A Regular Expression Matching Approach to Distributed Wireless Network Security System

Jie Wang, Yanshuo Yu, and Kuanjiu Zhou
(Corresponding author: Yanshuo Yu)

School of Software Technology & Dalian University of Technology
Economy and Technology Development Area, Dalian City 116620, P.R.China
(Email: yuyanshuo@gmail.com)
(Received Aug. 6, 2013; revised and accepted Mar. 13, 2014)

Abstract

There is a growing demand for wireless ad hoc network systems in examining the content of data packages in order to improve network security and application service. Whereas, each distributed wireless node has limited memory and computing power. Since regular expressions offer superior expression power and flexibility, taking advantage of distributed nodes and regular expression collaboratively can be a new perspective for wireless network security strategy. In this paper, a regular expression matching approach is introduced for distributed wireless network security system called DREM (Distributed Regular Expression Matching), which divides the matching into two stages: prefiltering stage and verifying stage. Intensive experiments were conducted on Snort and L7-Filter regular expression data sets to verify the system. The experimental results show that our strategy can speed up the efficiency to 1.7 times faster than conventional approaches for wireless security systems. It is also proved by emulation that our approach can be regarded as a firewall system and well applied in medium or large scale distributed wireless network systems.

Keywords: DREM, regular expression, wireless ad hoc

1 Introduction

An ad hoc network is a collection of wireless nodes dynamically forming a temporary network without the use of any existing network infrastructure or centralized administration. In such a network, each mobile node operates not only as a host but also as a router, forwarding packets for other mobile nodes in the network that may not be within direct wireless transmission range of each other. Some example of the possible uses of ad hoc networking include business associates sharing information during a meeting, soldiers relaying information for situational awareness on battlefield, and emergency disaster relief personnel coordinating effort after a hurricane or earthquake [9, 19]. Ad hoc network is much more vulnerable to malicious exploits than a wired network. Most of current studies focus on how to provide authentication, confidentiality, integrity, nonrepudiation and access control to ad hoc [4, 5, 6, 13]. Distributed authentication is a very common method to solve the ad hoc security problem [1]. However, the high level secure ad hoc approach is specified to a certain situation [17], and a common method is lacked.

On the one hand, since regular expressions offer superior expression power and flexibility, they have been widely used in a variety of network and security applications, such as anti-virus scanners, network intrusion detection and prevention systems [16], firewalls, and traffic classification and monitoring [10].

On the other hand, deterministic finite automata (DFA) and non-deterministic finite automata (NFA) representations are typically used to implement regular expressions. However they require either enormous memory or unacceptable time consumption, rendering them unsuitable for wireless nodes with limited memory and computation power.

In this paper, a regular expression matching approach is proposed for distributed wireless network security systems to take advantage of distributed nodes collaboratively. The matching is divided into prefiltering stage and verifying stage. Given a regular expression set R, another set R' is constructed so that any unmatched item of R' is also an unmatched item of R, but R' is much smaller than R. Any item that matches R' may be an unmatched item of R so that the verifying stage is needed to obtain the exact result. Thus, R is divided into small groups $R_1, R_2, , R_n$ according to their correlation coefficients at the prefiltering stage. Then these groups are used to accomplish the exact matching at the verifying stage.

There are three key contributions. First, we introduce an efficient method to generate the correlation sequence of sub expression print in the prefiltering stage. Second, we propose the Correlation Sets Partition (CSP) algorithm to cluster regular expressions into correlation sets and place them to distributed nodes. Third, we imple-

ment our approach to each wireless node to calculate the relevancy of the package and decide whether to conduct accurate matching.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 explains the generation and implementation of correlation sequence and correlation sets. In Section 4, we present experimental results. Finally, conclusion and future work are given in Section 5.

