INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS
Int. J. Commun. Syst. (2014)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/dac.2849

Traffic diversity and code coverage: a preliminary analysis

Ying-Dar Lin[1], Yuan-Cheng Lai[2], Chun-Nan Lu[1,*,†], Jui-Tsun Hung[1] and Chun-Pin Shao[1]

[1]*Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan*
[2]*Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan*

SUMMARY

It is generally assumed that using more diverse traffic to test network devices could achieve larger code coverage. However, how to describe the diversity of traffic traces and the relationship between the traffic diversity and code coverage is still an issue. In this paper, the traffic diversity is defined using the number of packets and the size of the subnets involved, and traces having various diversity are used to evaluate the corresponding code coverage for the programs in a network device. Experiment results show that more number of packets or larger size of network segments can generate larger diversity indices and thus larger code coverage. For Snort, as the number of packets increases from 1 to 10,000,000, representative diversity index and the code coverage can increase from 0 to 0.95 on the basis of Simpson's index and from 19.1% to 32.2%, respectively. As the size of network segments increases, representative diversity index and the code coverage can increase from 0.41 to 0.82 and from 28.2% to 32.2%, respectively. Similar results can be obtained in the case of Linux kernel. If the mappings among different diversity indices and the corresponding code coverage can be built beforehand, the quality of the tests can improved. Copyright © 2014 John Wiley & Sons, Ltd.

Received 11 March 2014; Revised 22 June 2014; Accepted 4 August 2014

KEY WORDS: traffic diversity; code coverage; diversity index; network test

1. INTRODUCTION

Network traffic traces can be used to test and verify network devices [1–3]. For example, NCTU Beta Site [4] captures daily traffic generated from hundreds of volunteers to test network devices and security appliances. The context of real-world traffic is hard to predict in advance because of different hosts, periods, and activities involved. Even for two traffic traces with the same number of hosts involved, it is still possible that they are totally dissimilar because of different network segments or during different capture time periods. In order to distinguish and measure the differences between traffic traces, the *diversity* of packet traces should be defined.

A metric, *biological diversity* [5], is defined to denote the degree of variation of life species within a given ecosystem for measuring the health of a given ecosystem. On the basis of the definition, greater diversity implies better health. Considering packets that belong to a defined type as individuals of a life species, the concept of biological diversity can be used to describe how diverse a network traffic trace is. Furthermore, a traffic diversity index is also needed to quantify the differences of any two traffic traces. Several well-known diversity indices, such as Simpson's index [6], Shannon's index [7, 8], and Renyi's index [9, 10], are thus proposed to calculate traffic diversity. In this work, the traffic diversity index is defined by the header of IP addresses with fixed-length fields. All diversity indices of a traffic trace will be grouped into a diversity vector and then

*Correspondence to: Chun-Nan Lu, Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan.
†E-mail: cnlu@cs.nctu.edu.tw

merged as a combined diversity index. The diversity of a traffic trace will be compared according to every single traffic diversity index and the combined traffic diversity index.

Code coverage [11, 12] is a metric defined to present the coverage of a source program being tested, where the source code is instrumented by code coverage analysis tools. More code coverage obtained can ensure more robust product quality because more codes can be verified. However, the code coverage is nontrivial to obtain in every different test round. Different context of tested traffic can trigger distinct kinds of product failures. In order to clarify the relationship between traffic diversity of traffic traces and the code coverage, packet traces with various traffic diversities are applied to test target programs installed in a network device, and then, the corresponding code coverage is measured and analyzed.

The organization of this paper is as follows. In Section 2, related works including introductions to diversity index, well-known diversity indices, and code coverage will be presented. In Section 3, definitions and problem statement will be presented. System architecture and proposed methodology will be presented in Section 4. Experiment results and observations will be presented in Section 5. Finally, conclusion of this thesis will be presented in Section 6.

2. RELATED WORK

In this section, two related terms, diversity index and code coverage, are introduced as follows.

2.1. Diversity index

A diversity index is a statistic to measure the local members of a set consisting of various types of objects. It was first introduced in ecology [8] to measure the biodiversity of an ecosystem and can be also applied to other areas. For example, in economics, the diversity index can be defined to measure the distribution over sectors of economic activity in a region; whereas in information science, the diversity index can be used to explain the complexity of a set of information.

There are two basic factors of measuring diversity index: species richness and species evenness [5]. Species richness is the number of different species present in an ecological system and assert that the more species present in a habitat will result in a higher richness of the habitat. Species richness does not take the abundances of the species or their relative abundance distributions into consideration. On the contrary, species evenness expresses how close in numbers each species in a system are. Species evenness is designed to show the relative abundance or proportion of individuals among the species. Three well-known metrics used to measure diversity degrees are described as follows.

2.1.1. Simpson's index [6]. Simpson's index in terms of ecology, denoted as D, concerns the species richness and the species evenness. It represents the probability that two randomly selected individuals in a habitat will belong to the same species. The calculation of Simpson's index D is defined as $D = \frac{\sum_{i=1}^{S} n_i(n_i - 1)}{N(N-1)}$, where N represents the total number of individuals of all species, n_i is the number of individuals in species i, and S is the number of species. However, it is more often to use $\widetilde{D}$, where $\widetilde{D} = 1 - D$, to present diversity intuitively. In such kind of notation, index 0 represents no diversity of species, and index 1 means infinite diversity. A higher value of $\widetilde{D}$ represents larger diversity.

