovávaly a doručovaly. Mimo jiné také ukládá pakety přímo do user-space (pamět vyhrazená pro uživatele) a nedochází tak ke kopírování mezi kernel-space (paměť vyhrazená pro OS) a user-space.

Klíčové části, ze kterých se DPDK skládá, jsou EAL[3] která zajišťuje komunikaci s operačním systémem a přístup k nízkoúrovňovým zdrojům, jako je paměťový prostor, PMD ovladače síťových karet a mempool knihovna, která umožňuje naalokovat paměť při spuštění a tím se vyhnout dalším alokacím na hromadě (část paměti, kterou si program za běhu rezervuje), které by vedly k přepnutí kontextu procesoru.

2.1 Použité knihovny DPDK

1. **librte_ring** – Tato knihovna obsahuje implementaci bez–zámkového kruhového bufferu. Tento kruhový buffer je tzv. multi-producer, multi-consumer, což znamená, že z něj může více vláken zároveň číst i do něj zapisovat. V mé aplikaci je použit jako základní komunikační médium mezi vlákny přijímajícími síťové pakety a vláknem, jež má za úkol tyto pakety uložit do souboru. Navíc podporuje možnost čtení/zápisu do kruhového bufferu po vícero prvcích, tzv. bulk, což výrazně zvyšuje výkon.

2. **librte_mempool** – Zajišťuje alokaci kontejnerů objektů v paměti. Každý mempool je identifikován pomocí unikátního jména a implementován pomocí již zmíněného kruhového bufferu. Při alokaci nového objektu je tedy možné vybrat ve kterém mempoolu bude nový objekt alokován. Všechny prvky daného mempoolu mají fixní velikost definovanou při vytváření mempoolu. Je také možné přiřadit mempool k určité frontě, pakety příchozí do této fronty budou tedy alokovány v daném mempoolu.

3. **librte_mbuf** – Knihovna obsahující implementaci mbufů, do nichž jsou ukládány převážně příchozí pakety pro snadnější manipulaci a zvýšení výkonnosti. Tyto mbufy jsou alokovány v mempoolech a mají fixní velikost, proto pokud potřebujeme uložit paket větší než je velikost jednoho mbufu, je potřeba paket rozdělit do vícero mbufů a ty poté svázat do seznamu.

4. **rte_reorder** – Knihovna využívající dva kruhové buffery pro implementaci algoritmu, jež seřadí pakety dle jejich sekvenčních čísel. Jeden zvaný 'ready', obsahující již seřazené prvky a druhý zvaný 'order', obsahující prvky jež na seřazení teprve čekají. Podporuje multi-consumer, tedy z něj lze číst z vícero vláken zároveň, což v mé aplikaci využito není, protože k zápisu do souboru by bylo nutno tyto vlákna synchronyzovat tak, aby zapisovalo vždy jen jedno vlákno (aby pakety nebyly uloženy ve špatném pořadí), což by bylo zbytečné.

5. **rte_eal** – Viz druhý odstavec DPDK – poskytuje rozhraní, které před aplikacemi a knihovnami skrývá implementační detaily a specifika dané architektury a rozhoduje, jak nejlépe alokovat tyto zdroje (jako je paměťový prostor, PCI zařízení, konzole atp.).

