RELIABILITY MEASUREMENT OF THE PATTERN MATCHING

Milan Dvořák
Master Degree Programme (2), FIT BUT
E-mail: xdvora66@stud.fit.vutbr.cz

Supervised by: Jan Kaštil
E-mail: ikastil@fit.vutbr.cz

Abstract: This paper deals with the pattern matching methods based on finite automata and describes their optimizations with possible faults in matching. We propose a methodology for the measurement of reliability of the pattern matching, by comparing it with the results of the PCRE library. Experimental results for DFA with perfect hashing and faulty transition table are discussed.

Keywords: regular expression, finite automaton, pattern matching, PCRE, perfect hashing, reliability

1 ÚVOD

Vyhledávání řetězců specifikovaných pomocí regulárních výrazů nalézá uplatnění v mnoha oblastech informačních technologií. Jednou z nejvýznamnějších je dnes kontrola síťového provozu. V posledních letech došlo k dynamickému vývoji síťových technologií, výrazně vzrostl objem přenášených dat i počet nabízených služeb. S tím ovšem přišel i nárůst škodlivého softwaru, cílených útoků a dalších bezpečnostních hrozeb. Proto se začaly využívat systémy IDS/IPS, které prohledávají obsah paketu a filtrují nežádoucí data.

Hlavním problémem v této oblasti je rychlost samotného prohledávání paketů. Systémy musejí v reálném čase monitorovat linky o rychlostech dosahujících několika gigabitů za sekundu. Byly navrženy různé metody a postupy, jak vyhledávání urychlit. Některé optimalizace ovšem přináší chybovost ve vyhledávání. Tato práce se proto zabývá ohodnocováním spolehlivosti vyhledávacích algoritmů.

2 VYHLEDÁVÁNÍ ŘETĚZCŮ SPECIFIKOVANÝCH REGULÁRNÍMY VÝRAZY

Vyhledávání pomocí regulárních výrazů se obvykle realizuje jejich převodem na konečný automat. Podle toho, zda se použije deterministický nebo nedeterministický automat, hovoříme o přístupech DKA nebo NKA.

U DKA je deterministická přechodová funkce, v každém stavu je tedy stav následující jednoznačně určen na základě čteného symbolu. Z toho vyplývá, že algoritmus pracuje s lineární časovou složitostí vzhledem k délce vstupního řetězce. Při realizaci v hardwaru je přechodová tabulka uložena v paměti. Díky tomu je rekonfigurace jednotky pro upravenou množinu regulárních výrazů jednoduchá, stačí nahrát nový obsah tabulky. Hlavním problémem DKA je paměťová náročnost. Při determinizaci automatu totiž může dojít až k exponenciálnímu nárůstu počtu stavů a tím i velikosti přechodové tabulky.

V NKA může v každém kroku existovat několik možností, jak dále pokračovat. V softwarové implementaci je toto obvykle řešeno pomocí zásobníku a metody zpětného vyhledávání (*backtracking*). V hardwaru se využívá paralelismus, kdy může být současně aktivováno několik stavů. Lze tak dosáhnout lineární časové složitosti. Přechodová tabulka je ovšem realizována logickým obvodem a rekonfigurace jednotky je složitější a časově náročnější.

Pro zrychlení základních přístupů byla navržena řada optimalizací. V případě DKA je cílem redukce velikosti přechodové tabulky (např. DKA s perfektním hašováním [1]), u NKA se řeší efektivní ma-

pování do hardwaru a redukce využití zdrojů na čipu. Zajímavou myšlenkou je kombinace výhod NKA a DKA v takzvaném hybridním přístupu, neboli Hybrid-FA [2].