2.1.2. Shannon's index [7, 8]. The Shannon's index H cares the species richness and the species evenness as well. It shows the information entropy of the distribution, and the species and the relative population sizes are treated as symbols and the probabilities, respectively. The calculation of Shannon's index H is defined as $H = -\sum_{i=1}^{S} p_i \ln p_i$, where p_i represents the relative abundance of each species and is calculated as n_i/N, S is the number of species, n_i is the number of individuals in species i, and N the total number of all individuals.

DOI: 10.1002/dac

2.1.3. Renyi's index [9, 10]. Renyi's index is a generalization of Shannon's index. The Renyi's index of order α is defined as $H = \frac{1}{1-\alpha} \ln \sum_{i=1}^{S} p_i^{\alpha}$, where $\alpha \geq 0$, $\alpha \neq 1$, p_i represents the relative abundance of each species and is calculated as n_i/N, n_i is the number of individuals in species i, N is the total number of all individuals, and S is the number of species. Lower value of α, approaching to 0, more equally gives an index with an increasingly weights to all possible events, regardless of their probabilities. If α is approaching to 1, Renyi's index will reach to Shannon's index; if α is approaching to 0, Renyi's index leads to the maximum of Shannon's index.

2.1.4. Comparison. All aforementioned indices consider both richness and evenness and are compared in the view of *sample size sensitivity* and the difficulty of calculation [8]. Because Simpson's index has lower sample size sensitivity and its calculation is easier than Shannon's index and Renyi's index, Simpson's index is our final choice.

2.2. Code coverage

Testing networking devices before releasing them onto the market is a way of ensuring quality and robustness. Raad and Monhamed [13] evaluated the performance comparison between different traffic load density and the throughput. Shun-Ren [14] proposed a conformance test tool to specify test cases at an abstract level, which makes it easier to generate more comprehensive and extensible test cases. However, the number of test cases is far more than imaginable. For example, regression testing is another type of testing that is performed to verify that new changes do not damage the existing behavior of this software. Test suites tend to grow in size as software evolves, which often makes it too costly to run entire test suites. Yoo and Harman [15] showed a survey of regression testing on each area of test suite minimization, regression test case selection, and test case prioritization. Test suit minimization seeks to eliminate redundant test cases, test case selection seeks to identify the test cases that are relevant to some set of recent changes, and test case prioritization seeks to order test cases in such a way that early fault detection is maximized.

Ying-Dar *et al*. [16] focused on the problem of how to achieve a maximal amount of function coverage under certain constraints, such as the minimal number of test cases, the minimal cost of test cases, limited testing time, or a required level of coverage, and proposed six corresponding algorithms to solve these problems. Code coverage, proposed in [11, 12], is a measure invented for systematic software testing to describe the coverage of the source code that a program has been tested. To measure how well the program is evaluated by a test suite, one or more coverage criteria have been proposed [17, 18]. Three main coverage criteria are the following: (i) function level coverage – the percentage of functions called in a program; (ii) branch level coverage – the percentage of branches of control structures decided in a program; and (iii) line level coverage – the percentage of lines executed in a program. For example, a C-like function is as follows:

```
int foo (int x, int y)
{
   int z = 0;
   if ((x > 0) && (y > 0)) z = x;
   return z;
}
```

This function is assumed to be part of some bigger program, and this program is evaluated with a test suite. If function *foo* is called at least once, the function level coverage for *foo* is satisfied. The branch level coverage can be satisfied with test cases of *foo*(1, 1), *foo*(1, 0), and *foo*(0, 0). These cases are necessary because in the first two cases, $(x > 0)$ evaluates to true; whereas in the third case, $(x > 0)$ is false. The line level coverage can be satisfied if *foo*(1, 1) is required. In this case, every line in this function is executed.

DOI: 10.1002/dac

3. TRAFFIC DIVERSITY VERSUS CODE COVERAGE

In this section, various diversity indices are elaborated to clarify the diversity of packet traces accordingly. The code coverage of branches, C, is used to present the percentage of branches executed in a program.

In order to find the relationship between the traffic diversity and code coverage, we define six diversity indices, which are calculated on the basis of the definition of Simpson's index. Simpson's index D can be used to represent the probability that two randomly selected individuals in the habitat will belong to the same species in ecology. In other words, we can use $\widetilde{D} = 1 - D$ to denote the probability that two randomly selected individuals will belong to different species.

With Simpson's index, six types of diversity indices are specified. D_{s_IP}, the diversity index of the source IP address in a packet trace, means the probability of randomly selecting any two packets from a packet trace with different source IPs. In the same way, D_{d_IP} stands for the diversity index of destination IP address, D_{s_port} is used for source port, D_{d_port} is used for destination port, and D_{app} is used for application header. A combined diversity index $D_{mix} = D_{s\_IP} \times D_{d\_IP} \times D_{s\_port} \times D_{d\_port}$ is defined to be the probability of randomly selecting any two packets from a packet trace, where the source IPs, destination IP, source port, and destination port are not the same.

Therefore, the goal of this work can be stated as follows. Given multiple packet traces with various traffic diversity, which is represented by D_{mix}, and different code coverage C, our goal is to clarify the relationship between the traffic diversity of packet traces and the corresponding code coverage based on defined diversity indices.

4. SYSTEM ARCHITECTURE AND METHODOLOGY

In this work, a system architecture including a diversity index calculator and a code coverage analyzer is designed. Furthermore, a methodology for calculating diversity indices and analyzing the corresponding code coverage is also proposed.