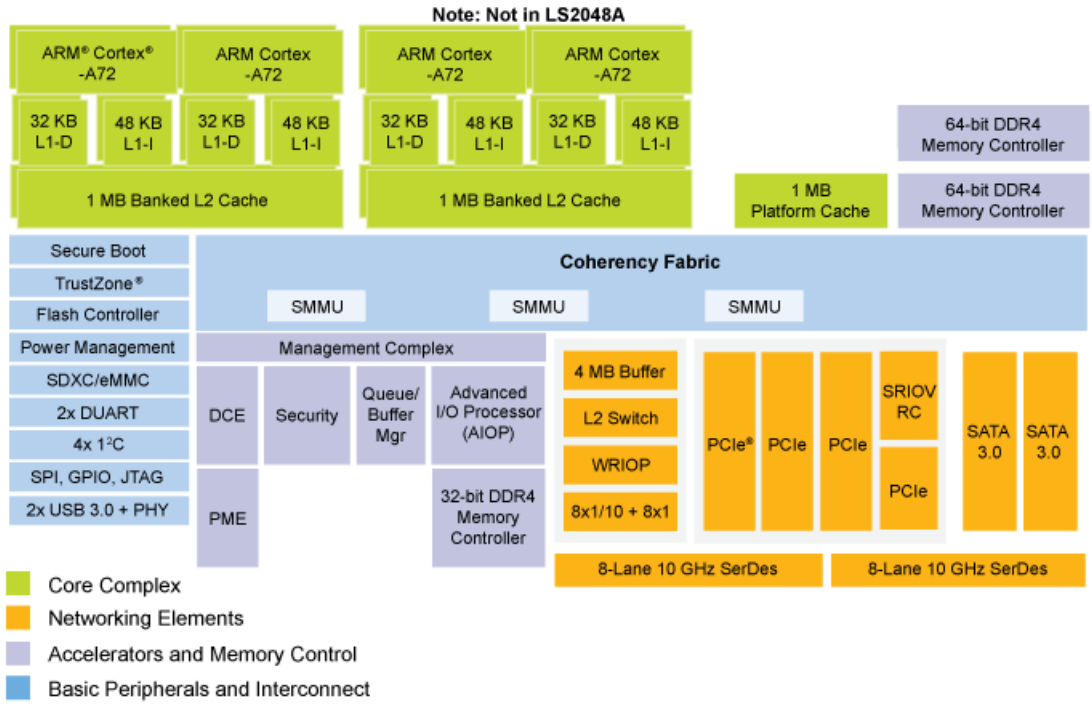
[1] Data plane development kit
[2] Poll Mode Driver
[3] Environment Abstraction Layer

3 Platforma

Hlavním výpočetním prvkem platformy LS2088A je procesor NXP QorlQ LS2088A. Procesor disponuje osmi 64bitovými výpočetními jádry ARM Cortex-A72 s maximálním taktem 2 GHz. Každé výpočetní jádro má k dispozici 32KB datové L1 cache paměti a 48KB instrukční L1 cache paměti. Jádra jsou spárována do dvojic a každá dvojice má k dispozici 1MB sdílené L2 cache paměti. Dohromady platforma disponuje 4MB L2 cache paměti. Disponuje také akcelerační jednotkou DPAA2 a L2 Switchem, který na základě nastavených pravidel funguje jako demultiplexor a podle MAC adresy paketu jej přesměruje do vybrané fronty, kde každé jádro má svou frontu [2].

Další funcionality tohoto procesoru v oblasti zpracování síťového toku

1. 10 Gb/s Pattern Matching regulerních výrazů
2. 20 Gb/s Datové komprese
3. 20 Gb/s SEC krypto akcelerace



Obrázek 1: Blokové schéma procesoru QorIQ LS2088A

3.1 Referenční návrhová deska LS2088A-RDB

Tato deska poskytuje komplexní platformu, která umožňuje využití procesoru LS2088A ve standardním linuxovém prostředí a síťovou inteligenci s novou generací Datapath (DPAA2). Pro zpracování síťového provozu deska poskytuje 4 SFP+ a 4 RJ45 porty s rychlostí až 10 Gb/s. Na desce jsou také vyvedeny čtyři porty 1Gb/s. Disponuje 128 MB NOR a 2GB 8-bit NAND flash paměti. Obsahuje 64 MB EEPROM a dvě 72bitové DDR4, 4 GB na slot. Pro periferie jsou připraveny dvě SATA rozhraní, dva USB 3.0 porty a sběrnici PCI Express třetí generace.

