OPTIMALIZACE MONITOROVÁNÍ SÍŤOVÝCH TOKŮ

Martin Žádník
Výpočetní technika a informatika, 2., prezenční
Supervisor: Sekanina Lukáš, doc. Ing., Ph.D.

Fakulta informačních technologií, Vysoké učení technické v Brně
Božetěchova 2, 612 66 Brno

izadnik@fit.vutbr.cz

Abstrakt. Sledování síťového provozu je nedílnou součástí infrastruktury moderní počítačové sítě. Mezi rozšířené techniky patří sběr statistik o IP tocích, jenž se využívá v nejrůznějších typech zařízení, například ve směrovačích, IDS zařízeních a síťových sondách. Disertační práce se věnuje optimalizaci sledování IP toků a tento článek je zaměřen konkrétně na identifikaci a sledování tzv. sloních toků. V rámci tohoto tématu je prezentován nový způsob identifikace sloních toků za použití modifikované LRU správy paměti. Tato metoda dovoluje sledovat vybrané toky již od prvního paketu, což je podstatný rozdíl oproti dosavadním metodám, které umožňují sledovat síťový tok až poté, co počet jeho paketů přesáhne určitou mez. Experimentální výsledky ukazují, že metoda je srovnatelná co do kvality identifikace s ostatními, a zároveň vhodná pro implementaci v programovatelném hradlovém poli (FPGA). To umožní tuto metodu použít i ve vysokorychlostních sítích.

Klíčová slova. Síť, tok, sonda, extrakce, indexace.

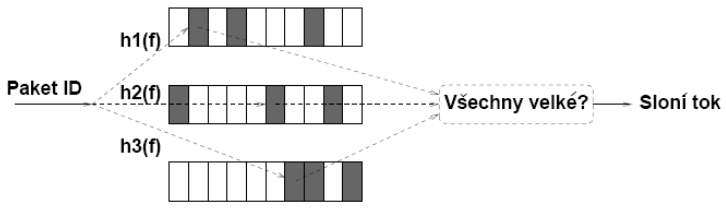
1 Úvod

Sledování sítě na úrovni toků je v dnešní době rozšířeno díky technologii NetFlow [1], která byla představena firmou Cisco koncem devadesátých let. Popularita NetFlow je způsobena vhodnou mírou abstrakce, kdy celkový provoz je rozdělen na toky, a ke každému toku je sledována nejen pětice klíčových údajů (zdrojová a cílová IP adresa, zdrojový a cílový port, číslo protokolu), ale i další statistiky jako je počet paketů a bytů, čas počátku a konce toku, nastavené TCP příznaky a další. Získaná data umožňují daleko lépe dohledávat incidenty na síti nebo ukázat vytížení sítě, ladit nastavení QoS, atd. NetFlow data je možné dále agregovat a vytvářet tak různé pohledy na provoz na síti a získávat důležité informace o chování síťového provozu, např. vyhledávat anomálie [2]. Sledování toků se dále implementuje a využívá v různých síťových zařízeních, například ve směrovačích, kde toto sledování umožňuje rozpoznat komunikující služby a regulovat přidělování šířky pásma. Dále sledování toků můžeme nalézt v IDS systémech a firewallech, kde je nutné vědět v jakém stavu se sledované spojení aktuálně nachází.
Přirozeným jevem současného síťového provozu je obrovské množství toků, které je způsobeno obrovským množstvím dostupných služeb a množstvím uživatelů [3]. Z tohoto důvodu je velmi problematické sledovat všechny toky, které jsou souběžně aktivní. Přechází se proto ke sledování takových toků, které se podílejí nejvíce na objemu přenesených dat, tzv. sloní toky. Studie zabývající se složením síťového provozu ukázaly, že malé procento těchto toků (do desíti procent) je odpovědné za velkou část síťového provozu. Pro většinu aplikací tak bude postačovat sledovat pouze tyto toky a ostatní ignorovat, čímž se významně sníží nároky na kapacitu paměti.