4.1. System architecture

Figure 1 illustrates the overview of this system. The system consists of two components, one is a diversity indices calculator and the other is a code coverage instrument/analysis tool. The diversity index calculator is used to calculate the different diversity indices, namely, D_{mix}, D_{s_IP}, D_{d_IP}, D_{s_port}, D_{d_port}, and D_{app}, by different packet header fields on the basis of the definition of Simpson's index, whereas the code coverage instrument/analysis tool is used to evaluate the code coverage of a target source code, which is instrumented first by a code instrument tool. Packet traces with different diversity indices are then replayed to a device under test (DUT), which is a product undergoing testing, to evaluate the corresponding code coverage. Finally, the results are collected. In our context, a DUT is a network device that connects computers or other devices together and handles network packets.

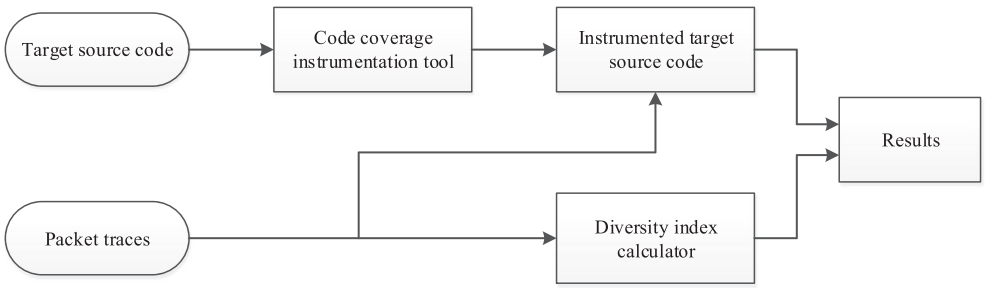


Figure 1. Diversity indices calculator and code coverage analysis tool.

DOI: 10.1002/dac

4.2. Methodology

Figure 2 illustrates the process of calculating diversity index using Simpson's index. The process includes the following: (i) the individual header field of packets in a trace, such as source IP, destination IP, protocol, source port, destination port, and application header, are extracted and stored separately in files; (ii) each file is sorted according to their contents, where identical items can be grouped first; and (iii) each diversity index of a specific header field is calculated on the basis of Simpson's index.

Figure 3 shows the process of a code coverage analyzer. In Figure 3, the target source code is first instrumented using tracking instructions by the instrumentation tool Gcov [19]. Gcov is a test coverage program and can be configured to accumulate statistics by line, basic block, or branch. Gcov creates a log file called *file.gcov* that indicates how many times each line or branch of a source file *file.c* has executed. Therefore, in order to evaluate the code coverage of the target source code, the target source code is first instrumented using Gcov, for example, on branch level. Next, packet traces with different diversity indices are replayed to the DUT that executes the instrumented source code to evaluate the code coverage. During the test, Gcov accumulates and logs the branch level statistics of code coverage. We can then use the corresponding log file, *file.gcov*, to evaluate the code coverage of the program. With the information of the different diversity indices and corresponding code coverage, the relationship between them can be found and explored.

Therefore, for a code coverage analyzer, it should try to find and formulate the relationship between different traffic diversity indices and corresponding code coverage. With this kind of information, the testers can design distinct test plans to fit different test scenarios, test quality, and product requirements.

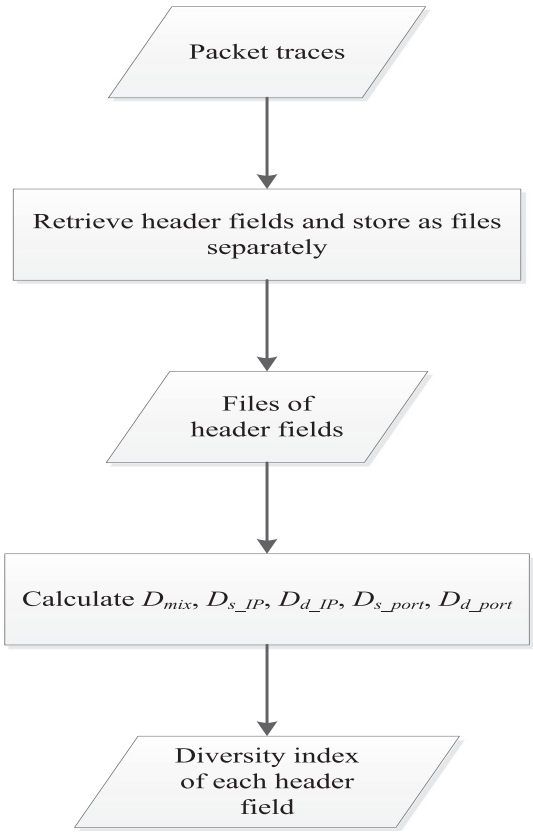


Figure 2. The process of calculating the diversity index of a field.

