Dokumentace: Paketový filtr pro síťový provoz s rychlostí 100 Gb/s

User documentation

Lukáš Kekely, Martin Žádník

Technical documentation FIT-TR-2014-XXX

Dokumentace: Paketový filtr pro síťový provoz s rychlostí 100 Gb/s

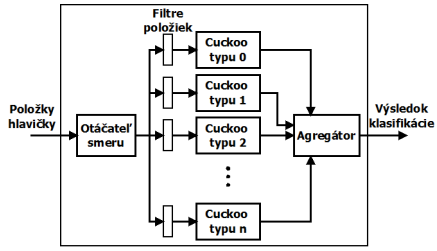
Lukáš Kekely, Martin Žádník

Fakulta informačních technologií
Vysoké učení technické v Brně
Božetěchova 1/2, 612 66 Brno
{ikekely, izadnik}@fit.vutbr.cz

Abstrakt Tento technický report obsahuje dokumentaci k paketovému filtru určeného pro filtraci paketů na rychlosti 100 Gb/s.

1 Popis fungovania

Základom je haš tabuľka s implementovaným mechanizmom Cuckoo hash na zvýšenie jej kapacity. Top-level schéma filtra je zachytená na nasledujúcom obrázku 1.



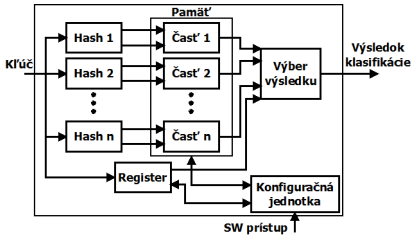
Obrázek 1. Architektura jednotky.

Popis zobrazenej top-level schémy:

- Položky hlavičky = IP adresy, čísla portov a protokol získané z hlavičky paketu (napr. pomocou HFE-X)
- Otáčateľ smeru = pre každý paket najprv prepustí položky hlavičky tak ako prišli a potom ich ešte raz zopakuje s navzájom vymenenými zdrojovými a cieľovými položkami. Pre každý paket sa tak vykoná porovnanie akoby v dvoch smeroch a sú získané 2 výsledky porovnania.
- Filtre položiek = realizujú jednoduchý výber zaujímavých položiek z hlavičiek pre danú haš tabuľku. Inak povedané, z extrahovaných položiek hlavičiek vytvoria konkrétne kľúče do haš tabuliek jednotlivých typov (napr. zdrojová IP adresa a protokol, alebo prefix IPv6/64).

– Cuckoo typu x = haš tabuľka s Cuckoo hash mechanizmom uchovávajúca daný typ pravidiel.
– Agregátor = spájanie informácií z jednotlivých haš tabuliek pre jedno porovnanie, a taktiež spájanie výsledkov pre oba smery porovnania jedného paketu. Pri spájaní výsledkov jedného porovnania sa, v prípade úspešného vyhľadania vo viacarých tabuľkách súčasne, vezme ten pre najvyšší typ pravidla (implicitná priorita rastúca s číslom typu pravidla).
– Výsledok klasifikácie = spojenie výsledkov pre oba smery vyhľadávania pre jeden paket. Teda 2x identifikátor zodpovedajúceho pravidla a príznak úspešnosti vyhľadania.

Podrobnejšia štruktúra jednotky realizujúcej haš tabuľku s Cuckoo hash mechanizmom pre jeden typ pravidiel je zobrazená na nasledujúcej schéme na obrázku 2.



Obrázek 2. Zapojení hash jednotky.

Popis schémy haš tabuľky s Cuckoo hash.