Tato práce navrhuje metodu Single Step Segmented Least Recently Used, S^3–LRU, pro identifikaci a sledování sloních toků. Na rozdíl od běžné LRU dovoluje S^3–LRU uchovávat sloní toky v paměti a chránit je před shluky toků obsahujících pouze několik paketů. Zároveň je možné sledovat sloní toky již od jejich prvního paketu, což je hlavní rozdíl od ostatních metod. Ty jsou založeny na určité formě pravděpodobnostního filtru, kdy je každému toku přiřazeno několik čítačů z velkého pole. Zároveň ale může jeden čítač patřit více tokům. Pokud dojde k překročení určeného prahu všemi čítači daného toku, pak je tok označen a sledován.
V současné době je pro zpracování provozu na vysokorychlostních sítích využívána hardwarová akcelerace, ať již v podobě ASIC řešení nebo s využitím programovatelného hradlového pole, nejčastěji v podobě FPGA. Právě tyto řešení jsou cílovou platformou pro implementaci navrhované S^3–LRU metody, neboť obsahují nebo mohou obsahovat malou ale rychlou paměť s vysoce paralelním přístupem, tzn. je možné během jednoho hodinového taktu přistoupit k více slovům na různých adresách umístěných v disjunktních blocích. V rámci práce je ukázáno, že prezentovanou metodu lze velice efektivně v FPGA implementovat a dosáhnout dobrých výsledků jak z pohledu zabraných zdrojů, tak z pohledu dosažené frekvence.
Tato práce je členěna následovně. Druhá kapitola je věnována současnému stavu v oblasti identifikace sloních toků a jejich sledování. Následuje kapitola přibližující vlastnosti a princip S^3–LRU. Čtvrtá kapitola prezentuje provedené experimenty. Poslední kapitola shrnuje důležité poznatky tohoto článku i mé disertační práce a diskutuje budoucí směřování mé disertační práce.

2 Současný stav řešené problematiky

První z prací v oblasti identifikace a sledování sloních toků byla publikována autory Estan a Varghese [4]. Zde byl definován pojem sloní tok jako tok podílející se na vytížení linky za určité časové období více než *N* procenty. Zároveň Estan a Varghese navrhli dvě metody identifikace a sledování sloních toků. První z nich byla pojmenována Sample and Hold a pracuje následovně. Pokud existuje záznam pro paket v paměti, pak tento paket vždy využijeme pro aktualizaci záznamu o toku. Pokud neexistuje záznam pro paket v paměti, pak ho přijmeme s předem stanovenou pravděpodobností. V případě, že byl paket přijat, založí se záznam v paměti.

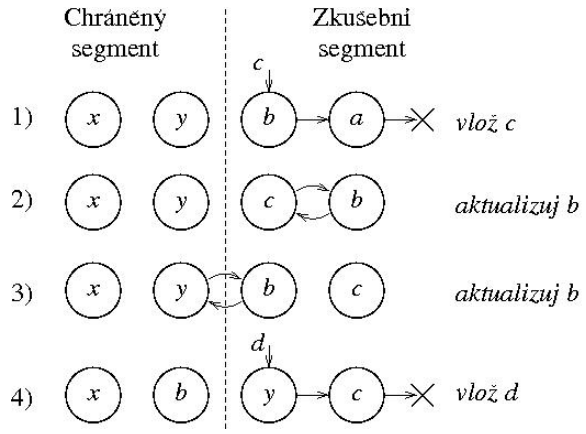


Obrázek 1: Princip metody Multistage Filters pro sledování sloních toků (převzato z [3])

Další navržená heuristika dosahující lepších výsledků je založena na vícenásobném filtru, tzv. Multistage Filters, využívající hash funkcí a pole čítačů (viz. obrázek 1). Paket náležící toku je namapován do několika paralelních polí pomocí několika různých hash funkcí. Každá položka pole obsahuje čítač, který je s příchodem paketu inkrementován. Pokud jsou všechny adresované čítače daného toku nad určeným prahem, pak se jedná o sloní tok. Může nastat i situace, kdy se malý tok namapuje na některé čítače velkého toku. Přidáváním dalších paralelních polí se exponenciálně snižuje pravděpodobnost, že malý tok bude identifikován jako velký, neboť všechny čítače musí překročit daný práh. Na publikaci autorů Estana a Varghese navázali další různými modifikacemi, např.[9]. Žádná z dosavadních metod neumožňuje sledovat sloní toky již od jejich prvního paketu. Přitom právě prvních *N* paketů toku bývá nejdůležitějších, například pro rozpoznání síťové služby [5].

3 S^3—LRU

Metoda S^3—LRU je modifikace metody Segmented LRU, jež byla publikována v [8] a je modifikací LRU. SLRU a LRU jsou ovšem náchylné na expiraci již existujících záznamů během zvýšeného výskytu nových toků. Takový jev je v Internetu běžný, známý jako *Internet background radiation* [3]. S^3—LRU s tímto jevem počítá, mechanismus správy probíhá následovně.



