Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Review

A survey on network data collection

Donghao Zhou[a], Zheng Yan[a,b,*], Yulong Fu[a], Zhen Yao[a]

[a] State Key Laboratory of ISN, School of Cyber Engineering, Xidian University, Xi'an, China
[b] Department of Communications and Networking, Aalto University, Espoo, Finland

ARTICLE INFO

Index Terms:
Intrusion detection
Attack detection
Network data collection
Network management
Network security
Packet capture

ABSTRACT

Networks have dramatically changed our daily life and infiltrated all aspects of human society. At the same time when we enjoy the convenience and benefits brought by the networks, we also suffer from a great amount of intelligent attacks and malicious intrusions. As a fundamental procedure of network security measurement, network data collection executes real time network monitoring, supports network performance evaluation, assists network billing, and helps traffic testing and filtering. Thus, it plays a crucial and essential role for dealing with network intrusion detection and unwanted traffic control. But an adaptive and effective data collection mechanism that can be pervasively applied into heterogeneous networks is still lacked. The literature we have hunted rarely comments and compares the performance of existing data collection mechanisms. In this paper, we conduct a survey on existing data collection methods, mechanisms and architectures. According to a number of proposed assessment criteria, we evaluate the performance of existing data collection mechanisms and summarize their characteristics. Furthermore, we figure out some open issues based our investigation and forecast future research directions.

1. Introduction

Networks have dramatically changed our daily life and infiltrated all aspects of human society. Every day, we use a wide variety of network services and applications, which produces a huge amount of network data. Although most of the data generated in the networks are meaningless to us, a part of them contain useful and sensitive information that should be well collected, protected and managed.

On the other hand, at the same time when we enjoy the convenience and benefits brought by the networks, we also suffer from a great amount of intelligent attacks and malicious intrusions. Various security threats are cause by different types of attacks, e.g., Denies of Service (DoS), Distributed Denies of Service (DDoS), viruses, wormhole, and password guessing or stealing attacks. For detecting these attacks, some network data should be collected in order to figure out network vulnerabilities. According to the vulnerabilities, network administrators can take corresponding actions, recover network functions, predict future network threats and enhance network security and robustness.

Network data collection can greatly help network attack detection and assist network administration. Through real-time monitoring, testing, configuring, controlling and evaluating based on network data, network administrators are able to obtain network system performance, evaluate Quality of Service (QoS) and find out network fault points. For Internet Service Providers (ISPs), network data plays the basis of traffic accounting. Statistical volume information of traffic impacts the policy of ISPs.

With the acceleration of the 5G technologies and the promotion of the Internet of things (IoT) services, large scale and high-speed networks become the focus of current research and development. In order to collect and analyze network data effectively, researchers and operators proposed a lot of systems and applications. Though numerous surveys of network traffic analysis were published, there are few investigations on network data collection (Liu et al., 2018; He et al., 2018). This survey focuses on network data collection to make up for this missing study.

For specific network scenarios and specific collection puposes, the requirements of network data are different. Thus, in the process of network data collection, it is not necessary to collect all available data from the networks (Lin et al., 2018). Since data collectors are required to collect useful information, useless and meaningless information should be dropped. Redundant information should be fused. Then the reserved data can be aggregated to generate useful features, which serve as the basis for attack identification, intrusion detection, and furthermore network security measurement.

Regarding network data collection for security measurement, some important types of data should be normally collected. Network packet is still the most common data format in current network environments. Thus, network packets are usually considered as the main objectives

* Corresponding author. State Key Laboratory of ISN, School of Cyber Engineering, Xidian University, Xi'an, China.
E-mail addresses: dhzhou@stu.xidian.edu.cn (D. Zhou), zyan@xidian.edu.cn, zheng.yan@aalto.fi (Z. Yan), ylfu@xidian.edu.cn (Y. Fu), 695154820@qq.com (Z. Yao).