2.1 CHYBY VE VYHLEDÁVÁNÍ

Některé ze zmíněných optimalizací algoritmů mohou způsobovat chybovost ve vyhledávání. Příkladem explicitní chybovosti je algoritmus [1]. Přechodová tabulka je zde indexována pomocí perfektní hašovací funkce, která pro známé klíče nevytváří kolize. Pro neznámé klíče (neexistující přechody) ovšem kolize vzniknout může. Proto v tabulce musí být uloženy hodnoty platných klíčů pro ověření validity přechodů. To je však náročné na paměť. Zavedením chybující přechodové tabulky lze paměťové nároky snížit. Pro ověření kolizí způsobených neexistujícími přechody se pak nepoužívá celý klíč do tabulky, ale pouze jeho haš. Pokud ovšem dojde ke kolizi hašovací funkce mezi neexistujícím a správným přechodem na daném místě přechodové tabulky, automat tento chybný přechod provede. To se pak může (ale nemusí) projevit chybou ve výsledku vyhledávání.

Další chyby může zapříčinit vyhledávání po paketech. Pokud se totiž bude hledaný řetězec nacházet na rozhraní dvou paketů, nebude nalezen. Řešením je vyhledávat po tocích, kdy je ovšem nutné ukládat stav automatu mezi jednotlivými pakety každého toku. To je problematické zejména u přístupů s NKA, kdy může mít automat několik aktivních stavů.

Implicitní chybovost obsahuje i zmíněný Hybrid-FA. Ten se dělí na rychlou (DKA) a pomalou (NKA) část. Pokud se rychlý automat dostane do stavu, který by způsobil expanzi stavů, předá zbytek paketu pomalé části a sám pokračuje dále. Problém nastává, pokud dojde k aktivaci pomalé části dalším paketem dříve, než bylo dokončeno prohledávání předchozího. Pak totiž nelze výsledek vyhledávání správně přiřadit ke konkrétnímu paketu.

3 METODIKA PRO OHODNOCOVÁNÍ SPOLEHLIVOSTI

V předchozí kapitole bylo vysvětleno, jak mohou některé algoritmy způsobit chybu ve vyhledávání. Pravděpodobnost této chyby se obtížně určuje v teoretické rovině. Proto jsem navrhl metodiku ohodnocování spolehlivosti vyhledávání na základě měření na reálných datech.

Vzniklé chyby mohou být dvou typů, *false positive* a *false negative*. Oba typy mohou mít různou závažnost a jiný dopad na funkčnost zařízení, proto je vhodné tyto chyby odlišovat. Jako samotné ohodnocení algoritmu jsem pak navrhl procentuální vyjádření počtu chyb *false positive* na počet správných negativních vyhodnocení vyhledávání a procentuální vyjádření počtu chyb *false negative* na počet správných pozitivních vyhodnocení vyhledávání. To vyjadřuje pravděpodobnost, s jakou se algoritmus dopustí chyby. Spolehlivost algoritmu získáme jako pravděpodobnost opačného jevu, tedy odečtením hodnoty od jedné.

Správný výsledek vyhledávání je určen pomocí referenčního modelu, jako který jsem zvolil knihovnu PCRE. Tato knihovna by měla být dobře otestována, protože je využívána v řadě aplikací, mezi které patří např. Apache, PHP nebo KDE. PCRE se navíc používá i pro detekci nebezpečného síťového provozu v systému Snort. Pro realizaci vyhledávání pomocí knihovny PCRE jsem vybral program *pcregrep*. Ten jsem musel mírně upravit, aby podporoval regulární výrazy v PCRE notaci včetně parametrů.

Samotná měření jsem prováděl nad daty zachycenými z reálného síťového provozu. Ty jsem rozdělil na soubory podle prohledávaných jednotek (pakety, toky) a pomocí programu *pcregrep* zjistil, ve kterých souborech nastala shoda. Pro realizaci testovaného algoritmu (např. DKA s perfektním hašováním) jsem použil knihovnu Netbench [4], která implementuje řadu přístupů a obsahuje i podpůrné funkce. Ve vytvořeném nástroji pak probíhá porovnávání výsledků, výpočet statistik a určení výsledného ohodnocení.