3.2 Programovatelné hradlová pole

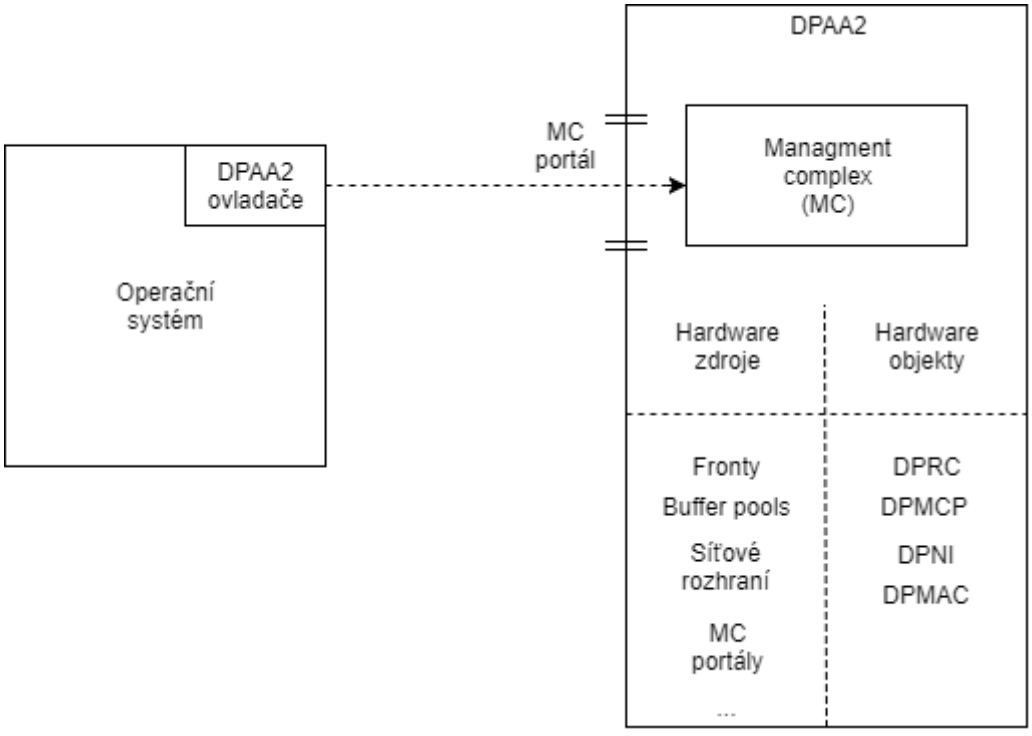
Programovatelné hradlová pole, dále jen FPGA, je typ logického integrovaného obvodu, který je vyroben tak, aby mohl být naprogramován (vytvořen design) až u zákazníka. Obsahuje pole programovatelných logických obvodů (PLD), logických bloků, umožňuje je navzájem propojit a tím vytvořit takřka libovolné číslicové zařízení[3].

FPGA nebylo součástí této práce, ale sloužilo k vytváření složených paketů (paket skládající se z vícero paketů), jež má aplikace vytvořená v rámci této práce zpracovávat.

3.3 DPAA2

DPAA2[5] je hardwarová architektura určená pro vysokorychlostní síťové zpracování paketů. DPAA2 se skládá ze sofistikovaných mechanizmů pro zpracování ethernetových paketů, správu fronty, správy vyrovnávacích pamětí, autonomního přepínání L2, virtuálního přemostění sítě ethernet a akcelerátorů – například kryptografických či akcelerátoru zpracování paketů o rychlosti 20Mp/s.

Hardwarová součást DPAA2 s názvem Management Complex (dále jen MC) spravuje hardwarové prostředky DPAA2. MC poskytuje objektovou abstrakci pro softwarové ovladače určené pro použití hardwaru DPAA2. MC používá hardwarové prostředky DPAA2, jako jsou fronty, vyrovnávací paměti a síťové porty pro vytvoření funkčních objektů / zařízení, jako jsou network interface (síťová rozhraní), L2 Switch nebo akcelerátory. MC také poskytuje paměťově mapované vstupně–výstupní příkazové rozhraní (MC portály), které softwarové ovladače DPAA2 používají pro práci s objekty DPAA2[4].