– Klúč = jednoznačný identifikátor položiek tvorený vybranými položkami hlavičiek paketov. Obecne može ísť o bitový vektor danej šírky, ktorý sa v tabuľke vyhľadáva.
– Hash x = jednotlivé hašovacie funkcie určujúce adresu prístupu do pamäte s haš tabuľkou. Každý vstupný kľúč je hľadaný na toľkých pozíciách, koľko je hašov (aspoň 2). Hašovacie funkcie sú implementované vybraním spodných bitov zo 16b CRC. Vzájomná rozdieľnosť haš funkcií je zaistená predradením rôzne konfigurovaných permutácií (statická konfiguráci + nastaviteľné semienko)
– Pamäť = tvorí základ pre ukladanie záznamov do filtra. Každý záznam je ukladaný ako dvojica (kľúč, identifikátor), kde kľúč cez haš určuje aj pozíci záznamu v pamäti. Pamäť je implicitne rozdelená na časti, kde do každej časti ukazuje práve jedna hašovacia funkcia. Rozdelenie zvyšuje priepustnosť (paralelný prístup do BRAM) a tiež zlepšuje dosiahnuteľnú zaplnenosť pamäti (empirický poznatok). Vyhľadávanie vstupného kľúča v pamäti pozostáva z vyčítania záznamov z pozícií daných hašmi a ich porovnaním so vstupným

kľúčom na zhodu (nezabudnúť overiť platnosť vyčítaného záznamu = valid bit). Informácia o nájdenej zhode a tiež identifikátory zo záznamov sú propagované ďalej.
- Register = paralelne s pamäťou sa porovnanie na zhodu realizuje aj pre záznam uložený v registry. Tento register je využívaný na zaistenie vhodného fungovania HW riadeného preskladávania položiek v pamäti popísaného ďalej.
- Výber výsledku = realizuje agregáciu výsledku vyhľadania pre všetky časti pamäte a register. Pri správnom plnení haš tabuľky, by mala obsahovať každý kľúč maximálne raz, teda každé vyhľadanie by malo skončiť nájdením maximálne jednej zhody. Výber výstupného identifikátoru je preto možné realizovať ako výber prvého s nájdenou zhodou (nájdenie prvej jednotky riadiaci multiplexor). Informáciu o úspešnosti vyhľadávania je možné agregovať OR funkciou.
- Výsledok klasifikácie = predstsavuje príznak úspešnoti vyhľadávania (nájdený vstupný kľúč v haš tabuľke?) a identifikátor patriaci zhodnému kľúču (má zmysel len pri úspechu vyhľadania).
- Konfiguračná jednotka = je riadená zo SW a realizuje všetky kroky potrebné pre atomické pridávanie a odoberanie záznamov z haš tabuľky. Zo SW teda chodia len príkazy tvaru: ODOBER_ZÁZNAM(kľúč) a PRIDAJ_ZÁZNAM(kľúč,identifikátor). Jednotka využíva register na odkladanie záznamov z haš tabuľky pri práci s ňou. Výhodou tohto registru je jeho zapojenie do porovnávacej cesty, ktoré vedie na možnosť nepozastavovať funkčnú cestu jednotkou počas jej konfigurácie (záznam odložený v ňom je stále súčasťou porovnávania). Na výpočet haš funkcií jednotka obsahuje len jeden CRC blok, znásobené sú permutačné bloky nastavené rovnako ako v porovnávacej ceste. Odoberanie záznamu je realizované po jednom atomicky tak, že sa pravidlo postupne hľadá a prípadne ruší na každej jeho možnej pozícii (vrátane registra). Počas odoberania je pozastavený proces preskladávania položiek haš tabuľky. Pridávanie záznamu je realizované jednoducho zápisom nového záznamu do registra. Následne sa jednotka snaží register vyprázdniť zaradením nového záznamu do haš tabuľky postupne na jednu z pozícií daných hašmi. V prípade, že preň nie je voľné miesto, je nový záznam vymenený so zvoleným záznamom z pamäte, ktorý sa tak dostáva do registra. Tento proces (preskladávanie) sa opakuje, dokým nie je vyprázdnený register. Pokiaľ je register plný jednotka sa pre SW tvári akoby mala zaplnenú haš tabuľku. V prípade prázdneho registra je možné úspešne vložiť aspoň jeden nový záznam.

2 Adresný priestor

Adresný priestor je rozdělen na prostor pro zápis (vizte tabulku 1) a prostor pro čtení (vizte tabulku 2).