4 VÝSLEDKY MĚŘENÍ

Provedl jsem experimentální měření spolehlivosti DKA s perfektním hašováním a chybující přechodovou tabulkou [1]. Byly vyhledávány regulární výrazy z L7 filtru [3] nad daty zachycenými v malé domácí síti. Automat byl rozšířen tak, aby každým přechodem přijímal dva symboly. Tabulka 1 ukazuje závislost spolehlivosti algoritmu na počtu bitů, které reprezentují přechod v chybující tabulce. Zobrazené chyby jsou pouze typu *false positive*, chyby *false negative* se nevyskytly. Měření jsem opakoval 25krát a výslednou pravděpodobnost jsem zprůměroval. Při každém měření byly náhodně generovány nové hašovací funkce.

Počet bitů	1	2	3	4	5	6	7	8	9	10
Počet chyb	160696	78256	52921	11072	5243	3673	113	9799	323	0
Spolehlivost [%]	57,7	79,4	86,0	97,1	98,6	99,0	99,9	97,4	99,9	100

Tabulka 1: Závislost spolehlivosti algoritmu na počtu bitů, které reprezentují přechod

V tabulce 1 je vidět, že počet chyb s každým přibývajícím bitem klesá přibližně na polovinu. Při velikosti 10 bitů a více se již chyby nevyskytovaly. Anomálii můžeme pozorovat u 8 bitů, kde počet chyb výrazně narostl a spolehlivost klesla o 2,5 %. V jednom z měření pro 8 bitů totiž vznikla kolize pro chybný přechod z počátečního stavu, který byl velmi často využíván. Jediná kolize tak způsobila téměř 100 000 špatných výsledků vyhledávání. Nalezl jsem tedy problém u tohoto algoritmu. Možným řešením by mohlo být oddělení validace přechodů pro frekventované stavy (počáteční stav), aby pro ně nenastávaly kolize.

5 ZÁVĚR

V tomto článku jsem popsal problematiku vyhledávání vzorů specifikovaných pomocí regulárních výrazů. Dále jsem rozebral, jak některé postupy mohou způsobit chybovost ve vyhledávání, i když to explicitně neuvádějí (např. zpracování po paketech). Proto jsem navrhl a implementoval nástroj, který dokáže ohodnocovat a srovnávat spolehlivosti různých přístupů a určit pravděpodobnost vzniku chyby. Dále jsem prezentoval výsledky měření u chybujícího DKA s perfektním hašováním [1] pro různé velikosti reprezentace přechodu. Díky měření byl odhalen možný nedostatek v návrhu algoritmu a tedy i prostor pro jeho vylepšení.

Jako další pokračování práce se nabízí srovnání jiných chybujících algoritmů a zkoumání vlivu různých parametrů, vstupních dat a regulárních výrazů na výsledky jejich vyhledávání.

REFERENCE

[1] Kaštil, J., Kořenek, J.: High Speed Pattern Matching Algorithm Based on Deterministic Finite Automata with Faulty Transition Table. In: Proceedings of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems. La Jolla, US: ACM, 2010, s. 2.

[2] Becchi, M., Crowley, P.: A hybrid finite automaton for practical deep packet inspection. In: Proceedings of the International Conference on emerging Networking EXperiments and Technologies (CoNEXT). New York, NY, USA: ACM, 2007. ISBN: 978-1-59593-770-4.

[3] Application Layer Packet Classifier for Linux. CLEARFOUNDATION. [online]. [cit. 2012-03-03]. Dostupné z: `http://l7-filter.clearfoundation.com/`

[4] Puš, V., Tobola, J., Košar, V., aj.: Netbench: Framework for Evaluation of Packet Processing Algorithms. In: Symposium On Architecture For Networking And Communications Systems. Los Alamitos, CA, USA: IEEE Computer Society, 2011, s. 95-96. ISBN 978-0-7695-4521-9.