https://doi.org/10.1016/j.jnca.2018.05.004
Received 18 December 2017; Received in revised form 9 April 2018; Accepted 7 May 2018
Available online 17 May 2018

that should be investigated in the field of network data collection. But, existing approaches of packet data collection usually suffer from packet loss, especially when coming across overwhelming traffic (Morariu and Stiller, 2008). And for high-speed lines, existing approaches are usually becoming useless because of substandard capability. Flow is a group of packets with same features. Commonly, the five-tuple features, including source and destination Internet Protocol (IP) addresses, source and destination ports, protocol types, are the features of the packets. A flow-based data collection mechanism, as an alternative of a packet-based mechanism, screens the flow rather than all packets (Lee et al., 2014; Kundu et al., 2009). The flow-based data collection mechanism reduces the tasks in packet analysis and performs much better than the packet-based mechanism in gigabit networks. However, it lacks fraction fidelity because of packet and flow filtering. Log file is a data storage form widely distributed in network devices. As one of the ways in data inspect, log analysis utilizes abundant log resources, such as system logs, device logs and Web logs, to extract and parse valuable information. However, it is troublesome to implement log analysis because log files are always with huge data quantity, low information density and disordered formats (Oliner et al., 2011). In the literature, there are various kinds of other data formats and collection mechanisms (Morariu and Stiller, 2008; Oliner et al., 2011; Lee et al., 2014; Kundu et al., 2009), with different pros and cons. Due to the importance of network data collection, it is essential to review the state-of-the art in order to summarize its current advance and figure out open issues for future investigation.

There are a number of existing surveys about network data collection mechanisms researched and deployed in network architectures. For instance, Sperotto et al. made a survey about IP flow data collection mechanisms (Sperotto et al., 2010). Xu et al. (2016) compared and analyzed collection mechanisms regarding Deep Packet Inspection (DPI). Other surveys focus on investigating data collection mechanisms (Davis and Clark, 2011; Moindze and Konate, 2014; Callado et al., 2009; Lin et al., 2018) with limited research scopes, so that researchers and practioners are hard to find a mechanism to satisfy their working purposes. The literature still lacks a thorough survey on network data collection that summarizes previous results by evaluating their performance with uniform criteria in order to instruct future research.

In this survey, we made a comprehensive review on network data collection mechanisms. We first introduce the types of data carriers for data collection. Then, we propose a number of criteria for evaluating the performance of different data collection mechanisms. By employing the proposed criteria, we review the current state-of-art in order to summarize current advance, find open issues and direct future research. Specifically, the contributions of our paper can be summarized as below:

1 We propose a series of criteria to evaluate current network data collection mechanisms.
2 We thoroughly review existing network data collection mechanisms and analyze their advantages and disadvantages by employing the proposed criteria as an evaluation measure.
3 We figure out a number of open issues and indicate several promising directions to instruct future research.

The rest of the paper is organized as follows. Section 2 introduces relevant knowledge of network data formats and data collection. Section 3 proposes and justifies the evaluation criteria of data collection. Section 4 provides a classification of different data collection mechanisms, followed by a review on the current state-of-art by employing the criteria as an evaluation measure in Section 5. Then we figure out open research issues and propose future research directions in Section 6. Finally, a conclusion is drawn in the last section.

2. Overview of data carriers and data collection

This section briefly introduces the carriers of network data. They are significant for data collection mechanisms. Packets, flows, logs are widely used in mainstream data collection mechanisms. Besides, some network components, such as the controllers of Software Defined Network (SDN) implement and assist data collection. The data collection mechanisms monitor data flow locally and record available information for network quality measurement, traffic estimation and attack prevention. In what follows, we will briefly introduce three basic types of network data collection methods.

2.1. Packet based data collection

Packet is a very significant data carrier in the networks based on the TCP/IP protocol. In a packet exchanging network, effective information is divided and encoded into packets. A source node sends packets that include source and destination addresses to a destination node. When the destination acquires the packets, decoding and aggregation are executed to get expected data.

The packet has various formats according to the types of networking protocols. The packet commonly consists of two parts: packet header and its payload. The header plays the role to guide the packet to transmit in a network and mark the source information of the packet. In many data collection methods, the header becomes important to identify and filter packets. For example, some header-based methods (Davis and Clark, 2011; Kim and Reddy, 2008) classify the packets into multiple flows according to their IP addresses, ports and protocols contained in the header. The payload contains the data exchanged between communicating parties, though some of them could be encrypted.