2.1 Obsah dát

TYPE_MASK

- bitová maska definujúca informáciu/platnosť operácie pre jednotlivé typy pravidiel
- i-ty bit odspodu (od LSB) sa vzťahuje na pravidlá pravidlá typu i, kde i je celé číslo z rozsahu <0,RULE_TYPES)
- bit hodnoty 1 = požadovaná informácia/operácia je platná pre daný typ pravidiel
- bit hodnoty 0 = požadovaná informácia/operácia je neplatná pre daný typ pravidiel
- NAPR: čítanie STATUS_FULL = 0x00000002 znamená, že pravidlá typu 1 sú plné a ostatné typy majú ešte voľné miesto
- NAPR: zápis IGNORE_REQ = 0x00000005 znamená, že sa zapína ignorovanie pravidiel typu 0 a 2, zároveň sa ruší ignorovanie ostatných typov pravidiel

TYPE_CFG

- podrobnejší popis pri preklade nastavených parametrov jednotlivých typov pravidiel podporovaných filtrom
- interný formát 32b hodnoty TYPE_CFG je: FLAGS(6b), ITEMS(18b), TABLES(8b)
- FLAGS = príznaky pre jednotlivé prvky pätice tvoriacej kľúč definujúce, ktoré z nich sú pri pravidlách daného typu využívané
- interný formát 6b hodnoty TYPE_CFG.FLAGS je: SRCIP, DSTIP, SRCPORT, DSTPORT, PROTO, IPV4_ONLY
- hodnota bitu 1 znamená, že daná položka je pre kľúč pravidla potrebná, naopak bit 0 znamená, že daná položka sa ignoruje
- príznak IPV4_ONLY hodnoty 1 povoľuje použitie len IPv4 adries, v prípade hodnoty 0 je povolené použiť IPv6 aj IPv4 adresy
- ITEMS = kapacita (maximálny počet záznamov) v jednej tabuľke Cockoo Hash bloku
- TABLES = počet tabuliek použitých pre Cockoo Hash blok realizujúci daný typ pravidiel
- NAPR: RULES_INFO[0] = 0x80020003 znamená, že kľúč pravidiel typu 0 pozostáva len zo zdrojovej IP adresy a tieto pravidlá majú vyhradené 3 tabuľky po 512 položiek

DIRRECT_ADDR

- adresa pre priamy prístup do Cockoo Hash tabuliek
- interný formát 32b hodnoty DIRRECT_ADDR je: RULE_TYPE(rtw) , TABLE(tw) , ITEM(iw)
- kde rtw=log2(RULE_TYPES), iw=log2(RULES_INFO[DIRRECT_ADDR.RULE_TYPE].ITEMS) a tw=32-rtw-iw
- RULE_TYPE = celé číslo označujúce typ pravidla z rozsahu <0,RULE_TYPES)
- TABLE = číslo tabuľky v Cockoo Hash bloku pre daný typ pravidiel z rozsahu <0,RULES_INFO[DIRRECT_ADDR.RULE_TYPE].TABLES>
- maximálna hodnota povoleného rozsahu je vyhradená pre adresovanie preskladávacieho registru v danom Cockoo Hash bloku

- ITEM = číslo položky v zvolenej tabuľke Cockoo Hash bloku
- NAPR: DIRRD_REQ = 0x800002FF pri rtw=2 a iw=9 znamená, že je čítané pravidlo typu 2 z tabuľky 1 a pozície 255

Firmvérovej komponente je možné jednoducho nastaviť podporované typy pravidiel aj kapacitu. Konfigurácia sa realizuje v súbore filter_core_config.vhd pred syntézou dizajnu. Potrebné je vyplniť 2 časti konfigurácie v uvedenom súbore, je možné sa pritom riadiť komentármi priamo v zdrojovom kóde. Nastavujú sa hlavne parametre