DOI: 10.1002/dac

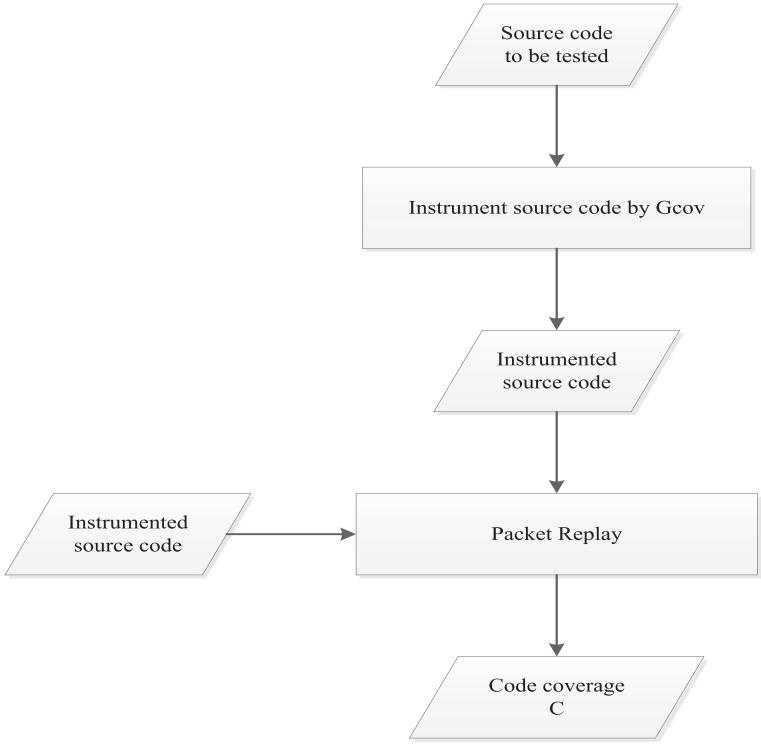


Figure 3. The process of a code coverage analyzer.

5. EVALUATIONS

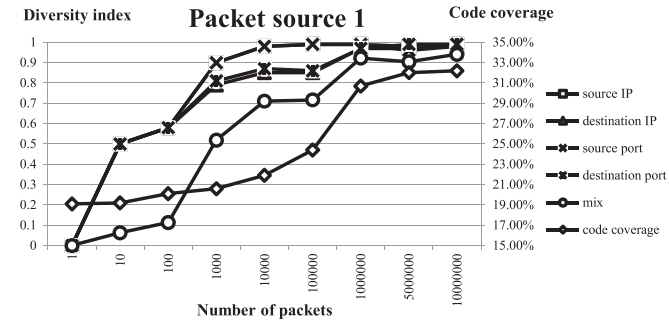
In our experiments, Snort [20] and Linux kernel [21] were used as the evaluated programs, and packets traces of different number of packets and distinct size of network segments were used to evaluate the code coverage during the test.

5.1. Configurations

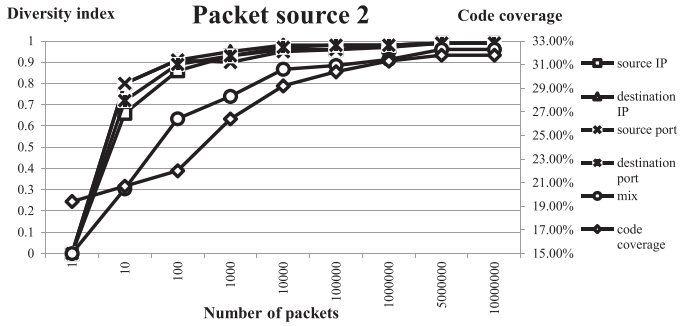
The programs to be tested are Snort-2.9.0.5 and Linux kernel-2.6.35. Snort is a user-level program, whereas Linux kernel is a kernel-level program. Both programs were instrumented by Gcov and ran on Ubuntu 10.10. A user-level program can be instrumented by Gcov directly, whereas a kernel-level program needs to be instrumented by Gcov with kernel patches and modules.

Packet traces used in following experiments are captured from two different network segments, 140.113.243.0/24 and 140.113.249.0/24, which are later named packet source 1 and packet source 2, respectively. The traffic captured from the network segment 140.113.0.0/16 is used for verification. Each IP is used by at least one host behind. Packet trace can be further divided by the following: (i) the number of packets (1.pcap, 10.pcap, 100.pcap, 1000.pcap, 10000.pcap, 100000.pcap, 1000000.pcap, 5000000.pcap, and 10000000.pcap) and (ii) the size of network segments (140.113.0.0/16.pcap, 140.113.243.0/24.pcap, 140.113.243.0/26.pcap, 140.113.243.0/28.cap, 140.113.249.0/24.pcap, 140.113.249.0/26.pcap, and 140.113.249.0/28.cap). Two sets of packet traces are retrieved from packet source 1 and packet source 2, respectively. In the category of the number of packets, a packet trace with fewer packets is designed to be a subset of a trace with more packets. For example, 10.pcap is a subset of 100.pcap and also a subnet of 10000.pcap. In the category of the size of network segments, a packet trace with a smaller segment is a subset of a trace with a larger segment. For example, 140.113.243.0/28.pcap is a subset of 140.113.243.0/26.pcap and also a subnet of 140.113.0.0/16.pcap.

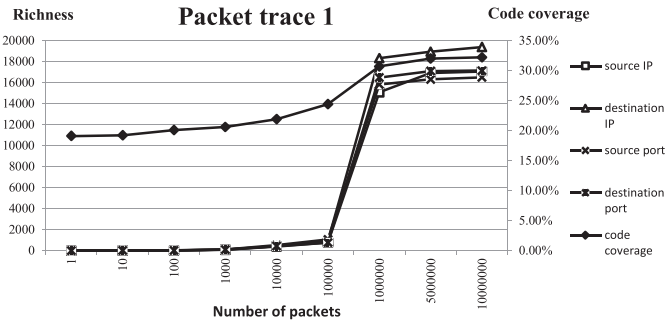
DOI: 10.1002/dac



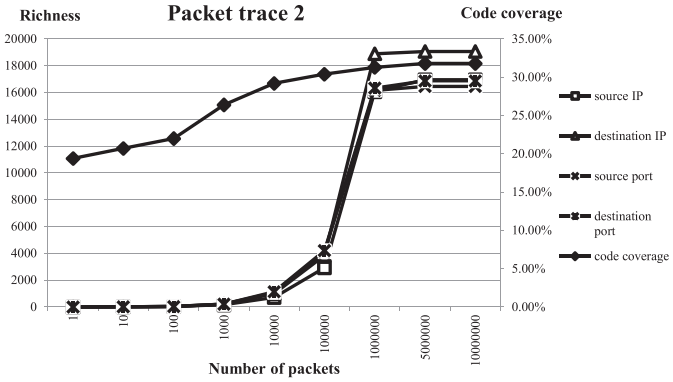
(a) Effects of different number of packets on diversity indices and code coverage