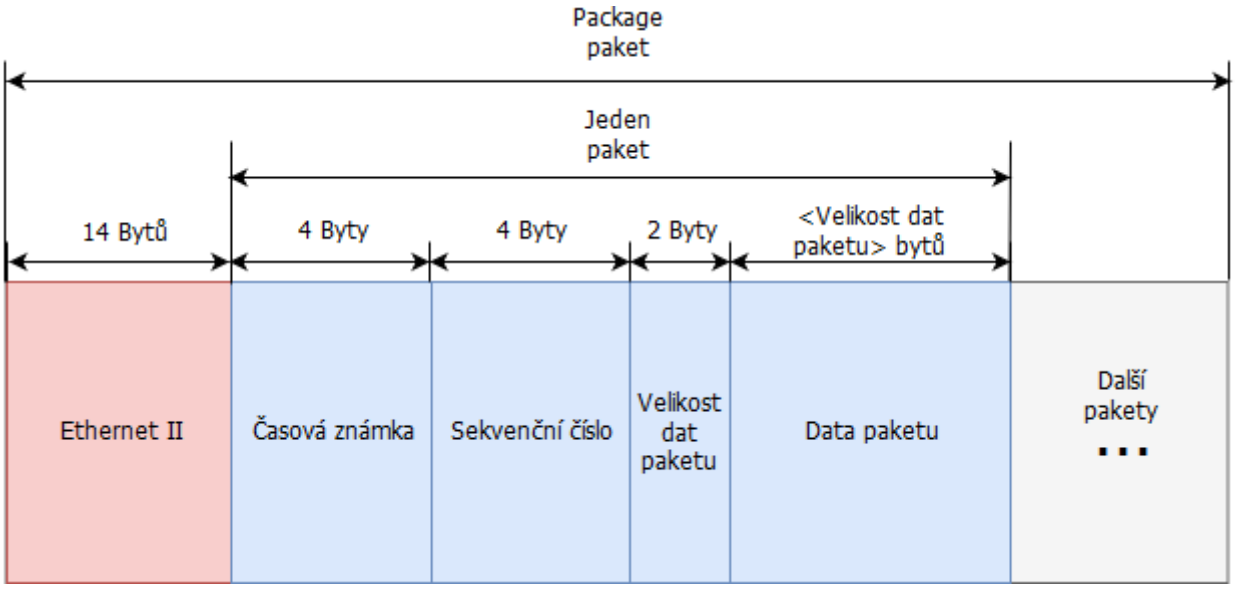


Obrázek 2: DPAA2 diagram

[5]Data Path Acceleration Architecture Gen2

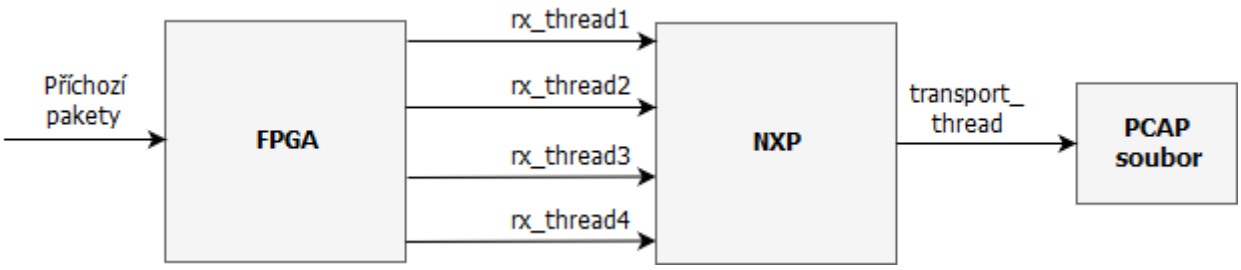
4 Implementace

Vytvořená aplikace ukládá síťový provoz do PCAP souboru s přesnými časovými známkami a ve správném pořadí. Aby to bylo možné, jsou všechny pakety nejdříve zpracovány v FPGA, kde se každému paketu přiřadí sekvenční číslo a časová známka a poté se zabalí do složeného paketu. Složené pakety jsou takové pakety, které se skládají z ethernetové hlavičky a sekvence obyčejných paketů. Viz obrázek 3. Až takovýto složený paket dosáhne určité velikosti (10 kB), případně po vypršení časového limitu, tak je odeslán do NXP procesoru.