2 Related Work

As DFA is the preferred representation of regular expression matching, recent work has focused on reducing the huge memory usage of DFA-based regular expression matching. However they achieve memory reduction only for signature sets of simple or specific regular expression. None of them can achieve high-speed regular expression matching for real-world signature sets that contain thousands of complex regular expression. Meiners et al. [14] proposed a well-designed Ternary Content Addressable Memory (TCAM)-based regular expression matching solution that introduces three novel techniques (transition sharing, table consolidation, and variable striding) to reduce TCAM space and improve matching speed. A Minimize then Union framework for constructing compact alternative automata focusing on the D2FA (Delayed DFA) was proposed by Liu et al. This algorithm runs up to 302 times faster and decreases 1390 times of memory consumption than previous algorithms [11].

TCAMs are off-the-shelf chips and have been widely deployed in modern networking devices. Whereas, tables of DFA and NFA are too big to be stored in TCAMs even if we employ techniques such as D2FA [7, 11]. In 2012, Liu et al. [12]presented the RegexFilter, a high-speed and memory-efficient technique. They attempted to speedup regular expression matching by quickly searching these regular expressions that may match each arriving item as little as possible. However, this method only focuses on the prefiltering stage and has nothing to do with the verifying stage. Therefore, we proposed an approach based on a two-stage matching method and the main idea is as follows: first, the filtering stage performs high-speed regular expression print matching on each arriving item. If one regular expression print is matched, the corresponding regular expression will be checked in the verifying stage.

2.1 Prefiltering Stage

Given a regular expression set R, we want to construct another set R' so that any unmatched item of R' is also an unmatched item of R. An unmatched item of a regular expression set is an item that does not match any regular expression in the set [12]. Given an item i, we first match it against R' to get set $O(R', i)$. If $O(R', i)$ is empty, it does not obviously match any member in R and therefore we can skip this item safely; otherwise, we continue to match it against $T(R, O(R', i))$, where

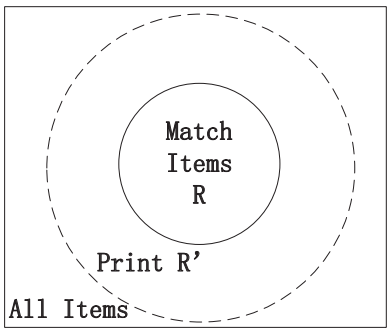

Figure 1: Relationship of prefilterging stage set and matches item

$O(R, i) \subseteq T(R, O(R', i)) \subseteq R$.

The relationship between print (R') and match items is shown in Figure 1. Because most items are unmatched and the match cost of R' is much less than that of R, the overall throughput of this approach is much higher than directly matching against R.

On the one hand, we want the regular expression print to filter out as many unmatched items as possible. On the other hand, we desire matching efficiency of the regular expression print as high as possible. These two goals are conflicting. Therefore two variables called $ES(r)$ (Expression Size) and $MP(r)$ (Matching Probability) are introduced. $ES(r)$ is an estimation variable to quantitatively measure the complexity of regular expression print and $MP(r)$ is the filtering effectiveness of regular expression print. The completely definitions of $ES(r)$ and $MP(r)$ are given as follows.

Definition 1. *Given a Regular Expression r, its Expression Size, denoted by $ES(r)$, is the number of strings represented by r, namely $ES(r) = |S(r)|$.*

$$\begin{cases} \text{if } r = \varepsilon, ES(r) = r \\ \text{if } r = \alpha(\alpha \in \sum), ES(r) = 1 \\ \text{if } r = (r_1), ES(r) = ES(r_1) \\ \text{if } r = r_1 \cdot r_2, ES(r) = ES(r_1) \times ES(r_2) \\ \text{if } r = r_1 | r_2, ES(r) = ES(r_1) + ES(r_2) \\ \text{if } r = r_1^*, ES(r) = \sum_{t=0}^{\infty} ES(r_1)^t = \infty \end{cases}$$

Definition 2. *Given a Regular Expression r, the Minimum Expression Length of r, denoted by $L(r)$, is the number of characters in s, where s is the shortest string in set $S(r)$.*

$$\begin{cases} \text{if } r = \varepsilon, L(r) = 0 \\ \text{if } r = \alpha(\alpha \in \sum), L(r) = 1 \\ \text{if } r = (r_1), L(r) = L(r_1) \\ \text{if } r = r_1 \cdot r_2, L(r) = L(r_1) + L(r_2) \\ \text{if } r = r_1 | r_2, L(r) = min(L(r_1), L(r_2)) \\ \text{if } r = r_1^*, L(r) = 0 \end{cases}$$