- Počet typov pravidiel = koľko rôznych typov pravidiel (číslované od 0) bude filter podporovať. Povolené hodnoty sú z rozsahu 1 až 30.
- Pole definícií typov = definuje ako budú vyzerať kľúče jednotlivých typov a aká kapacita bude pre ne vyhradená. Počet položiek je podľa nastavenej hodnoty FILTER_RULE_TYPES. Typy sú radané od typu 0 a priorita pri vyhľadávaní rastie s číslom typu. Pre každý typ je potrebné definovať nasledujúce hodnoty:
- key_width = šírka vektoru tvoriaceho kľúč v bitoch
- key.srcip = dĺžka prefixu zdrojovej IP adresy použitá v kľúči (0 znamená nepoužívať)
- key.dstip = dĺžka prefixu cieľovej IP adresy použitá v kľúči (0 znamená nepoužívať)
- key.srcport = má sa použiť zdrojové číslo portu v kľúči
- key.dstport = má sa použiť cieľové číslo portu v kľúči
- key.proto = má sa použiť číslo L4 protokolu v kľúči
- key.ipv4_only = podpora len pre IPv4 adresy (vzťahuje sa na srcip a dstip)
- items = veľkosť jednej časti pamäte s haš tabuľkou v počte položiek (musí byť mocnina 2)
- tables = počet častí pamäte (použitých haš funkcií na prístup)
- bram_type = nastavenie šírky použitých BRAM pamätí na vyskladanie priestoru pre haš tabuľku (obecne nemeniť!)
- use_crc_reg = povolenie preregistrovania jednotky pred CRC blokom (obecne nemeniť!)
- use_cmp_reg = povolenie preregistrovania jednotky vnútri komparátoru (obecne nemeniť!)

```
constant FILTER\_CFG : config_t := (
    (
      key_width   => 24,
      key         =>
        (
          srcip     => 24,
          dstip     => 0,
          srcport   => false,
          dstport   => false,
          proto     => false,
          ipv4_only => true
```

```
      ),
    items       => 512,
    tables      => 3,
    bram_type   => 36,
    use_crc_reg => true,
    use_cmp_reg => true
  ),
  -- Skrátená verzia zápisu toho istého
  (24,(24,0,false,false,false,true),512,3,36,true,true)
);
```

3 Softvérové ovládanie filtračnej jednotky

Softvérový nástroj afilterctl ovládá filtr v hardvér. Parametry nástroje jsou zdokumentovány v tabulce 3.

4 Operace

4.1 Pridanie pravidla

Atomicky je pridané jedno špecifikované pravidlo (kľúč+hodnota) do aktuálneho porovnávaného súboru pravidiel.

- **Čakanie kým je filter pripravený na novú operáciu**
- realizované vyčítaním a kontrolou príznakov v BUSY registre, kontroluje sa globálny BUSY príznak a príznak patriaci danému typu pravidiel
- príznak BUSY by mal zotrvať nastavený maximálne rádovo desiatky taktov po začatí poslednej operácie s filtrom
- nedodržanie čakania na pripravenosť môže nešpecifikovane ovplyvniť výsledok poslednej započatej operácie
- **Naplnenie dátových registrov položkami pridávaného pravidla**
- je potrebné vyplniť potrebné položky kľúča aj položku s identifikátorom pravidla (pozn. používá big-endian)
- **Čakanie kým je ukončené pridávanie predošlého pravidla daného typu**
- realizované vyčítaním a kontrolou príznaku vo FULL registre patriaceho danému typu pravidiel
- príznak FULL zotrváva nastavený počas preskladávania položiek Cockoo Hash bloku, ktoré môže štandardne trvať rádovo stovky až tisíce taktov
- ak je príznak FULL nastavený príliš dlho indikuje existenciu neriešiteľnej kolízie v Cockoo Hash bloku a teda jeho zaplnenie
- nedodržanie čakania na ukončenie predošlého pridávania vedie na prepis nešpecifikovaného z pravidiel
- **Zavolanie príkazu na pridanie pravidla**
- realizované zápisom do ADD_REQ s hodnotou nastavenou na masku typov pravidiel, do ktorých je pridávané
- pridanie pravidiel sa navonok vykonáva atomicky a skoro okamžite

4.2 Odobranie pravidla

Po jednom atomicky sú odoberané všetky pravidlá s rovnakým kľúčom ako bol zadaný.