Packet capturing is a traditional method for information acquisition in network management. It is also the most commonly used scheme (Qadeer et al., 2010; Ficara et al., 2008; Morariu and Stiller, 2008; An and Liu, 2016; Antichi et al., 2012) to accomplish the goal of network data collection. A libpcap function library provides the capability to collect all the contents of a packet flow. Nevertheless, with the wide network access from mobile devices and the popularity of cloud services, high-speed and large-scale network systems become common. The volume of network will overwhelm packet capturers. Though some improved mechanisms were presented, there is still a dilemma with regards to packet capturing. It is therefore essential to consider alternatives to packet capturing mechanisms. Some existing researches (Zhao et al., 2007; Zhang et al., 2009; Kamiyama and Mori, 2006; Ji et al., 2009) abandon the idea to capture all packets, but to adopt packet sampling mechanisms. Simple sampling and stratified sampling are two instances of them. The simple sampling mechanism randomly extracts packets from all the traffic, while stratified sampling classifies the packets and drops some packets according to groups. Moreover, some other sampling mechanisms were also proposed to suit for real-time traffic data collection.

2.2. Flow based data collection

Network flow collection is another important way for network data collection. Flow is a set of packets with the same characteristics passing through a specific observation point over a period of time. Network flow monitoring can occur at every location of the network. But network core devices are the most effective nodes to be monitored and controlled since these devices can obtain important data about cyber threats and attacks. Therefore, flow collection at network core devices is the most prevalent data collection mechanism currently. Flow collection also exists in network edge nodes and hosts. Contrary to core devices, hosts only monitor the flows pass through the hosts and collect the flow records accordingly. In edge nodes and gateways, network flows of inside switches and hosts are monitored. By applying the monitoring and collection mechanisms, inbould and outbound flow streams in a Local Area Network (LAN) can be recorded and further used for network quality measurement and evaluation. Especially, this data collection mechanism is effective to monitor the flow of distributed terminals, such as hosts in a cloud data storage system (Mann et al., 2013).

The most common mechanisms of flow detection are based on five tuples including source and destination IP addresses, source and destination ports, and protocol types. NetFLow (Hofstede et al., 2013; Elsen et al., 2015) is one of the main protocols to implement five-tuple flow data collection mechanisms. The protocol handles the first packet of a kind of flows and generates corresponding caches. When another packet comes into the collector, its policies such as access control are inherited from previous packets in the same flow. The NetFLow mechanism simply concerns packet headers, but it does not consider packet payloads.

However, flow is not only defined as a 5-tuple flow. An ingress and egress filtering mechanism is another mechanism to collect and monitor flow data. Distinct from 5-tuple flows, ingress and egress filtering mechanisms focus on the whole packet headers. The mechanisms are usually deployed in edge nodes and extract features adequately from packet headers. And then these packets can be classified into diverse flows according to the features. Meanwhile, Deep Flow Inspection (DFI) technology provides an effective accomplishment. It is based on flow behaviors to classify flows. The data collection mechanisms based on flow detection show advance in high-speed networks. However, information integrity is influenced by flow filtering.

Although traditional networks are widespread, their architectures bundle control plane and data plane together. Each device has both the capabilities of data forwarding and logic control. Besides, network configurations are customized by its vendors based on underlying network requirements. When the network configurations need to be updated or revised, a network operator has to change each individual node's configurations with specific commands (Kreutz et al., 2015). This situation could become even worse if there are several device vendors with different unique configuration commands. Software Defined Network (SDN) is an emerging paradigm proposed to optimize the complex structures of traditional networks. As show in Fig. 1, the SDN architecture comprises of three planes: data plane, control plane and application plane (Yao and Yan, 2016; Bian et al., 2016). It implements network architecture by separating the functions of controlling and forwarding. When a networking strategy is defined in the application plane, the control plane enforces the strategy and the data plane implements it by executing corresponding forwarding and dropping actions of network traffic.