Definition 3. *Given a Regular Expression r, the Shortest Expression Size of r, denoted by $SES(r)$, is the number of strings of length $L(r)$ in set $S(r)$.*

$$
\begin{cases}
if\ r = \varepsilon, SES(r) = 1 \\
if\ r = \alpha(\alpha \in \sum), SES(r) = 1 \\
if\ r = (r_1), SES(r) = SES(r_1) \\
if\ r = r_1 \cdot r_2, SES(r) = SES(r_1) \times SES(r_2) \\
if\ r = r_1 | r_2 : \\
\Rightarrow if\ L(r_1) is\ bigger\ than\ L(r_2), SES(r) = SES(r_2) \\
\Rightarrow if\ L(r_1) is\ smaller\ than\ L(r_2), SES(r) = SES(r_1) \\
\Rightarrow if\ L(r_1) is\ equal\ to\ L(r_2), SES(r) = SES(r_2) \\
\quad + SES(r_2) \\
if\ r = r_1^*, SES(r) = 1
\end{cases}
$$

Definition 4. *The matching probability of a Regular Expression r, denoted by $MP(r)$, is dened as the ratio of $SES(r)$ to $SCS(r)$, where $SCS(r)$ represents the total number of strings of length $L(r)$ over alphabet $\sum$.*

$$
\begin{aligned}
MP(r) =& \frac{SES(r)}{|\sum|^{L(r)}} = \frac{SES(r_1) \times SES(r_2)}{|\sum|^{L(r_1)+L(r_2)}} \\
=& (r_1) \times MP(r_2) \leqslant MP(r_1)
\end{aligned}
$$

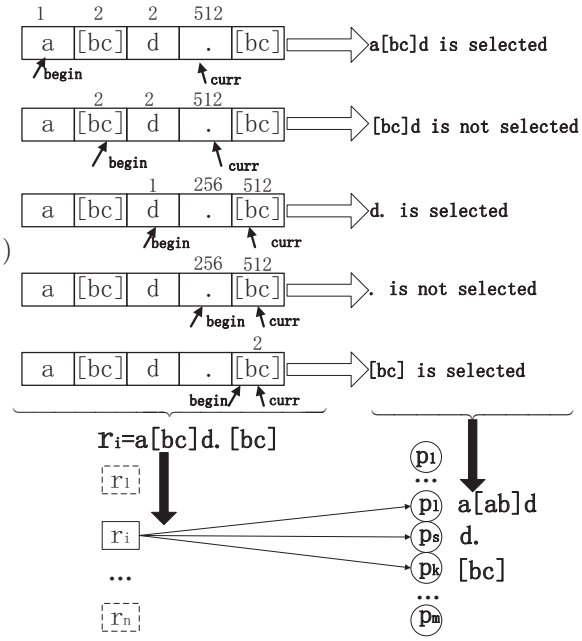

Figure 2: The process of generating print p_i from r_i

2.2 Verifying Stage

In the verifying stage, we mainly focus on how to build correlation from prefiltering print and reduce the memory cost of DFA tables.

A DFA is a 5-tuple $(Q, \sum, \delta, q_0, A)$ where Q is a set of states, $\sum$ is an alphabet, $\delta : \sum \times Q \rightarrow Q$ is the transition function, Q_0 is the start state, and $A \subseteq Q$ is a set of accepting states. The fundamental issue with DFA-based algorithms is the large amount of memory required to store the transition table δ, which is the major part we should deal with.

Prior regular expression matching algorithms are either software-based [2, 3, 8] or FPGA-based [15, 18, 20]. TCAM-based solutions have the advantages of easy encoding and high parallelism [11]. Liu et al. proposed three novel techniques: transition sharing, table consolidation, and variable striding to reduce TCAM space and improve matching speed.