Obrázek 2: Vývoj záznamů v paměti spravovaných pomocí S^3—LRU

Paměť spravovaná pomocí S^3—LRU je rozdělena do dvou segmentů: *zkušební* a *chráněný*, jak je zobrazeno na obrázku 2. Když nastane výpadek záznamu (záznam není k dispozici, protože tok je nový), je záznam o toku přidán na začátek zkušebního segmentu a LRU záznam tohoto segmentu odstraněn (obrázek 2 (varianta 1)). Záznam, na který přichází pakety se dostává zkušebním segmentem na začátek výměnou se sousedním záznamem (obrázek 2 varianta 2). Přesun záznamu do chráněného segmentu způsobí přesun posledního záznamu v chráněném segmentu do zkušebního (obrázek 2 varianta 3). Výše uvedeným způsobem chrání S^3—LRU stávající záznamy před náhlou záplavou nových položek, které nebudou v budoucnu použity, neboť se nedostanou do chráněného segmentu. Velikost chráněného segmentu je volitelná a je diskutována v provedených experimentech.
Frekvence přístupů a čas tak určují pozici záznamu v segmentu a zároveň určují, který záznam má být uvolněn. Přesunem záznamů pouze o jednu pozici způsobí, že pouze sloní toky budou soutěžit o přesun a pozici v chráněném segmentu. To je rozdíl oproti SLRU, jenž při každém přístupu k záznamu přesune záznam na začátek segmentu, takže jsou oba setříděny pomocí LRU. To je nevýhodné z hlediska sloních toků, neboť Jakmile malý tok obsahuje dva po sobě jdoucí pakety, pak je přesunut na začátek chráněného segmentu a trvá dlouho než může být uvolněn.

4 Implementace v FPGA s využitím naivní hash tabulky

Technologie FPGA je pro implementaci a testovaní S^3—LRU velmi vhodná z několika důvodů. Procesory neumožňují ovládat správu paměti a proto přístup k záznamům do paměti bude jejich úzké hrdlo. Procesory tak neumožní škálování měření sloních toků na vysokých rychlostech. ASIC technologie je příliš nákladná a neumožňuje experimentovat s různými variacemi S^3—LRU.
Paměťová architektura Xilinx FPGA [6] je složena z registrů, LUT, a tzv. BlockRAM. Pro sledování toků uvažujeme využití pouze paměti BlockRAM, neboť poskytuje nejvyšší možnou kapacitu dostupné paměti na čipu a zároveň je možné data číst či zapisovat během jednoho hodinového cyklu.
Za účelem efektivní implementace sledování sloních toků je paměť záznamů indexována pomocí naivní hash tabulky. Celá paměť je rozdělena pomocí hash na disjunktní bloky (řádky), které jsou spravovány pomocí S^3—LRU. Vyhledávání v této struktuře pracuje následovně. Nad klíčovými

položky (nejčastěji IP adresy a porty) příchozího paketu je spočítána hash, která vybere blok, a v tomto bloku se sekvenčně vyhledá příslušný záznam. Aby každý záznam nemusel obsahovat všechny pole klíče, je zde uložena sekundární hash, čímž se významně ušetří místo zabrané jedním záznamem. Kolize, které vzniknou tímto schématem je možné libovolně zmenšovat dimenzováním délky sekundární hash. Pravděpodobnost kolize p_f je tedy závislá na počtu bloků 2^h a počtu možných výsledků sekundární hash, tedy 2^b. Konkrétní hodnoty získáme výpočtem rovnice pro tzv. narozeninový problém [7]:

$$p_f = 1 - \frac{m!}{m^n(m-n)!} = 1 - \frac{2^{(h+b)}!}{2^{(h+b)^n}(2^{(h+b)}-n)!}$$

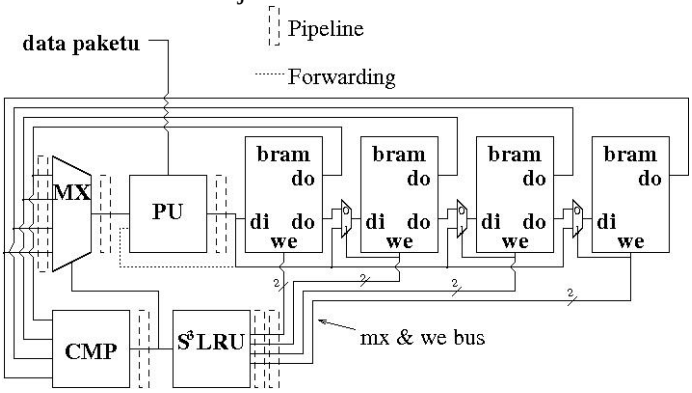
$$\approx 1 - e^{-\frac{(n-1)n}{2\times 2^{(h+b)}}}$$

kde *m* je počet možných hash výsledků první a sekundární hash a *n* je počet záznamů v paměti. Tabulka 1 udává pravděpodobnost pro různé dimenze paměti.