- **Čakanie kým je filter pripravený na novú operáciu**
- realizované vyčítaním a kontrolou príznakov v BUSY registre, kontroluje sa globálny BUSY príznak a príznak patriaci danému typu pravidiel
- príznak BUSY by mal zotrvať nastavený maximálne rádovo desiatky taktov po začatí poslednej operácie s filtrom
- nedodržanie čakania na pripravenosť môže nešpecifikovane ovplyvniť výsledok poslednej započatej operácie
- **Naplnenie dátových registrov položkami mazaného pravidla**
- je potrebné vyplniť potrebné položky kľúča, identifikátory pravidiel sú ignorované
- **Zavolanie príkazu na zmazanie pravidla**
- realizované zápisom do REM_REQ s hodnotou nastavenou na masku typov pravidiel, ktoré majú byť zrušené
- zmazané sú vždy všetky pravidlá daného typu so zhodným klúčom ako bol špecifikovaný
- rušenie pravidiel sa navonok javí ako atomické a vykonané okamžite po skončení spracovania aktuálneho paketu
- rušenie pravidiel je nezávislé na nastavenosti príznaku FULL
- rušenie pravidiel môže viesť na okamžité alebo oneskorené zrušenie aj dlho tvajúceho príznaku FULL (zrušené pravidlo uvoľní kolízne políčko v tabuľke)

4.3 Vyčítanie všetkých pravidiel

Všetky aktuálne platné pravidlá sú postupne vyčítané do SW.

- **Čakanie kým je filter pripravený na novú operáciu**
- realizované vyčítaním a kontrolou príznakov v BUSY registre, kontroluje sa globálny BUSY príznak a aj príznaky patriaci všetkým typom pravidiel (obecne 0xFFFFFFFF)
- príznak BUSY by mal zotrvať nastavený maximálne rádovo desiatky taktov po začatí poslednej operácie s filtrom
- nedodržanie čakania na pripravenosť môže nešpecifikovane ovplyvniť výsledok poslednej započatej operácie
- **Zavolanie príkazu na zastavenie preskladávania položiek v Cockoo Hash tabuľkách**
- realizované zápisom do STOP_REQ s hodnotou nastavenou na jednotkovú masku všetkých typov pravidiel (obecne 0xFFFFFFFF)
- nezastavenie preskladávania môže viesť na niekoľkonásobné prečítanie niektorých pravidiel, prípadne na neprečítanie niektorých pravidiel
- zastavenie preskladávania vedie na pozastavenie snahy HW zrušiť FULL stav, teda nastavenosť FULL príznakov sa počas pozastaveného preskladávania nemení

- **Vyčítaj postupne všetky položky všetkých Cockoo Hash tabuliek všetkých typov pravidiel**
- realizované využítím priameho prístupového módu, teda zápisom do DIRRD_REQ s hodnotou nastavenou na adresu požadovaného políčka tabuliek (presný formát adresy je popísaný v popise firmvéru)
- zavolaním príkazu priameho čítania sú hodnotami čítaného políčka naplnené dátové registre jednotky
- hodnoty z dátových registrov je potom možné postupne vyčítať
- platnosť pravidla na vyčítanej pozícii je možné určiť na základe hodnoty prečítaného identifikátoru
- **Zavolanie príkazu na povolenie preskladávania položiek v Cockoo Hash tabuľkách**
- realizované zápisom do STOP_REQ s hodnotou nastavenou na nulovú masku všetkých typov pravidiel (obecne 0)

4.4 Rušenie všetkých pravidiel

Všetky aktuálne platné pravidlá sú navonok atomicky a okamžite zrušené.

- **Čakanie kým je filter pripravený na novú operáciu**
- realizované vyčítaním a kontrolou príznakov v BUSY registre, kontroluje sa globálny BUSY príznak a aj príznaky patriaci všetkým typom pravidiel (obecne 0xFFFFFFFF)
- príznak BUSY by mal zotrvať nastavený maximálne rádovo desiatky taktov po začatí poslednej operácie s filtrom
- nedodržanie čakania na pripravenosť môže nešpecifikovane ovplyvniť výsledok poslednej započatej operácie
- **Zavolanie príkazu na zastavenie preskladávania položiek v Cockoo Hash tabuľkách**
- realizované zápisom do STOP_REQ s hodnotou nastavenou na jednotkovú masku všetkých typov pravidiel (obecne 0xFFFFFFFF)
- nezastavenie preskladávania môže viesť na nezrušenie niektorých pravidiel
- zastavenie preskladávania vedie na pozastavenie snahy HW zrušiť FULL stav, teda nastavenosť FULL príznakov sa počas pozastaveného preskladávania nemení
- **Zavolanie príkazu na ignorovanie výsledkov porovnávania v HW**
- realizované zápisom do IGNORE_REQ s hodnotou nastavenou na jednotkovú masku všetkých typov pravidiel (obecne 0xFFFFFFFF)
- filter je atomicky prepnutý do stavu, kedy je výsledok každého vyhľadávania ignorovaný a na výstup komponenty je stále zavedená identifikácia neúspešného vyhľadania
- inak povedané, filter atomicky prejde do stavu kedy sa správa akoby aktuálne neobsahoval žiadne pravidlá
- **Vynuluj postupne všetky položky všetkých Cockoo Hash tabuliek všetkých typov pravidiel**