In this paper, we try to use the correlation from prefiltering print to divide the regular expression set R into small groups. Then we adopt the state transition compressing method to improve verifying efficiency.

3 DREM Approach

3.1 Generating Correlation Sequence

At the prefiltering stage, the union of regular expression set is defined as R, and every regular expression has its own identifier r_i. Then the sub expression print is generated as a sequence of print called P_i. The relationship of the three variables is shown in Equation (1):

$$
R = \bigcup_{i=1}^{n} r_i = \bigcup_{i=1}^{n} p_i \tag{1}
$$

First, every regular expression r_i executes PrintSelect algorithm to generate expression print P_i. According to the definitions of $ES(r)$, $SES(r)$ and $MP(r)$ in Section 2.1, the pseudo code is shown in Algorithm 1. Figure 2 shows the selecting process of regular expression $''a[bc]d.[bc]''$, which has five atoms. Given the parameter $\beta = 256$, we should make sure that the expression size of every print is less than β. The selecting stage begins from the first atom. The *curr* pointer keeps moving to the next atom if $ES(r)$ value of the regular expression print between the begin pointer and end pointer is less than or equal to β. When the *curr* pointer arrives at the fourth atom $''.''$, $ES(a[bc]d.) = 1 * 2 * 1 * 256 = 512 > \beta$. Condition $ES(r) \leqslant \beta$ does not hold, and print $a[bc]d$ is selected. Then we make a directed line from r_i to P_i to mark the correlation relationship. In step 2, although $[bc]d$ satisfies the condition, it is included in the already selected print $''a[bc]d''$. According to section 2.1 that $a[bc]d$ has higher matching probability (MP) than $[bc]d$, thus $[bc]d$ is not selected. Step 3, 4 and 5 follow the same criteria to select print.

After selecting the print, we can get a directed graph from $r_{[1,\cdots n]}$ to $p_{[1,\cdots,m]}$. The relationship of this graph is called correlation sequence.

3.2 CSP Algorithm

The most common way to compute the similarity of regular expression r_i, r_j is to compute the times they appear simultaneously. The basic idea is: the more times they appear in the same expression, the closer their relationships are.

Definition 5. *Correlation between Regular Expressions*

Algorithm 1 PrintSelect(R,P)

```
Begin
Initialize the start,current and end pointer
Assign end to R.size()
Send service request to the SCA.
while start is less than end do
  if ExpressionSize(R,begin,curr) less than β then
    curr:=curr+1
  else
    P.add(R[start,curr])
    start:= start +1
  end if
end while
End
ExpressionSize (R,begin,curr)
Begin
Initialize the map<char,int>and make pair of('a − z0 − 9', 1) etc.
Initialize: ES as expression size.
for r ∈ R[begin, curr] do
  ES := ES * calculator [R[r]]
  return ES
end for
End
```

$Sim(t_i, t_j)$ *is defined as the similarity of* r_i *and* r_j*:*

$$\begin{cases} Sim\,(r_i, r_j) = 1\,(i = j) \\ Sim\,(r_i, r_j) = \frac{|\{t|t_{k,i}>0, t_{k,j}>0\}|}{min(|\{t|t_{k,i}>0\}|, |\{t|t_{k,j}>0\}|)}\,(i \neq j) \end{cases} \quad (2)$$

where $t_{k,i}$ *is the time of resource* p_k *labeled by* t_i*. According to this definition, the similarity of two regular expressions will be 1 when two* r_i *produce the same print or one is the subset of the other.*

The similarity of any two regular expressions can be calculated by Equation (2) and a similarity matrix of R can be obtained: $TT\{1, 2, \cdots |T|\} \times \{1, 2, \cdots |T|\}$*. The matrix element value is the* $Sim(t_i, t_j)$ *in Formula (2).* $|T|$ *is the total number of the regular expression. Then a clustering algorithm is used to cluster the regular expression.*