OpenFlow is the most notable protocol and Application Programming Interface (API) to connect the control plane and the data plane in SDN. The concept of flow in an OpenFlow architecture refers to a series of flow tables. Every flow table is a group of table entries to define the operations on the packets received. All of packet actions, such as packet forwarding, dropping, matching and classification are directly executed in the data plane. The control plane manages the underlying data planes with specific messages. These messages are sent not only from the controller to the data plane, but also from the data plane to the controller. For example, when the controller wants to modify one of the flow tables, a modify-state message is sent to the OpenFlow switch. Then the switch responses the command and modifies the corresponding tables accordingly. If some special requirements are executed or some errors occur, messages will be fed back to the controller. According to the structure of the SDN, it is not a challenge to collect data in this network. In SDN, each flow entry in the flow table specifies a rule. Hence, it can arbitrarily achieve complex linear policies when there are sufficient flow entries. When a packet is captured, it is classified into a group according to matched rules in the data plane (Yan et al., 2017; Huang et al., 2014; Zhang et al., 2016). The group is also called flow and the SDN-based data collection is also a type of flow-based data collection. Yao and Yan studied using trusted computing technology to collect trusted data at the data plane for trust management on the applications in the application plane and for selecting trustworthy flow control policies (Yao and Yan, 2018). However, researchers have not contented with a generic data collection mechanism in SDN networks.

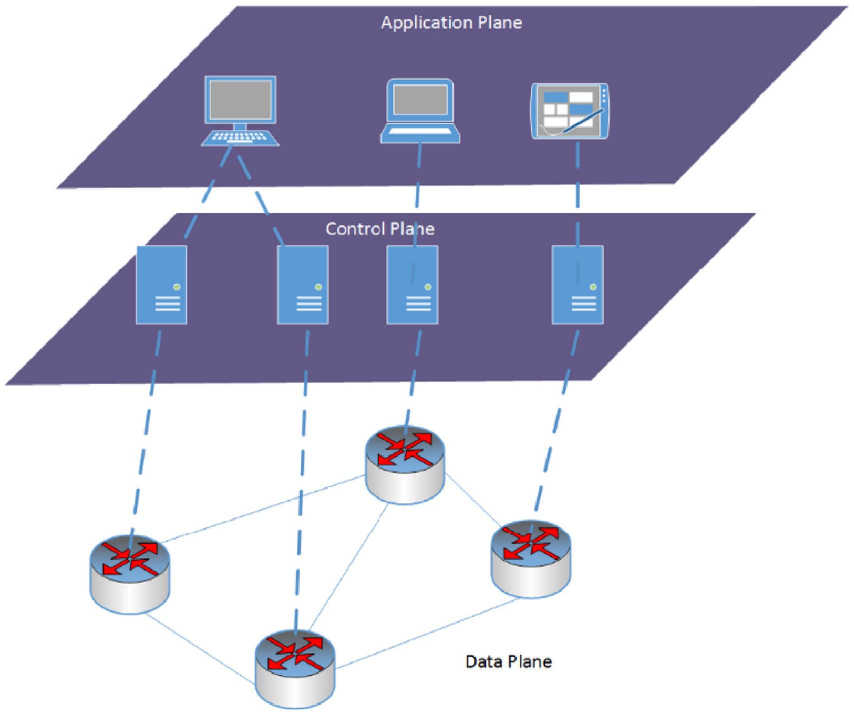

Fig. 1. SDN architecture.

2.3. *Log based data collection*

Log file is a data format widely used in a recording system for network events. The log can be comprised of event log and message log. The event log records user traces, event status and diagnosis failures if necessary. When a service is turned on, its log file is therewith created. Due to user privacy concern, the message logs including Internet Relay Chat (IRC), Instant Messaging (IM), etc., are generally encrypted by service providers. The event logs are commonly used for data collection. According to the sources of logs, they are comprised of operating system logs, Web logs and equipment logs. Log files have no standard formats. And counting how many types of log files in each class is too hard to be drawn up. However, there are some generic features of logs. For example, while a routine is running, each line of the log writes down the information to record, the date, the exact time, the operator and the action in appropriate places.

Log detection is another mechanism applicable for data collection. Contrary to other collection mechanisms, the log files mostly have stored in persistent storages. However, the log files usually occupy a large memory, has low information density and applies complex file formats. Thus, it is hard to handle log files manually and extract valid features from each independent log. To solve these tricky problems, automatic and adaptive solutions are proposed in previous work (Yan et al., 2012; Qiu et al., 2010; Shi et al., 2012; Moh et al., 2016; Asselin et al., 2016). When collectors have probed real-time logs from probes or acquired files from storages, filtering components parse the contents of logs to extract essential features. LogParser (Yan et al., 2012) is a simple log screening tool to accomplish an ordinary log analysis task of several log files. The most common approaches in these processes are pattern matching and machine learning methodologies.