Obrázek 3: Složený paket

Aplikace běžící na procesoru NXP LS2088A celkově využívá až 5 vláken (jader), kde jedno až čtyři (podle počtu alokovaných portů) vlákna slouží ke zpracovávání vstupu – mají za úkol přijímat a zpracovávat složené pakety vytvořené v FPGA a poté je ukládat do komunikačního kruhového bufferu pro exportní vlákno. Viz obrázek 4. Toto ukládání probíhá až do doby, kdy je volné místo v mempoolu.



Obrázek 4: Implementace

Poslední vlákno spuštěné na řídícím jádře je exportní, jenž už v průběhu čtení vstupu načítá data z komunikačního kruhového bufferu a vkládá je do reorder bufferu. Ten pakety seřadí na základě sekvenčního čísla a

v případě potřeby (zaplnění reorder bufferu) provede předčasný export/exporty paketů ve správném pořadí do pcap souboru. V případě zaplnění mempoolu (tedy už není možné přijat žádný další paket) nebo přerušení programu (linux – ctrl+c) provede úplný export všech paketů do souboru.

Protože složené pakety odesílané z FPGA do NXP procesoru mohou mít velikost maximálně 10kB a jsou odeslány z FPGA až když jsou maximálně zaplněny (případně po vypršení časového limitu), jsou tyto příchozí pakety ukládány do 10kB mbufů (Mbuf – datová struktura sloužící k ukládání a jednodušší/efektivnější manipulaci s pakety). Po zpracování těchto složených paketů jsou jednotlivé pakety ukládány do mbufů o velikosti 2kB. Pokud je paket větší než 2kB, je uložen do více mbufů svázaných do lineárního seznamu.

Ačkoli export již není důležitý z pohledu časové složitosti, program má dvě možnosti exportu paketů do souboru, a to jsou:

1. Všechny pakety z tohoto seznamu se nakopírují do jednoho bufferu, který je poté zapsán pomocí funkce pcap_dump().
2. Jednotlivé pakety jsou ukládány do souboru postupně pomocí systémové funkce fwrite().

Z pohledu časové složitosti se jako optimálnější způsob ukázala první možnost.

4.1 Yocto

Yocto je projekt, jenž dokáže vytvářet systémy založené na linuxu nazávisle na architektuře hardware. Byl využit k vytvoření systému, jenž byl nabootován na NXP zařízení. Yocto bylo rozšířeno tak, aby překládalo i vytvořenou diagnostickou aplikaci a tedy bylo možné ji spouštět přímo na procesoru NXP.

4.2 PCAP

PCAP je API [6] pro zachytávání síťové komunikace (zkratka Packet Capturing). Aplikace využívající knihovnu libpcap/WinPcap dokážou pracovat s formátem souborů PCAP. V tomto formátu ukládají síťovou komunikaci od úrovně linkové vrstvy výše. Dokážou tedy vytvářet i číst zmiňované soubory, což se v aplikaci využívá pro zápis přijatých paketů.

4.3 Fronty

Fronty paketů jsou zpravidla základní součástí libovolného síťového zařízení. Umožňují komunikaci asynchronních modulů a zvyšují výkon.

Fronta paketů je zpravidla implementována jako vyrovnávací paměť vstupu (first-in, first-out – FIFO) s pevnou velikostí. Fronta neobsahuje paketová data. Místo toho se skládá z deskriptorů, které odkazují na pakety v operační paměti. Pakety se do síťové karty následně dostanou pomocí DMA přenosů.