Definition 6. *Regular Expressions Grouping Our approach divides* $r_1, \cdots, r_n$ *into small correlative groups based on the relationship of pair* r_i *and* p_i *so that they can be stored in each signal wireless node. If* $Sim(r_i, r_j) > \gamma$*,* r_i*,*r_j *will in the same group. The union of each group satisfies* $G = g_1, g_2, \cdots g_n = R$*.*

The value of γ *is decided by the wireless node number and available memory. We want to group as many as correlative expressions in one group when the memory usage is smaller than or equal to available memory. Obviously this is a multiple 0-1 knapsack problem. In this paper, we achieve this goal by using the classical dynamic programming solution for 0-1 knapsack problem [21].*

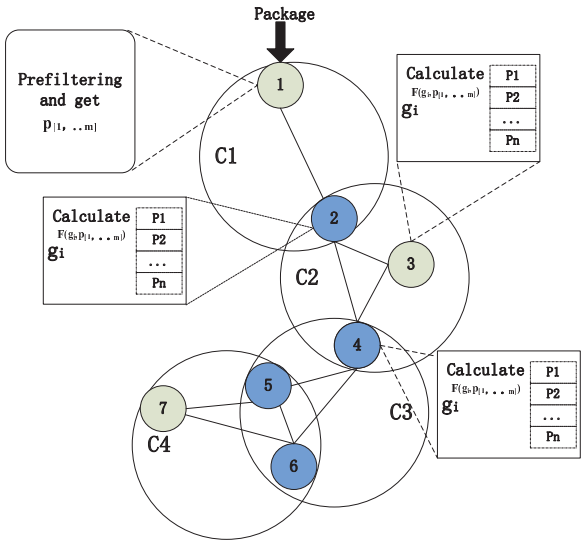

Figure 3: The matching process of ad hoc packages

3.3 Network Security System Matching Strategy

In the ad hoc wireless environment, each package will be transmitted across a certain nodes $N_1, N_2, \cdots N_n$. We will divide these nodes into two groups, one for prefiltering stage and the other for verifying stage. Extra package fields are added to make each node work collaboratively.

Definition 7. *Prefiltered Correlation Sequence For every package* T*, we use* $p_{[1\cdots m]}$ *to represent the prefiltered correlation sequence if it matches print* $p_1, p_2, \cdots, p_m$ *respectively after the prefiltering stage. To measure the correlation of* $p_{[1\cdots m]}$ *and* g_i*, we introduce a search engine model.* $p_{[1\cdots m]}$ *is treated as the keyword of Web Page, and* g_i *as a signal page. Then, we can get the correlation by calculating the frequency of* $p_{[1\cdots m]}$ *in* g_i*. More specifically, if we have m print called* $p_1, p_2, \cdots, p_m$*, and their frequencies are* $f_1, f_2, \cdots f_m$*, we can get the correlation rate to the group* g_i *by* $F(g_i, p_{[1,\cdots,m]})$ *according to Formula (3). Afterwards, we will conduct the verifying step if* $F(g_i, p_{[1,\cdots,m]})$ *is more than* ψ*.*

$$\begin{cases} f_i = \sum (p_i \subseteq g_i) \\ F\left(g_i, p_{[1,\cdots,m]}\right) = f_1 + f_2 + \cdots + f_m. \end{cases} \quad (3)$$

The package matching process is shown in Figure 3. Since each wireless node has limited computing power, we simplify the calculation of $F(g_i, p_{[1,\cdots,m]})$ to be addition only. At first, we need to store the feature vector p_i so that we only need to calculate the sum of pi to get $F(g_i, p_{[1,\cdots,m]})$. In Equation (3), when the package reaches Node 2 after prefiltering of Node 1, the prefiltered correlation sequence $p_{[1\cdots m]}$ will be obtained. Then, Node 2 should calculate the $F(g_i, p_{[1,\cdots,m]})$ by adding the feature vector p_i if $p_{[1\cdots m]}$ is not empty. Node 2 will continue the verifying process using the group g_i in its memory as long

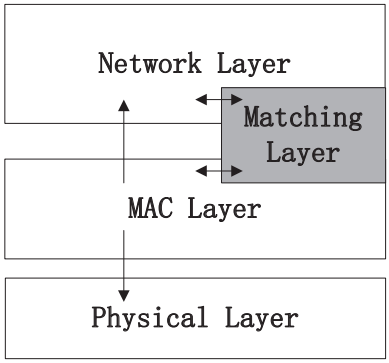

Figure 4: The matching framework of each node

as $F(g_i, p_{[1,\cdots,m]})$ is larger than ψ.Otherwise, the package will be transmitted to the next hop to match g_j.