(b) Effects of different number of packets on diversity indices and code coverage



(c) Effects of different number of packets on richness and code coverage



(d) Effects of different number of packets on richness and code coverage

Figure 4. Effects of different number of packets on diversity index, richness, and code coverage for different packet sources.

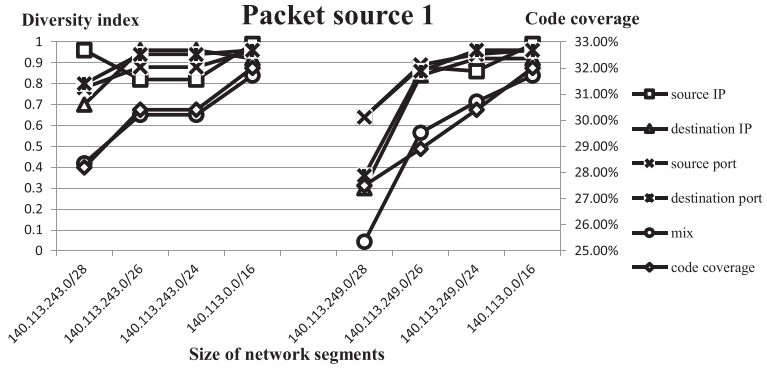
DOI: 10.1002/dac

5.2. Experiment results

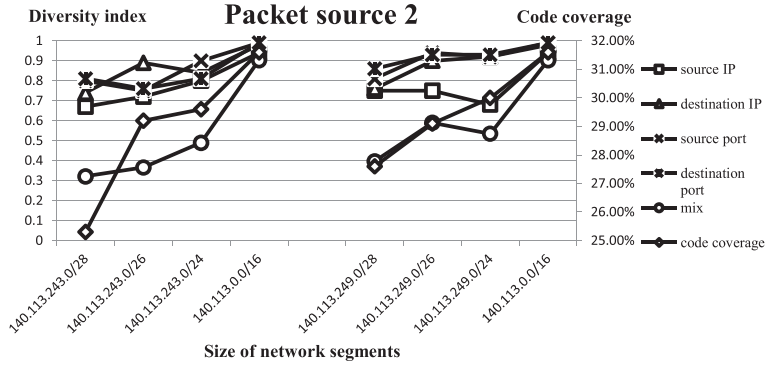
In the following results, from Figures 4–8, the relationship between traffic diversity index, richness, evenness, and code coverage were showed using different packet sources. In the following figures, related results were put into the same figure to show their relationship. The left vertical axis denoted diversity index, ranging from 0 to 1, and the right vertical one showed the numeric results corresponding to different features. The horizontal axis shows different number of packets or size of network segments.

5.2.1. Snort. Figure 4 illustrated that diversity indices and code coverage increased as the number of packet increased. In Figure 4(a), the representative diversity index D_{mix} and code coverage of packet traces from packet source 1 increased from 0 to 0.95 and from 19.1% to 32.2% as the number of packets increased from 1 to 10,000,000.

Different diversity indices usually increased as the number of packets increased except two conditions in our experiments: (i) the source IP diversity index D_{s_IP} and the source port diversity index D_{s_port} when the number of packets increased from one million to five million and (ii) the destination port diversity index D_{d_port} when the number of packets increased from 10,000 to 100,000. These exceptions are because Simpson's index considers both richness and evenness. For the first case, D_{s_port} decreased when the number of packets increased from one million to five million because the number of several source ports grew much larger than other source ports, causing that the value of diversity index decreased from 0.98 to 0.96. For the second case, because the number of several destination ports grew much larger than other destination ports, D_{d_port} of the number of packets as 10,000 and 100,000 are 0.87 and 0.86, respectively. Also, D_{mix} might therefore decrease because D_{mix} is defined as the product of D_{s_IP}, D_{d_IP}, D_{s_port}, and D_{d_port}. A decrease in any of the



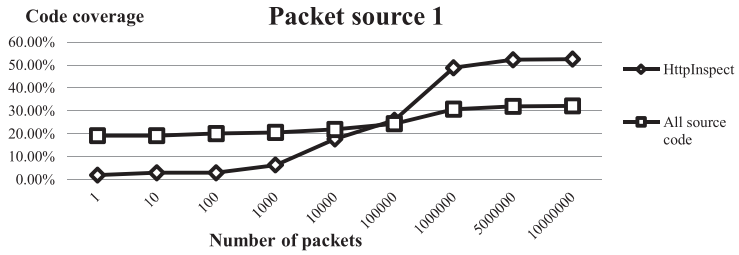
(a) Effects of different size of network segments on diversity indices and code coverage



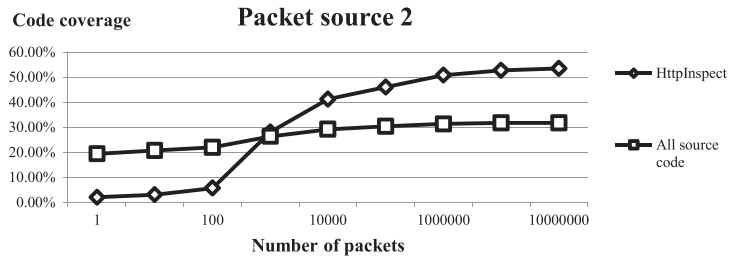
(b) Effects of different size of network segments on diversity indices and code coverage

Figure 5. Effects of different size of network segments on diversity indices and code coverage for different packet sources.