- realizované využítím priameho prístupového módu, teda zápisom do DIRCLR_REQ s hodnotou nastavenou na adresu požadovaného políčka tabuliek (presný formát adresy je popísaný v popise firmvéru)
- zavolaním príkazu priameho mazania je zneplatnená hodnota políčka na definovanej adrese
- alternatívne je možné vyprázdňovanie tabuliek riešiť pomocou štandardného postupu na mazanie jednotlivých pravidiel zavolaného pre všetky pravidlá nachádzajúce sa vo filtry
- **Zavolanie príkazu na povolenie preskladávania položiek v Cockoo Hash tabuľkách**
- realizované zápisom do STOP_REQ s hodnotou nastavenou na nulovú masku všetkých typov pravidiel (obecne 0)
- **Zavolanie príkazu na neignorovanie výsledkov porovnávania v HW**
- realizované zápisom do IGNORE_REQ s hodnotou nastavenou na nulovú masku všetkých typov pravidiel (obecne 0)
- filter je atomicky prepnutý do stavu, kedy je výsledok každého vyhľadávania braný do úvahy a je privádzaný na výstup komponenty
- POZN: výsledkom vyhľadávaní bude stále neúspech, pretože bol filter reálne vyprádnený
- POZN: vypnutie ignorovania je potrebné aby sa prejavili budúce operácie pridávania pravidiel

5 Teoretický rozbor

Výsledky filtru byly publikovány na mezinárodní vědecké konferenci v článku: KEKELY Lukáš, ŽÁDNÍK Martin, MATOUŠEK Jiří a KOŘENEK Jan. Fast Lookup for Dynamic Packet Filtering in FPGA. In: 17th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems. Warszawa: IEEE Computer Society, 2014, s. 219-222. ISBN 978-1-4799-4558-0.

Adresa	Názov	Obsah dát	Popis
0x0000	STOP_REQ	TYPE_MASK	Požiadavok na pozastavenie preskladávania položiek v Cockoo Hash tabuľkách daných typov.
0x0004	IGNORE_REQ	TYPE_MASK	Požiadavok na ignorovanie pravidiel daných typov.
0x0008	ADD_REQ	TYPE_MASK	Požiadavok na vloženie pravidiel daných typov s kľúčom a dátami (identifikátorom) vyplnenými v dátových registroch.
0x000C	REM_REQ	TYPE_MASK	Požiadavok na odstránenie pravidiel daných typov s položkami kľúča zhodnými ako sú vyplnené v dátových registroch.
0x0010	DIRRD_REQ	DIRRECT_ADDR	Požiadavok na priame vyčítanie hodnôt položiek pravidla na danej adrese do dátových registrov.
0x0014	DIRCLR_REQ	DIRRECT_ADDR	Požiadavok na priame zneplatnenie (zrušenie) pravidla na danej adrese.
0x0018	DATA (ID)	n-bitový identifikátor	Dáta (identifikátor) nesené pravidlom, súčasť dátových registrov.
0x001C	PROTOCOL	8b číslo protokolu	Protokol tvoriaci časť pätice kľúča pravidla, súčasť dátových registrov.
0x0020	PORTS	DstPort SrcPort	Čísla portov tvoriace časť pätice kľúča pravidla, súčasť dátových registrov.
0x0024-0x0030	SRCIP	32b slovo IP adresy	Zdrojová IP adresa tvoriaca časť pätice kľúča pravidla, súčasť dátových registrov.
0x0034-0x0040	DSTIP	32b slovo IP adresy	Cieľová IP adresa tvoriaca časť pätice kľúča pravidla, súčasť dátových registrov.
0x0044-	SEEDS	32b semienko	Semienka (inicializačné vektory) pre používané hašovacie funkcie.