4.4 Postup a problémy při implementaci

Při měření byl od začátku problém s poměrem přijatých (zpracovaných) ku chybným paketům, kdy chybných paketů bylo o řád více než přijatých, ovšem zahozených bylo minimum. Při vytvoření programu, jenž pouze přijímal pakety a ukládal je do naalokovaného kruhového bufferu byl poměr přijatých (zpracovaných) ku chybným paketům stejný. Zajímavé ovšem bylo, že pakety byly označeny jako chybové, nikoli jako zahozené. Chybové pakety jsou v dokumentaci definovány pouze jako chyby hardware.

Při snaze o snížení tohoto poměru přijatých/chybových paketů – například pomocí různých kombinací konfigurací DPAA2 jako změny počtu front na port, počtu front na jádro, různých konfigurací síťových rozhraní, přidáním managment complex objektů a v neposlední řadě zvýšením počtu deskriptorů vstupních front, tedy jejich zvětšení se neprojevilo žádné výrazné zlepšení.

[6] Aplikační rozhraní

Nakonec jsem přišel na řešení, že jsou tyto pakety zahozeny, protože nejsou žádné volné mbufy, do nichž by se příchozí pakety uložily. Proto jsou tyto pakety označeny jako chybové a ne jako zahozené. Tudíž jsem odesílal pouze dávku (burst) paketů, jež přesně zaplnil mempool a statistiky se zlepšily, ovšem pořád nebyly ideální.

Poté jsem začal pracovat s více porty zároveň a tedy více vlákny, kdy program v některých případech končil chybou neoprávněného přístupu do paměti. Ovšem řešit tento problém na NXP procesoru nebylo snadné, zejména protože není možné použít valgrind na DPDK aplikace – pouze na x86 architektuře. Proto bylo potřeba přidat do yocta další recept s gdb a gdb–serverem, pomocí nichž už se chybu podařilo odhalit. Chybou bylo nenastavení kruhového bufferu mezi přijímacími a exportním vláknem jako "multi–producer", tedy při zápisu z více vláken zároveň vznikaly chyby. Velký počet chybových paketů to ale neřešilo, ten se podařilo odstranit až s nahrazením funkce rte_pktmbuf_ops_get_count(), která vracela počet volných mbufů sloužící k detekci, že došly volné mbufy. Tato funkce podle všeho prochází celý mempool a počítá již alokované mbufy, což výrazně snižovalo výkonnost a tedy způsobovalo vysoký počet chybových paketů.