SDN log, especially controller log, is better-organized than the logs in other networks. Thus, log-based data collection mechanisms are proposed in SDN. Meanwhile, management related data and statistical data generated and aggregated at the control plan can also be collected. Similar to Deep Packet Inspection (DPI) mechanisms in the Internet, the log-based data collection suits for SDN to detect protocol features. Log collectors filter useless data and aggregate features from raw logs probed from the interfaces in SDN.

3. Evaluation criteria

In this section, we will propose a series of criteria and explain their necessity for achieving network data collection with high quality. We aim to employ these criteria to set up a common measure for comparing different data collection mechanisms.

3.1. *System performance*

3.1.1. *Instantaneity (IN)*

For almost all data collection mechanisms, collection instantaneity is a critical factor that should be ensured. It is the most direct standard to reflect data collection performance. When a mechanism is deployed in a Low-Speed (LS) network, the data collection process could be immune from resource occupation, such as CPU load and consumed storage. The mechanism can mostly perform well in such an operating environment in theory. But for a High-Speed (HS) and high-load network, an inefficient mechanism is lethal for data collection. It could lead to intolerable information loss, which impacts the accuracy of sequential data analysis and processing.

3.1.2. *Effectiveness (EF)*

In a large-scale and high-speed network, effective and accurate data collection mitigates the volume of data and optimizes data for further analysis. Otherwise, false-positive data could mislead the process of data collection, which makes prior expert knowledge incredible. Because aggressive and harmful attacks could reach internal networks, node compromise and performance abnormity are prone to be brought out. As a result, internal attacks tend to negatively impact network data collection through wrong information provision by compromised nodes. On the other hand, false-negative data could be discarded hence information integrity cannot be ensured. In the case of using the data provided by network users, such a situation becomes equally serious. More importantly, misjudgment of some critical data could cause big loss of users.

3.1.3. *Scalability (SC)*

Scalability requires that the data collection mechanism should suits for the evolution of networks. With the development of network and communication technologies, more and more types of devices and networks connect to the Internet and share network resources. We cannot just propose a mechanism that suits for current network environments. With the increase of network devices, network data collection should support the change of network scale. The scalability of network data collection becomes an important criterion to evaluate the performance of network data collection. Generally, a scalable collection mechanism usually provides reliable external interfaces to bottleneck components for supporting high capability. However, this capability is not unlimited. It is limited to inherent defects of networks, such as bandwidth.

3.1.4. *Expense (EX)*

The expense of network data collection is also an important factor that should be considered in terms of data collection performance. Investigators and operators are prone to develop an effective and low-cost system for data collection. In a large-scale data collection process, each extra component in an individual thread could cause big redundancy. Typically, the expense of data collection consists of development expense and maintenance cost. Accordingly, a data collection mechanism should be well designed by paying attention to the costs of development and maintenance. Deployment difficulty and applying expensive equipment in data collection increase the expense. Trade-off should be made in order to solve this dilemma. Maintenance Expense (ME) heavily depends on update cycle and error probability. For example, a pattern matching method based on expert knowledge needs to update and revise the knowledge frequently for achieving high accuracy, which causes high maintenance expense. However, a precise and adaptive mechanism may decrease such cost due to the need of little maintenance.

3.1.5. *Network performance (NP)*

Obviously, network performance is of great significance for both network users and network management. Bad performance affects the efficiency of networking and may make some services unavailable. Generally, data collection mechanisms affect network performance in following ways:

1 They could cause Invasive Taffic (IT).
2They could use Extra Transmission (ET) paths.

With redundant interventions of network devices and extra traffic caused by data collection, network performance is prone to be worsened. Every time an operator sends testing traffic, network load is increased. Especially in large-scale data collection, a lot of routines perhaps overwhelm the intervened network. Thus, a complete non-invasive data collection mechanism is considered as ideal since it does not affect the network performance although it is hard to achieve it. Data collection mechanisms could increase the length of transmission paths between communication nodes. Besides, extra nodes set for data collection may become the bottleneck of the network. As a result, we draw two sub-criteria, non-invasive traffic and no extra network topology, for performing progressive evaluation on the performance of a data collection mechanism.

Table 1
WBEM components.