One possible architecture to implement our matching framework in each wireless node is shown in Figure 4. The implementation called Matching Layer is set between the MAC and network layers, which can be envisioned as an enhancement to existing MAC and routing protocols.

Under this matching strategy, the network package will be checked by the prefiltering stage because it will pass through at least one node. Then we can decide whether this package is safe or needs to be verified by the correlation sequences. Therefore, our matching approach can be equivalent to the lightweight intrusion detection method for networks based on regualr expression [17].

4 Experimental Results

4.1 Experiment Set

Table 1: Experimental parameters setting

Parameter	L7-Filter	Snort
Num of RegExp	161	166
Num of DFA state	1432	1257
β (Expression Size)	256	256
ψ (relevance frequency)	[0.23,0.34]	[0.23,0.34]
η (Match Probability)	[0.75,0.99]	[0.75,0.99]
γ (similarity)	[0.5,10]	[0.5,10]

In this paper, our matching approach is evaluated on regular expression sets extracted from two real world systems, namely L7-Filter and Snort. L7-Filter is a popular open source application layer traffic classifier for Linux. It re-assembles the payload content of a flow and identified its application level protocol through regular expression matching. Snort is a famous open-source intrusion detection system, which can be configured to perform protocol analysis, content inspecting over online traffic to detect a variety of worms, attacks and probes.

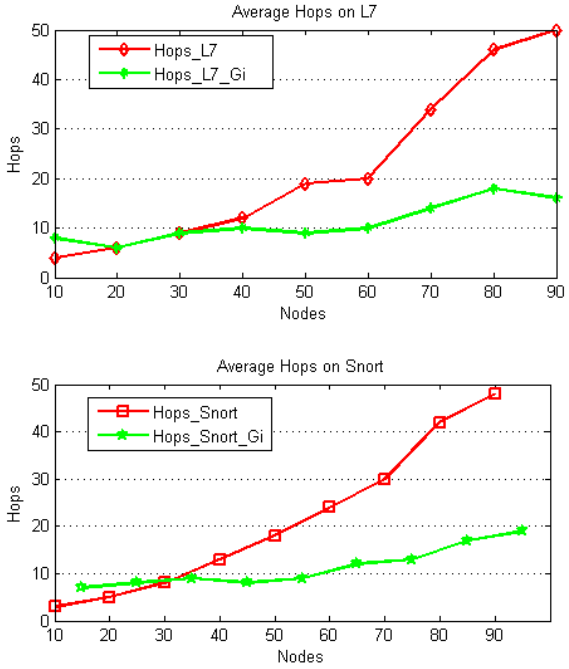

Figure 5: Results of node average hops in NS-2

We choose two sets as $R = r_1, r_2, \cdots, r_n$ to perform our experiments. We also set the experiment parameter: ES, MP, Sim and F to corresponding boundary according to the algorithm in section 3, as shown in Table 1. Then we try to get the local optimal value during our experiments.

4.2 Results on Sequence Generation and Group Division

In the experiments, we select three sets of parameters of L7-Filter and Snort, respectively. The setting of parameters (η, γ, ψ) and the results of each experimental step are shown in Table 2.