Počet záznamů	Délka hash (h + b)		
	40	44	48
4K	1,90E-06	1,20E-07	7,50E-09
8K	4,10E-05	3,20E-07	2,40E-08
16K	1,20E-04	7,60E-06	4,80E-07
32K	4,90E-04	3,10E-05	1,90E-06

Tabulka 1: Pravděpodobnost záměny dvou toků

BlockRAM paměti tvoří na čipu FPGA pravidelnou strukturu. Každou BlockRAM paměť je možné individuálně adresovat a přistupovat k jejímu obsahu pomocí dvou portů. Slova se stejnou adresou v paměti pak tvoří blok naivní hash tabulky, ve kterém se dohledává, v případě FPGA je možné vyčíst celý blok a v něm paralelně vyhledat odpovídající záznam porovnáním sekundární hash (CMP—comparator). Tento záznam obsahuje kromě hash také statistiky, které u toků sledujeme. Ty jsou aktualizovány (PU—Processing Unit) a záznam je uložen zpět do bloku na nové místo dané S^3—LRU politikou. Schéma implementace v FPGA je znázorněno na obrázku 3.



Obrázek 3: Implementace naivní hash tabulky s S^3—LRU správou záznamů

Zpracování záznamu je vhodné zřetězit, v opačném případě by nebylo možné dosáhnout dostatečně vysoké frekvence 125 MHz až 156 MHz.Ta při šířce sběrnice 64 bitů dovoluje zpracovat až 8--10 Gbps síťový provoz. Řešení lze dále optimalizovat zavedením tzv. forwading, kdy místo zpětného uložení záznamu je přiveden záznam na vstup PU. To je možné v případě, kdy po sobě následují dva a více paketů stejného toku. Řešení je možné dále škálovat zavedením paralelních naivních hash tabulek, kdy první hash rozdělí toky do disjunktních NHT.

5 Experimenty

Sledování sloních toků za pomoci S^3—LRU bylo implementováno na platformě NetFPGA[1], která obsahuje Virtex II Pro FPGA. Naivní hash tabulka s S^3—LRU správou byla napsána v jazyce Verilog a syntetizována pomocí XST 10.1. Samotná infrastruktura na NetFPGA, tj. komunikace s hostitelským počítačem přes PCI, síťové rozhrání a další zabírají polovinu čipu paměti, druhá polovina je k dispozici pro implementaci S^3—LRU. Výsledky syntézy tohoto řešení jsou prezentovány v tabulce 2.

Konfigurace	BRAM	Slice	Frekvence
512 x 32 x 32	33	2394	154 MHz
512 x 32 x 40	65	2560	143 MHz
1024 x 16 x 32	33	1117	154 MHz
1024 x 16 x 40	65	1708	154 MHz

Tabulka 2: Zabrané zdroje a dosažená frekvence

Výše uvedené výsledky jsou prozatím bez zahrnutí NetFPGA infrastruktury. Do frekvence se tak nepromítají logické cesty vzniklé interakcí s okolními moduly. Proto byla NHT s S^3—LRU umístěna do designu implementující funkcionalitu běžné síťové karty (ta zabírá 131(56%) BlockRAM a 16789(71%) Slice). Tento design bez dalších úprav dosahuje frekvence po fyzické syntéze 125 MHz. Do tohoto designu byl vložen zkoumaná S^3—LRU paměť a pozorována dosažená frekvence a čas, za který se design podařilo syntetizovat. Výsledky jsou shrnuty v tabulce 3.

Záznamů v bloku	Délka sekundární hash		
	24	32	40
16	125 (3)	125 (5)	125 (11)
32	125 (39)	125 (96)	116

Tabulka 3: Výsledná frekvence dosažená při fyzické syntéze (závorka uvádí dobu syntézy v minutách)

Výsledky ukazují, že velikost bloku v naivní hash tabulce je limitována na 32 záznamů s 32 bitovou délkou sekundární hash. Kritická cesta omezující frekvenci je v tomto případě tvořena komparátorem (CMP) sloužícím pro vyhledání záznamu.
Z pohledu úspěšnosti detekce sloních toků byly provedeny experimenty na dvou 30 minutových vzorcích síťových dat. Složení dat je uvedeno v tabulce 4.