Tabulka 1. Adresný priestor pre zápis

Adresa	Názov	Obsah dát	Popis
0x0000	STATUS_DISABLE	TYPE_MASK	Príznaky nepodporovania jednotlivých typov pravidiel.
0x0004	STATUS_IGNORE	TYPE_MASK	Príznaky ignorovania jednotlivých typov pravidiel.
0x0008	STATUS_FULL	TYPE_MASK	Príznaky zaplnenia Cockoo Hash blokov pre jednotlivé typy pravidiel.
0x000C	STATUS_BUSY	TYPE_MASK GLOBAL_BUSY	Príznaky zaneprázdnenosti Cockoo Hash blokov pre jednotlivé typy pravidiel a príznak globálneho zaneprázdnenia pokročilého filtra.
0x0010	RULE_TYPES	32b číslo	Počet podporovaných typov pravidiel, celé číslo v rozsahu <1,30>.
0x0014	IV_NUMBER	32b číslo	Počet používaných 32b semienok pre hašovacie funkcie.
0x0018	DATA (ID)	n-bitový identifikátor	Dáta (identifikátor) nesené pravidlom, súčasť dátových registrov.
0x001C	PROTOCOL	8b číslo protokolu	Protokol tvoriaci časť pätice kľúča pravidla, súčasť dátových registrov.
0x0020	PORTS	DstPort SrcPort	Čísla portov tvoriace časť pätice kľúča pravidla, súčasť dátových registrov.
0x0024-0x0030	SRCIP	32b slovo IP adresy	Zdrojová IP adresa tvoriaca časť pätice kľúča pravidla, súčasť dátových registrov.
0x0034-0x0040	DSTIP	32b slovo IP adresy	Cieľová IP adresa tvoriaca časť pätice kľúča pravidla, súčasť dátových registrov.
0x0044	SEEDS	32b semienko	Semienka (inicializačné vektory) pre používané hašovacie funkcie.
0x0052	RULES_INFO	TYPE_CFG	Podrobnejšie informácie o jednotlivých podporovaných typoch pravidiel.

Tabulka 2. Adresný priestor pre čtení

Parameter	Popis
-h	Výpis nápovedy
-v	Zvýšenie kontrolnich vypisu, (verbose level) programu.
-q	Zákaz všetkých sprievodných výpisov (záporný verbose level)
-x [path]	Cesta k súboru s [[Knihovna_hwio Funkce popisom zariadenia (src)]]
-r	Zapnutie resetovacieho módu príkazov, ktoré ho podporujú (patria medzi ne len -C a -R)
-d [path]	Cesta k súboru [[Knihovna_hwio Funkce zariadení (phys)]]
-A	Príkaz bude prevedený nad všetkými kompatibilnými a dostupnými komponentami.
"PRÍKAZY" -I	Výpis podrobností o pokročilom filtri.
-L	Výpis všetkých dostupných pokročilých filtrov.
-C	Výpis počítadiel paketov a ich prípadné vynulovanie (s parametrom -R).
-D [type;key]	Odobranie pravidiel s daným typom a kľúčom.
-G	Zobrazenie konfigurácie podporovaných typov pravidiel prečítanej z pokročilého filtra.
-R	Prečítanie a výpis všetkých aktuálne platných pravidiel vo filtri a ich prípadné zrušenie (s parametrom -R).
-S	Zobrazenie stavovej informácie pre jednotlivé typy pravidiel filtra.
-T [type;key;data]	Pokus o pridanie špecifikovaného pravidla do filtru.
-V	Zobrazenie aktuálne používaných semienok (inicializačných vektorov) pre hašovacie funkcie haš tabuliek filtra.
-W [seeds]	Nastavenie semienok hašovacích funkcií na novo definované hodnoty (vedie na vyprázdnenie filtra).

Tabulka 3. Parametry nástroje afilterctl