Print size denotes the memory occupation of prints after the prefiltering stage; *Group number* is the group number of correlative regular expression according to the parameter γ; *Average similarity* is the average value of similarity in each group (see Equation (2)); *Package size* is the testing packages length. Our simulation environment is based on NS-2 (Network Simulator, version 2). We set the number of nodes from 20 to 100. *Num of suspicious package* is the number of package that needs to be verified after the prefiltering stage. Lastly, we calculate our experiment efficiency by their average executing cost: $Efficiency = (Num\ of\ suspicious\ package/Total\ package\ Number) \times (F(g_i)/\ GroupNumber)$. We can see from the table that we can get a good result from 71.71% to 87.37%.

Our Regular Matching experiments were also performed with our strategy and normal approach to make a comparison. Figure 5 is the Average Hop and Average $F(G_i)$ variation tendency alone with the number of

Table 2: Experimental data

Parameters	L7-Filter			Snort		
	$\eta1$=0.23, $\gamma1$=0.75, $\psi1$=0.5	$\eta2$=0.28, $\gamma2$=0.85, $\psi2$=5	$\eta3$=0.34, $\gamma3$=0.95, $\psi3$=10	$\eta1$=0.23, $\gamma1$=0.75, $\psi1$=0.5	$\eta2$=0.28, $\gamma2$=0.85, $\psi2$=5	$\eta3$=0.34, $\gamma3$=0.95, $\psi3$=10
Print Size	0.15MB	0.31MB	0.11MB	0.24MB	0.36MB	0.21MB
Group Number	13	16	10	16	15	10
Average Similarity	0.82	0.88	0.96	0.79	0.89	0.98
Package Size	1024B	2048B	4096B	1024B	2048B	4096B
Num of suspicious package	250	80	120	40	112	38
Efficiency	87.37%	71.71%	74.0%	80.0%	75.49%	82.132%

nodes. We test L7-Filter and Snort separately. X-axis indicates the nodes number and Y-axis is hops. Then a significant difference can be obtained from results figure that the hops number of using G_i decreases sharply when nodes are more than 30.

Figure 5 shows that the matching approach can test and verify the packages efficiently when the number of wireless nodes is more than 30. This indicates that our approach can be well applied in medium or large scale distributed wireless network systems. However, there is no major difference in average hops when dealing with a small group of wireless nodes. Compared with other end-to-end strategies [1, 16], our approach provides a well scalable way by integrating distributed wireless nodes to detect intrusion. By selecting appropriate parameters, our system can effectively monitor network attacks.

5 Conclusion and Future Work

In this paper, a regular expression matching approach for distributed wireless network security systems is proposed to take advantage of distributed nodes collaboratively, which divides the matching into a prefiltering stage and a verifying stage. We compare the performance of the proposed method with the standard regular expression matching approach. Experimental results show that our strategy can speed up the efficiency of regular expression by at least 71% for the regular expression set of Snort and L7-Filter systems. Emulation also proves that our approach can be well applied in medium or large scale distributed wireless network systems if the number of nodes is more than 30.

Several aspects of the verifying strategy could be improved in the future, such as the regular expressions grouping coefficient selection and the robustness of multiple matching. Future work will also extend the proposed approach and explore its feasibility for other network areas.

Acknowledgments

This study was supported by the National Natural Science Funds of trusted software major research projects (No. 91018003) and Fundamental Research Funds for the Central Universities (No. DUT14QY32). The authors gratefully acknowledge the anonymous reviewers for their valuable comments.

References

[1] S. S. Ahmeda, "ID-based and threshold security scheme for ad hoc network," in *IEEE 3rd International Conference on Communication Software and Networks*, pp. 16–21, Xi'an, China, 2011.

[2] M. Becchi and P. Crowley, "Efficient regular expression evaluation: theory to practice," in *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pp. 50–59, San Jose, USA, 2008.

[3] B. C. Brodie, D. E. Taylor, and R. K. Cytron, "A scalable architecture for high-throughput regular-expression pattern matching," in *ACM SIGARCH Computer Architecture News*, vol. 34, pp. 191–202, 2006.

[4] H. Deng, W. Li, and D. P. Agrawal, "Routing security in wireless ad hoc networks," *IEEE Transactions on Communications Magazine*, vol. 40, no. 10, pp. 70–75, 2002.