Vzorek	Trvání	Paketů	Byte	Toků
1	30 min	57M	35G	14,9K
2	25 min	58M	54G	4,4K

Tabulka 4: Charakteristika použitých testovacích dat

Velikost naivní hash tabulky jsme byla dimenzována na 1024 záznamů, každý obsahující 64 bytů, stejně jako v NetFlow [1]. Následně bylo pozorováno jakou úspěšnost S^3—LRU dosahuje v identifikaci sloních toků, které zabírají více jak 0,1% kapacity linky, 0,1% až 0,01% a 0,01% až 0,001% (viz. tabulka 5).

[1] www.netfpga.org

Algoritmus	Skupina		
	> 0,1%	0,1 až 0,01%	0,01 až 0,001%
S3-LRU	0,14	0,24	0,88
SLRU	17,11	10,21	35,51
LRU	23,53	12,61	41,72

Tabulka 5: Porovnání úspěšnosti detekce sloních toků (vyjádřeno v procentu neidentifikovaných)

V porovnání s běžnými algoritmy pro správu cache dosahuje S^3—LRU lepších výsledků. Zároveň dosahuje srovnatelných výsledků s doposud publikovanými algoritmy, bohužel přesné porovnání není možné díky odlišným testovacím sadám, které nejsou veřejně k dispozici. Oproti dosavadním algoritmům S^3—LRU umožňuje sledovat sloní toky již od prvního paketu.

6 Závěr

Při zpracování síťového provozu je většinou vyžadováno uchování stavu pro každý tok To vede k použití velkých externích pamětí, aby bylo možné uchovat veškeré záznamy o všech tocích. Externí paměti jsou ovšem pomalé a pro vysokorychlostní linky představují úzké hrdlo zpracování provozu. Navíc je známo, že až padesát procent toků obsahuje pouze jeden paket, a proto není potřeba udržovat k takovém toku jakoukoliv informaci.
Pro mnoho aplikací je postačující, pokud stavová informace bude udržována jen pro sloní toky. Z tohoto důvodu byla navržena implementace sledování sloních toků v interní paměti FPGA. Tato implementace využívá S^3—LRU správu paměti, která poskytuje lepší výsledky než běžné LRU, či SLRU. Oproti dosavadním algoritmům pro identifikaci sloních toků poskytuje navržené S^3—LRU sledování toků od jejich prvního paketu, což je v mnoha aplikacích nezbytné.
Budoucí směřování mé disertační práce má za cíl analyzovat vlastnosti navržené S^3—LRU více do detailu. Důraz bude kladen na různé modifikace této politiky za účelem zvýšení schopnosti identifikace sloních toků a snížení paměťové režie. V rámci disertační práce se budu zabývat obecně identifikací silných/význačných toků pomocí odlišných metod tak, aby vznikl ucelený soubor experimentů Na jeho základě bude možné optimalizovat daný typ sledování toků. Například přístup vhodný pro FPGA nemusí být vhodný na síťových procesorech, kde je architektura a přístup k paměti naprosto odlišný.

Reference

[1] B. Claise. Cisco systems netflow services export version 9. RFC(Informational) 3954, IETF, 2004.

[2] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment, New York, NY, USA, 2002. ACM Press.

[3] K. Lan and J. Heidemann. A measurement study of correlations of internet flow characteristics. Computer Networks, 50(1):46 – 62, 2006.

[4] C. Estan and G. Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. ACM Trans. Comput. Syst., 21(3):270–313, 2003.

[5] W. Li et al., “Efficient application identification and the temporal and spatial stability of classification schema,” Computer Networks, vol. 53(6): 790–809, 2009.

[6] Xilinx white paper: Virtex 5 Multiplatform FPGA. http://www.xilinx.com/products/virtex5.

[7] D. Wagner, A generalized birthday problem, in CRYPTO ’02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology. London, UK: Springer-Verlag, 2002, pp. 288–303.

[8] R. Karedla et al., Caching strategies to improve disk sys. performance, Computer, vol. 27(3): 38–46, 1994.

[9] J. Li, C. Hu, and B. Liu. Monitoring large flows in network. Network and Com. Systems, vol. 464, 2005.