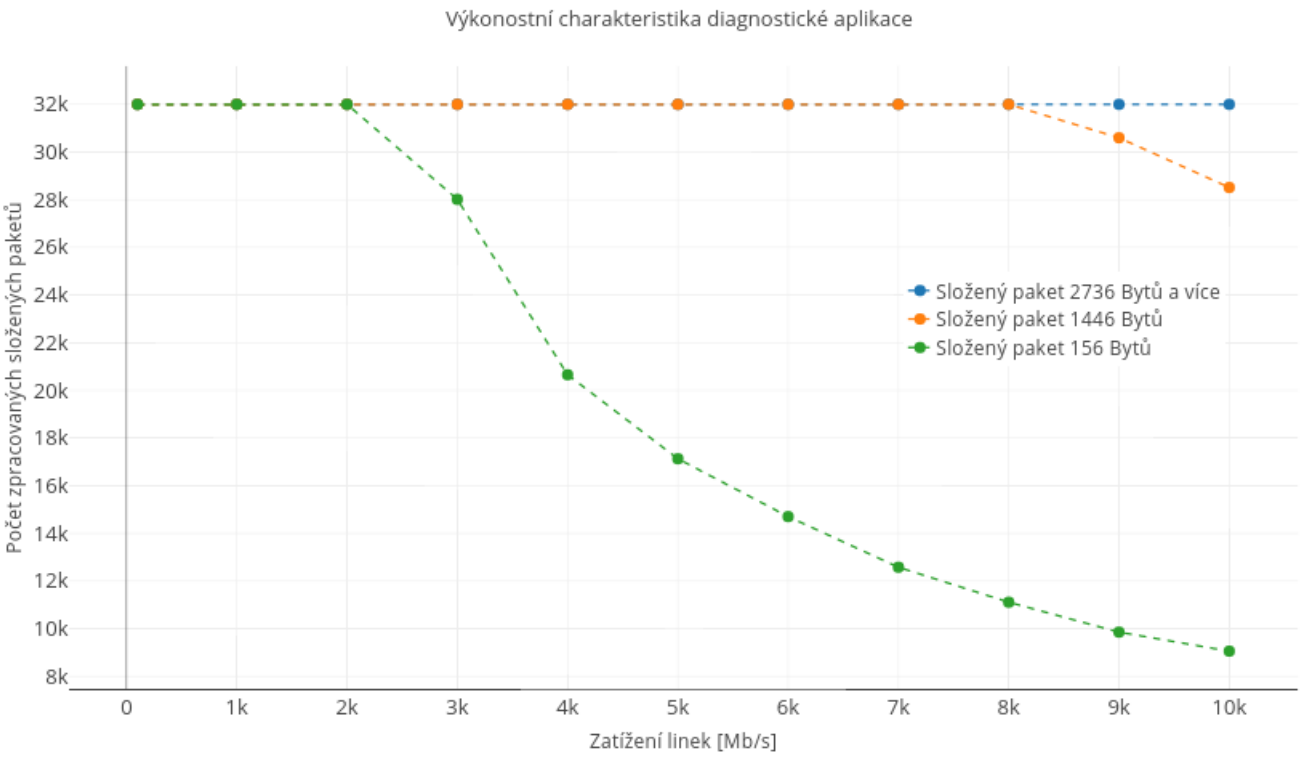
K navýšení výkonnosti také pomohlo vyměnit vadné optické kabely/transceivery za funkční. Při konfiguraci DPAA2 se NXP procesor často zaseknul a následně bylo potřeba celý systém restartovat. Dokumentace byla mnohdy neúplná, což je způsobeno tím, že software k CPU je ještě v betaverzi a mnohdy bylo nutné najít informace až v implementaci.

5 Testování a měření

Pro ověření správnosti a kvality implementace bylo potřeba provést testování a výkonnostní měření.

Při měření se zjistilo, že čím větší pakety přicházejí na vstup aplikace, tím se výkonnost zvyšuje. Je to dáno tím, že při přenosu na 10 Gb/s lince je doba mezi přijmutím dvou malých paketů mnohem kratší, než doba mezi přijmutím dvou velkých paketů. Takže čím menší pakety přicházejí, tím je jich více a tím rychleji se zaplní vstupní fronta jednotlivých portů. V případě, že je vstupní fronta plná, paket je uvolněn, tedy zahozen.

To znamená, že od určité velikosti paketů (cca. 3kB) nahoru dokáže aplikace pracovat beze ztrát (zahozených paketů). Vzhledem k tomu, že FPGA odesílá maximálně zaplněné složené pakety až do 10kB, zvládá aplikace zpracovávat složené pakety na plně vytížených linkách (40Gb/s) beze ztrát viz graf níže. Měření probíhalo na dvou portech, kde na každý byl odeslán burst 16000 složených paketů a měřilo se, kolik dokáže aplikace běžící na NXP procesoru zpracovat a kolik se zahodí v závislosti na zátěži linky a velikosti složeného paketu.



Obrázek 5: Výkonnostní charakteristika diagnostické aplikace

5.1 Spirent

Spirent SPT–2000A je nástroj pro testování síťových zařízení. Základní vlastností Spirentu je generování paketových toků. Toky jsou definovány v tzv. streamblocku, v nichž je možné definovat hodnoty jednotlivých položek v hlavičkách podporovaných protokolů.

V daném spirentu byl použit rozšiřující modul se dvěmi SFP+ porty, které jsou schopny přenést až 10 Gb/s.