[5] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pp. 41–47, Dalian, China, 2002.

[6] Y. C. Hu, D. B. Johnson, and A. Perrig, "Sead: Secure efficient distance vector routing for mobile wireless ad hoc networks," *Ad Hoc Networks*, vol. 1, no. 1, pp. 175–192, 2003.

[7] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. Turner, "Algorithms to accelerate multiple regular expressions matching for deep packet inspection," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 339–350, 2006.

[8] S. Kumar, J. Turner, and J. Williams, "Advanced algorithms for fast and scalable deep packet inspection," in *Proceedings of the 2006 ACM/IEEE symposium on Architecture for Networking and Communications Systems*, pp. 81–92, New York, USA, 2006.

[9] B. M. Leiner, R. J. Ruther, and A. R. Sastry, "Goals and challenges of the DARPA glomo program global mobile information systems," *Personal Communications, IEEE*, vol. 3, no. 6, pp. 34–43, 1996.

[10] J. Levandoski, E. Sommer, M. Strait, et al. "Application layer packet classifier for linux,", 2008.

[11] J. P. A. X. Liu and E. Torng, "Bypassing space explosion in regular expression matching for network intrusion detection and prevention systems," in *Proceedings of the 19th Annual Network & Distributed System Security Symposium*, pp. 11–34, San Diego, USA, 2012.

[12] T. Liu, Y. Sun, A. X. Liu, L. Guo, and B. Fang, "A prefiltering approach to regular expression matching for network security systems," in *Applied Cryptography and Network Security*, pp. 363–380, Berlin, Germany, 2012.

[13] R. Matam and S. Tripathy, "Provably secure routing protocol for wireless mesh networks," *International Journal of Network Security*, vol. 16, no. 3, pp. 182–192, 2014.

[14] C. R. Meiners, J. Patel, E. Norige, E. Torng, and A. X. Liu, "Fast regular expression matching using small TCAMs for network intrusion detection and prevention systems," in *Proceedings of the 19th USENIX Conference on Security*, pp. 111–126, Washington DC, USA, 2010.

[15] A. Mitra, W. Najjar, and L. Bhuyan, "Compiling pcre to fpga for accelerating snort ids," in *Proceedings of the 3rd ACM/IEEE Symposium on Architecture for Networking and Communications Systems*, pp. 127–136, New York, USA, 2007.

[16] A. Prathapani, L. Santhanam, and D. P. Agrawal, "Detection of blackhole attack in a wireless mesh network using intelligent honeypot agents," *The Journal of Supercomputing*, vol. 64, no. 3, pp. 777–804, 2013.

[17] M. Roesch et al., "Snort: Lightweight intrusion detection for networks," in *LISA*, vol. 99, pp. 229–238, 1999.

[18] R. Sidhu and V. K. Prasanna, "Fast regular expression matching using FPGAs," in *The 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 227–238, California, USA, 2001.

[19] N. Siegel, D. Hall, C. Walker, and R. Rubio, "The tactical internet graybeard panel briefings," *US Army Digitization Office*, 1997.

[20] I. Sourdis and D. Pnevmatikatos, "Pre-decoded CAMs for efficient and high-speed NIDS pattern matching," in *The 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 258–267, Napa, USA, 2004.

[21] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

Jie Wang is a Lecturer in Department of Embedded Systems Engineering, Dalian University of Technology, China. His main research interests focus on parallel computing and network security. He received his B.S., M.S. and Ph.D. degree from Department of Computer Science & Technology, Harbin Institute of Technology, China.

Yanshuo Yu received BS degree in software engineering from Dalian University of Technology in 2012. He is currently working toward a master degree in Department of Embedded Systems Engineering, Dalian University of Technology His research interests include network security and parallel computing.

Kuanjiu Zhou is a professor in Department of Embedded Systems Engineering, Dalian University of Technology, China. His main research interests focus on embedded software modeling and trusted software. He received his B.S.,M.S. and Ph.D. degree from Department of Computer Science & Technology, Harbin Institute of Technology, China.