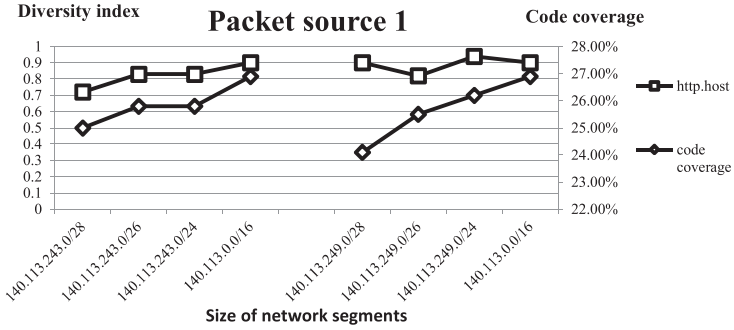
DOI: 10.1002/dac



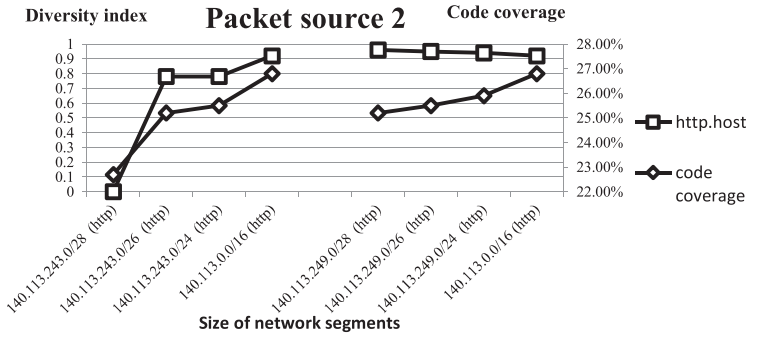
(a) Effects of different number of packets on different codes



(b) Effects of different number of packets on different codes



(c) Effects of different size of network segments on different codes



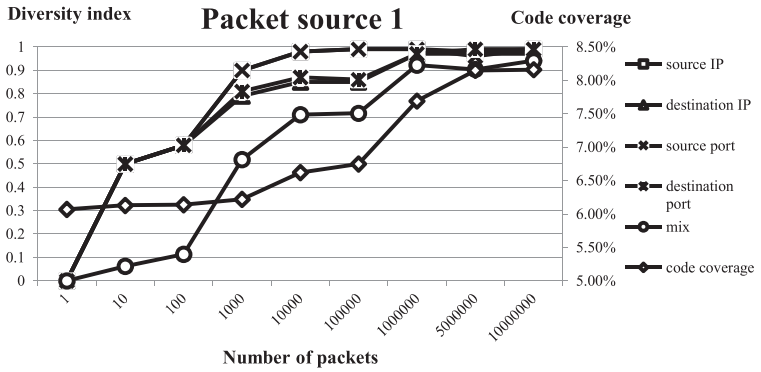
(d) Effects of different size of network segments on different codes

Figure 6. Effects of different number of packets and different size of network segments for different codes.

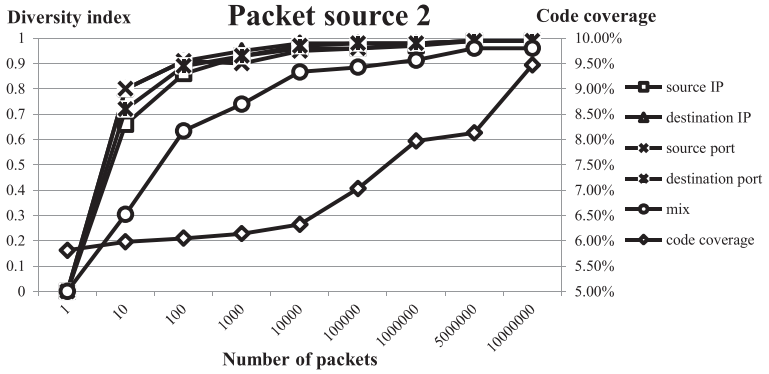
four diversity indices will make D_{mix} decrease as well. We also noticed that when the number of packets exceeded 1,000,000, code coverage increased slowly. It means that the packet source 1 covered most of source code when the packet number exceeded 1,000,000.

Similarly, in Figure 4(b), code coverage of packet traces from packet source 2 ranged from 19.4% to 31.8% as the number of packets increases from 1 to 10,000,000. D_{mix} increased from 0 to 0.96, and other diversity indices also increased as the number of packets increased. Note that as the

DOI: 10.1002/dac



(a) Effects of different number of packets on diversity indices and code coverage



(b) Effects of different number of packets on diversity indices and code coverage

Figure 7. Effects of different number of packets on diversity indices and code coverage for different packet sources.