5.2 Generátor složených paketů

V době vývoje aplikace ještě nebyl hotový design FPGA, jež generuje složené pakety, proto bylo pro testovací účely na platformě x86 potřeba vymyslet alternativní způsob generování těchto paketů. Nejlepším řešením bylo vytvoření aplikace, jež příchozí pakety balí do jednoho společného paketu o zadané maximální velikosti i s informací o jeho velikosti, sekvenčním číslu (kde se dalo dobře otestovat, zda funguje přeskládávání paketů), a časové známce. Poté je uloží do pcap souboru, ze kterého se daly jednoduše pomocí virtuálního zařízení přeposlat na vstup diagnostické aplikace.

Později při testování na platformě LS2088A tento způsob nebyl dostatečný a jako další řešení se nabízelo použít spirent, kde bylo možné si vytvořit šablonu složeného paketu a tyto pakety poté posílat rychlostí až 10 Gb/s.

6 Závěr

Celá práce byla od počátku velmi závislá na porozumění akcelerační jednotce DPAA2 a DPDK a jejich správné konfiguraci. Hlavní úlohou bylo ovšem implementovat diagnostickou aplikaci, jež provede zpracování složených paketů a uloží je do souboru. Zde se vyskytla řada problémů spojených s vysokým počtem chybných paketů. Zejména bylo nutné provést mnohé optimalizace k dosažení co nejvyšší propustnosti. Nakonec byl vytvořen plně funkční program, jenž bez jakýchkoli ztrát dokáže zpracovávat složené pakety na všech linkách o plné rychlosti 10Gb/s, tedy u platformy LS2088A dohromady 40Gb/s. Vzhledem k tomu, že design FPGA ještě nebyl v době vývoje aplikace hotový, tak ani aplikace není otestovaná na reálných datech, s nimiž je určena pracovat, ale pouze na datech generovaných zařízením Spirent a malých datech zachycených pomocí programu tcpdump, jenž však dokáží věrohodně nasimulovat reálná data, která se z FPGA budou odesílat na vstup diagnostické aplikace běžící na NXP procesoru. Pro správné fungování diagnostické aplikace je potřeba provést správnou konfiguraci DPAA2 pomocí přiloženého skriptu a zadat správné parametry aplikaci při startu. Zejména masku jader pro vyřazení nultého jádra (protože použití nultého jádra není doporučováno – slouží ke zpracování všech přerušení a jeho použití pro síťové aplikace způsobuje nedeterministický výkon – platí pouze pro procesory řady LS), poměr velikostí jednotlivých bufferů a počtu mbufů, jež je možné do jednotlivých mempoolů uložit. Velikost mempoolů, z nichž jsou alokovány mbufy, mají architekturou omezenou velikost, jenž se nepodařilo obejít a tudíž bylo potřeba vytvořit alespoň dva mempooly. Charakteristika výkonnosti aplikace odpovídala hodnotám naměřeným v předchozí práci Analýza výkonnosti síťového procesoru NXP.

Literatura

Reference

[1] *Data plane development kit: DPDK* [online]. [cit. 2017-12-12]. Dostupné z: http://dpdk.org

[2] *QorIQ 2088A: NXP* [online]. [cit. 2017-12-12]. Dostupné z: https://www.nxp.com/products/processors-and-microcontrollers/arm-based-processors-and-mcus/qoriq-layerscape-arm-processors/qoriq-layerscape-2088a-and-2048a-multicore-communications-processors:LS2088A?&fsrch=1&sr=1&pageNum=1

[3] Programovatelná hradlové pole. *Wikipedia* [online]. [cit. 2018-05-03]. Dostupné z: https://cs.wikipedia.org/wiki/Programovateln%C3%A9_hradlov%C3%A9_pole

[4] *Data Path Acceleration Architecture Gen2: DPDK* [online]. [cit. 2017-12-13]. Dostupné z: http://dpdk.org/doc/guides/nics/dpaa2.html