DETERMINISTIC FINITE AUTOMATON FOR PATTERN MATCHING IN HIGH SPEED NETWORKS

Jan Kaštil
Magister Degree Programme, FIT BUT
E-mail: xkasti00@stud.fit.vutbr.cz

Supervised by: Jan Kořenek
E-mail: korenek@fit.vutbr.cz

ABSTRACT

This paper deals with pattern matching in high speed networks using Deterministic Finite Automaton. We propose new method how to reduce transition table size in multichar automaton. The method is based on shared decoder, which is used to transform input alphabet to alphabet with reduced number of symbols. According to our preliminary results, the transition table can be significantly reduced.

1 ÚVOD

S rostoucím počtem lidí připojených na internet i se vzrůstající kvalitou jejich připojení se zvedají počty útoků na počítačové sítě. Je proto nutné, aby odpovídající pozornost byla věnovaná výzkumu v oblasti obrany proti útokům. Jedním z možných způsobů ochrany jsou systémy Indtrusin Detection Prevention Systems (IDPS) [2]. Tyto systémy bývají často založeny na prohledávání celého síťového provozu s cílem objevit vzory, které signalizují útok. Útoky bývají popsány pomocí řetězců nebo regulárních výrazů. V současnosti je nejrozšířenějším IDS systémem volně dostupný program Snort, který dosahuje rychlostí 100 - 200 Mb/s, což je nedostačující pro moderní 10Gb/s sítě. Zrychlení tohoto řádu lze docílit pouze hardwarovou akcelerací. Tento článek představuje metodu pro konstrukci deterministického konečného automatu, který s využitím technologie FPGA je schopen analyzovat síťový provoz i na takovýchto rychlostech.

2 ROZBOR

V současnosti existuje mnoho rychlých metod pro vyhledávání řetězců. Příkladem může být Boyer-Moorova metoda [1], která má v ideálním případě časovou složitost O(m/n), kde m je délka prohledávaného řetězce a n je délka hledaného řetězce. Bohužel nejhorší možná časová složitost této metody je O(3n). Při použití v Intrusion Detection System (IDS) je třeba předpokládat nejhorší možný případ. Útočník totiž vždy může cíleně zkonstruovat takový síťový provoz, aby náš systém vyřadil z provozu. Další podstatnou výhodou konečných automatů je schopnost vyhledávat také regulární výrazy.

Konečné automaty dělíme na deterministické a nedeterministické. V další práci se budu zabývat pouze deterministickými automaty. Tyto automaty obsahují pouze jednu funkční jednotku s

pamětí, která obsahuje celou přechodovou tabulku automatu. Systém je tedy omezen velikostí paměti, ale je možné operativně měnit množinu vyhledávaných vzorů.

V IDS chceme vyhledávat mnoho vzorů zaráz. Počet vyhledávaných vzorů nemá vliv na rychlost, ale pouze na složitost automatu, která se projevuje velikostí přechodové tabulky. Deterministický automat vyhledávající všechny potřebné vzory je často příliš velký i pro softwarová řešení. Jedním z možných řešení je rozdělit množinu vzorů na několik podmnožin a pro každou podmnožinu vytvořit samostatnou vyhledávací jednotku. V [4] je představena metoda dělení množiny pravidel na několik podmnožin o stejné velikosti nepřekračující hodnotu zadaného limitu. Bohužel nelze rozdělit množinu pravidel bez znalosti všech úprav, které budeme s výslednými automaty provádět, což činí diskutovnou metodu extrémně časově náročnou.

Současné hardwarové architektury operují na frekvenci přibližně 100MHz. Pokud použijeme klasický minimální konečný automat, dosáhneme propustnosti 800Mb/s, což je však pro současné 10Gb/s sítě zcela nedostačující. Proto je třeba přistoupit k optimalizacím automatu. Pro dosažení kvalitních výsledků je možno využít více metod. Za zmínku stojí metoda Delayed Input DFA [3], která drasticky snižuje velikost přechodové tabulky za cenu snížení propustnosti. Zároveň implementuje několik desítek vyhledávacích jednotek, které pracují paralelně.

Tato práce se zabývá víceznakovými automaty. Pro potřeby snížení velikosti jejich přechodové tabulky představuje úpravu definující symbol jako množinu znaků.

2.1 PŘIJÍMÁNÍ VÍCE ZNAKŮ V JEDNOM TAKTU

Základní metodou zvýšení propustnosti je rozšíření počtu přijímaných znaků na vstupu automatu. S každým přijímaným znakem se zvedne propustnost o dalších 800Mb/s. Tedy pro 10 znaků v jednom taktu bude propustnost 8000Mb/s a pro 14 znaků překročí propustnost hranici 10Gb/s. Nevýhodou této metody je skutečnost, že pokud z jednoho stavu existuje více výstupních přechodů, tak po rozšíření na k znakový automat z tohoto stavu bude vycházet k krát více přechodů. Tím může docházet k prudkému zvyšování počtu přechodů, což znamená větší spotřebu paměti pro tabulku přechodů. Protože zdroje v FPGA jsou velmi omezené, je třeba použít metodu pro snížení velikosti tabulky přechodů.