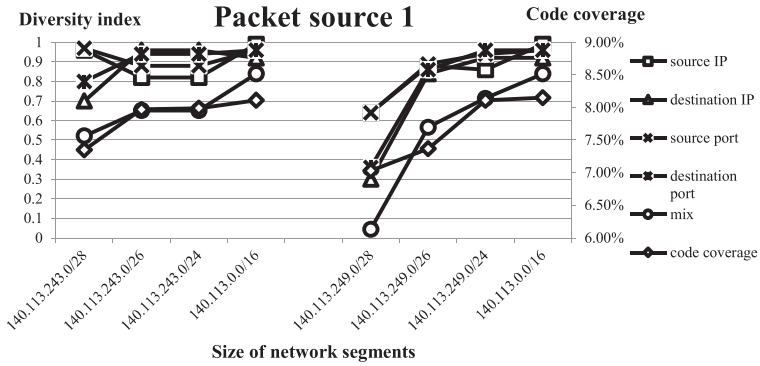
number of packets increases, diversity indices increase from a macro view, but sometimes, it decreases from a micro view, as shown in Figure 4(a) and explained earlier. This anomaly will appear sometimes because the bias distributions among different species, but in most cases, diversity indices increase as the number of packets increases.

In Figure 4(c), the values of richness for various header fields of packet traces from packet source 1 increased as the number of packets increased. A burst of network traffic from the same source IP address leads to the decrease of evenness, which can cause the decrease of diversity indices. In Figure 4(d), the richness of each header fields of packet traces from packet source 2 also increased as the number of packets increased.

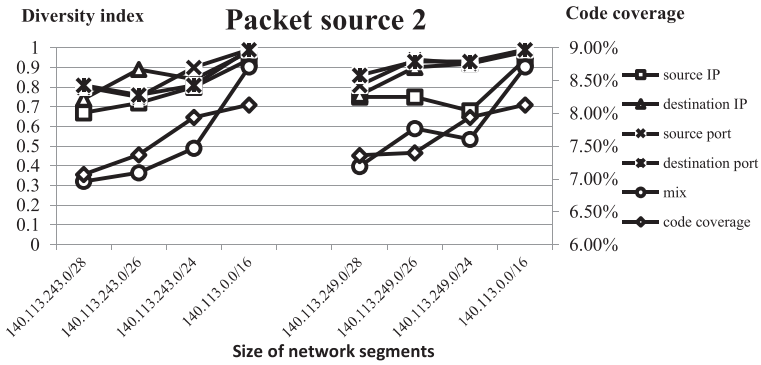
Figure 5 showed that diversity indices and code coverage increased as network segment size became larger. In Figure 5(a), D_{mix} and code coverage of packet source 1 increased from 0.41 to 0.82 and from 28.2% to 32.2%, respectively, as the size of network segments became larger. A larger size of network segments contained more hosts, implying more richness in IP diversity. In Figure 5(b), D_{mix} and code coverage of packet source 2 increased from 0.31 to 0.9 and from 27.5% to 31.8%, respectively, as the size of network segments became larger. However, source IP diversity index D_{s_IP} decreased in network segment 140.113.249.0/24, which means that traffic bursts occurred in 140.113.249.0/24.pcap, resulting in worse evenness.

In Snort, the source codes in the directory 'HttpInspect' were used to decode user applications. Given a data buffer, HttpInspect codes decode the data, locate HTTP fields, and normalize the fields. Thus, this part of Snort source code should be greatly influenced by different packet traces. The percentage of branches of the directory HttpInspect to all source code is around 7%. In Figure 6(a), the code coverage of HttpInspect codes and the whole Snort codes that packet source 1 could achieve ranged from 2.1% to 52.6% and from 19.2% to 32.2% as the number of packets increased from 1 to 10,000,000.

DOI: 10.1002/dac



(a) Effects of different size of network segments on diversity indices and code coverage



(b) Effects of different size of network segments on diversity indices and code coverage

Figure 8. Effects of different size of network segments on diversity indices and code coverage for different packet sources.

In Figure 6(b), the code coverage of HttpInspect codes and the whole Snort codes that packet source 2 can achieve ranged from 2.1% to 53.5% and from 19.4% to 31.8% as the number of packets increased from 1 to 10,000,000.

The field 'http.host' having variable length in HTTP header was selected to monitor the code coverage. Packet traces used here were pure HTTP traffic. In Figure 6(c), code coverage that packet source 1 could achieve increased from 24.1% to 26.9% when the size of network segments became larger. However, evenness made the diversity index decrease at the case of 140.113.249.0/26, whereas richness increased when the size of network segments became larger. In Figure 6(d), code coverage that packet source 2 could achieve increased from 22.7% to 26.8%, while the size of network segments became larger. However, evenness decreased the diversity index from 140.113.249.0/28 to 140.113.0.0/16.

5.2.2. Linux kernel. Linux kernel-2.6.35 was instrumented by Gcov with kernel patches and recompiled again. We only evaluated the source codes in the directory '/net' because the directory /net is directly related to process network traffic. Figure 7 showed that diversity indices and code coverage increased as the number of packets increased.

In Figure 7(a), D_{mix} and code coverage that packet source 1 could achieve increased from 0 to 0.92 and from 6.07% to 8.16%, respectively, as the number of packets increased from 1 to 10,000,000. In Figure 7(b), D_{mix} and code coverage that packet source 2 can achieve increased from 0 to 0.96 and from 5.81% to 9.47%, respectively, as the number of packets increased from 1 to 10,000,000.

Figure 8 illustrated that diversity indices and code coverage increased as the size of network segments became larger. A larger size of network segment contains more hosts, which causes higher richness in IP diversity. However, the source IP diversity index D_{s_IP} decreased in 140.113.249.0/24 using packet source 1, which meant evenness was worse.