2.2 REPREZENTACE PŘECHODŮ POMOCÍ MNOŽIN

Analýzou pravidel programu Snort se podařilo zjistit, že v automatu existuje značné množství přechodů, které mají stejný počáteční i koncový stav a liší se pouze symbolem, při němž se provedou. Bez újmy na obecnosti lze všechny tyto přechody pro danou dvojici stavů nahradit jediným přechodem, jehož symbol bude reprezentován množinou. Takovýto přechod nahradí několik záznamů v původní tabulce přechodů a tím dojde k úspoře místa. Přechod se provede, pokud načtený symbol patří do množiny. V ideálním případě by tímto způsobem bylo dosaženo značné úspory velikosti tabulky přechodů. Při každém přechodu by však bylo třeba ověřovat přítomnost prvku v množině, což by vyžadovalo uchovávat pro každý stav informaci o jeho množinách. Vlastní úspora místa by byla tedy nulová. Na abecedu automatu proto klademe ještě další doplňující podmínku. Požadujeme, aby bylo možno určit příslušnost do množiny bez znalosti aktivního stavu v automatu. To lze, pokud pro každou n-tici znaků na vstupu existuje pouze jedna množina v celé abecedě automatu. Pokud je tato podmínka splněna, nazýváme abecedu deterministickou.

Formálně pak definujeme symbol A jako uspořádanou n-tici množin znaků. A_i pak značí mno-

žinu na pozici i v symbolu A. $Delka(A)$ je pak definována jako počet množin v symbolu. Průnik symbolů definujeme jako

$$(P = A \cap B) \Leftrightarrow (Delka(A) = Delka(B)) \wedge \forall (i < Delka(A)) : P_i = A_i \cap B_i \wedge P_i \neq \emptyset)$$

Abecedu automatu pak definujeme jako množinu symbolů. Pro determinitickou abecedu M navíc platí:

$$\forall x \in M, \forall y \in M : \text{ pokud } y \neq x, \text{ pak neexistuje } P = x \cap y.$$

3 ZÁVĚR

Tato práce se zabývá tvorbou algoritmů vhodných pro hardwarovou akceleraci vyhledávání vzorů v sítích. Představuje se zde koncept víceznakového deterministického konečného automatu, jehož abeceda symbolů je tvořena množinami. Tím je umožněno zvýšení propustnosti z 100-200Mb/s na 10Gb/s při zpracování 14 znaků v jednom hodinovém cyklu. Byly zkoumány i jiné alternativy k množinové reprezentaci, jako jsou například intervaly a jejich vliv na úsporu místa v tabulce přechodů.

Přínosem práce je myšlenka použití množin při reprezentaci přechodů ve víceznakovém automatu pro detekce vzorů v provozu vysokorychlostních sítí. Tato myšlenka umožní vytvoření sdíleného dekodéru, který bude převádět n-tice znaků na jediný znak vstupující do rychlých programovatelných hardwarových jednotek pro vyhledávání vzorů. Tento přístup umožní snížení použitých prostředků na čipu při zachování škálovatelnosti řešení. Myšlenka byla formálně popsána a následně implementována v jazyce C++.

Představená myšlenka umožňuje aplikaci dalších metod pro zefektivnění konečných automatů, což může vést k další úspoře prostředků na čipu nebo ke zvýšení propustnosti.

PODĚKOVÁNÍ

Tento příspěvek vznikl za podpory výzkumného záměru MSM6383917201 v rámci výzkumné aktivity Programovatelný hardware sdružení CESNET z.s.p.o.

REFERENCE

[1] Boyer Robert S., Moore Strother J.: A Fast String Searching Algorithm, Communication of ACM, October 1977

[2] Scarfone K., Mell P.: Guide to Intrusion Detection and Prevention Systems, Recommendation of the National Institute of Standarts and Technology

[3] Sailesh Kumar, Sarang Dharmapurikar, Fang Yu, Patrick Crowley, Jonathan Turner: Algorithms to Accelerate Multiple Regular Expressions Matching for Deep Packet Inspection,SIGCOMM'06, September 11-15, 2006, Pisa, Italy.

[4] Fang Yu, Zhifeng Chen, Yanlei Diao, T. V. Lakshman, Randy H. Katz : Fast and Memory-Efficient Regular Expression Matching for Deep Packet Inspection