DOI: 10.1002/dac

6. CONCLUSIONS

In this work, we defined diversity indices for both fixed-length and varied-length header fields and proposed a methodology for calculating the diversity index and analyzing code coverage of a packet trace. Traffic diversity was calculated on the basis of the definition of Simpson's index that considers the richness and evenness of the contents simultaneously. Two programs, Snort and Linux kernel, were used as DUTs. The related programs were instrumented and analyzed by Gcov to obtain the corresponding code coverage.

According to experiment results, diversity indices and code coverage all increased as the number of packets or the size of network segments of different packet traces increased. For Snort, code coverage of HttpInspect codes and the whole Snort codes that packet source 1 achieved increased from 2.1% to 52.6% and from 19.1% to 32.2%, respectively, as the number of packets increased from 1 to 10,000,000. As the size of network segments increased, the code coverage of the variant length of http.host field and the whole Snort codes that packet source 1 achieved increased from 24.1% to 26.9% and from 28.2% to 32.2%, respectively. With packet source 2, code coverage of HttpInspect codes and the whole Snort codes increased from 2.1% to 53.5% and from 19.4% to 31.8%, respectively, as the number of packets increased from 1 to 10,000,000. As the size of network segments increased, the code coverage of the variant length of http.host field and the whole Snort codes increased from 22.7% to 26.8% and from 27.5% to 31.8%, respectively.

The effects of different number of packets and different size of network segments on diversity indices and code coverage were similar for the case of Linux kernel. Code coverage of Linux /net codes increased from 6.07% to 8.16% for packet source 1 and from 5.81% to 9.47% for packet source 2 as the number of packet increased from 1 to 10,000,000. Richness of packet traces increased when the number of packets or size of network segments increased. However, evenness may be influenced by traffic bursts, which leads to decrease the diversity indices.

Network traffic is suitable to test network devices because it has complicated user behaviors and hundreds to thousands of applications inside. With the help of traffic from different number of packets or different size of network segments, distinct coverage of codes can be evaluated and verified. In the trend of shorter development and test cycles and emerging number of products, if the code coverage can be controlled or even predicted, the test efficiency and the quality of products can be raised.

REFERENCES

1. Chen Y-S, Deng D-J, Hsu Y-M, Wang S-D. Efficient uplink scheduling policy for variable bit rate traffic in IEEE 802.16 BWA systems. *International Journal of Communication Systems* 2012; **25**(6):734–748.
2. Adami D, Callegari C, Giordano S, Pagano M, Pepe T. Skype-Hunter: a real-time system for the detection and classification of Skype traffic. *International Journal of Communication Systems* 2012; **25**(3):386–403.
3. Khosroshahy M. UARA in edge routers: an effective approach to user fairness and traffic shaping. *International Journal of Communication Systems* 2012; **25**(2):169–184.
4. Lin Y-D, Chen I-W, Lin P-C, Chen C-S. On campus beta site: architecture, designs, operational experience, and top product defects, in *IEEE Communications Magazine* 2010; **48**:83–91.
5. Dasmann RF. *A Different Kind of Country*. Collier Macmillan: New York: MacMillan, 1971. ISBN: 0-02-072810-7
6. Simpson EH. Measurement of diversity. *Nature* 1949; **163**:688.
7. Shannon CE. A mathematical theory of communication. *Bell System Technical* 1948; **27**:379–423, 623-656.
8. Menhinick EF. A comparison of some species-individuals diversity indices applied to samples of field insects. *Ecology* 1964; **45**:859–861.
9. Le Cam LM. On some asymptotic properties of maximum likelihood estimates and related Bayes' estimates. *University of California Publications in Statistics* 1953; 277–330.
10. Renyi A. On the measures of entropy and information, in *Proceedings of Fourth Berkeley Symposium on Mathematics, Statistics and Probability*, Statistical Laboratory of the University of California, Berkeley, 1961; **1**:547–561.
11. Myers GJ. *The Art of Software Testing*, (2nd edn). Wiley, 2011. ISBN: 0471469122
12. Miller JC, Maloney CJ. Systematic mistake analysis of digital computer programs. *Communications of the ACM* 1963; **6**(2):58–63.
13. A1-Qassas RS, Ould-Khaoua M. Performance comparison of end-to-end and on-the-spot traffic-aware techniques. *International Journal of Communication Systems* 2013; **26**(1):13–33.
14. Yang S-R, Leong C-W. A conformance test tool for next generation network applications and systems. *International Journal of Communication Systems* 2010; **23**(6-7):708–731.

DOI: 10.1002/dac

15. Yoo S, Harman M. Regression testing minimization, selection and prioritization: a survey. *Software Testing, Verification and Reliability* 2012; **22**(2):67–120.
16. Lin Y-D, Chou C-H, Lai Y-C, Huang T-Y, Chung S, Hung J-T, Lin FC. Test coverage optimization for large code problems. *Journal of Systems and Software* 2012; **85**(1):16–27.
17. Kolawa A, Huizinda D. Automated defect prevention: best practices in software testing. *IEEE Computer Society Press* 2007; 254.
18. Woodward MR, Hennell MA. On the relationship between two control-flow coverage criteria: all JJ-paths and MCDC. *Information and Software Technology* 2006; **48**(7):433–440.
19. Gcov. Available: http://gcc.gnu.org/onlinedocs/gcc/Gcov.html (accessed April 2013).
20. Roesch M. SNORT: lightweight intrusion detection for networks, *Proceedings of the 13th USENIX conference on System administration*, Seattle, Washington, USA, 1999.
21. Linux Kernel. Available: https://www.kernel.org (accessed April 2013)

DOI: 10.1002/dac