WBEM components	Function
CIM	The model that manages resources and presents the standard format MOF;
CIMOM	The database of instances of the CIM class;
XML API	The API that provides standardized access to data and enables operations for managed resources displayed in CIMOM
MOF	The formal description of classes and associations derived from the CIM model
standard module	The collection of CIM classes that are used to represent a special management domain

3.1.6. Resource occupation (RO)

Resource occupation is another criterion to evaluate the quality of a data collection mechanism. An excellent mechanism usually has the capability to control CPU consumption, memory usage and bandwidth consumption. Excessive occupation of resources in data collection may influence or even suspend other threads. While multiple data collection threads are running at a same node, low resource occupation ensures the execution of numerous threads at the same time.

3.1.7. System security (SS)

System security requests that a data collection system cannot be compromised by any attacks. Only a legal party can operate the collected data in an authorized way. Otherwise, corresponding alarms should be raised. It ensures the robustness of the collection system so that its data collectors accomplish their tasks without falling into any security threats.

3.1.8. Adaptability (AD)

An ideal data collection mechanism should be adaptive to networking contexts. Otherwise, it is difficult to accurately collect needed data in numerous and disparate circumstances. According to different network environments, self-learning and self-adjusting abilities are expected to ensure expected collection performance.

3.2. Information security

Information security becomes essential when we talk about collected data. In what follows, we summarize the concrete criteria to ensure the security of collected network data.

3.2.1. Integrity (II)

In the process of network data collection, the collector must have the capability to ensure data integrity. When the data is transmitted in the network, benign network data should not be illegally updated, deleted, tampered or discarded. Otherwise, network users are not able to get original information, especially sensitive information.

3.2.2. Confidentiality (CO)

There is abundant sensitive or private information contained in the collected network data. For some data providers, they are unwilling to let private information open to the public. On this account, the devised data collection mechanism should have the capability to prevent the collected data from being illegally accessed and operated (Zhang et al., 2017).

3.2.3. Availability (AV)

Availability is a user-oriented security requirement. For network and information users, information availability is the first thing for both secure data collection and collected data. It is meaningless for users if the data collection mechanism is not available. However, many unpleasant issues such as high packet loss and long-time delay greatly worsen data collection performance. Thus, we should take availability into consideration when we discuss the performance of a data collection mechanism.

3.2.4. Traceability (TR)

Traceability refers to the capability that a network node can identify the source of invaders when malicious traffic is discovered. It is a necessary requirement for a robust system. In the process of data collection for intrusion detection, a system with traceability usually performs more accurately than other systems. Furthermore, with the improvement of a network legal system, network operators should be accountable for their own network behaviors. Network tracing is a very practical measure to supervise network behaviors.

3.2.5. Authenticity (AU)

Authenticity ensures that the collected data is the real data sourced from networks. There are many intruders that forge traffic at different network locations. It is also possible that the collectors can be compromised. The forged network data or the data provided by the compromised collectors badly influence the accuracy of data analytics, e.g., for detecting an attack. This criterion requires the data collection mechanism can identify forged and abnormal data during collection.

4. Network data collection

In this section, we will singly review the state-of-art of the network data collection mechanisms proposed in recent twelve years. We refer to papers about network data collection or attack detection from the following databases: ACM digital library, Spring digital library, IEEE digital library and Elsevier ScienceDirect. We commented reviewed mechanisms by employing the criteria proposed in Section 3. A summary of their performance is provided in Tables 2–4.

4.1. Packet-based data collection

Network packet, which is significant in the TCP/IP protocol, carries great amount of information. In daily network management processes, packets are widely used in the fields of fault correction, configuration control, performance management, security monitoring and billing recording. Normally, monitored data in packets contain the contents as described below (Zeng and Wang, 2009):

1 Static information: hardware and software parameters, users and administrators, registration information, etc.
2 Dynamic information: CPU consumption and memory usage, interface traffic, etc.
3 Network service information: network protocols such as HTTP, FTP, TCP and some defined network functions such as DNS, SQL requests and responses.
4 Network performance information: packet loss, time delay, bandwidth consumption, etc.

Generally, some convenient network packet capture tools are chosen for data collection and further analysis. WireShark (Das and Tuna, 2017) and TCPdump (Therdphapiyanak and Piromsopa, 2013) are two classic applications of them. They rely on libpcap library and Berkeley Packet Filter (BPF). Furthermore, some Network Intrusion Detection Systems (NIDS), such as Snort and Bro similarly collect data by packet capturing based on libpcap and BPF for detecting malicious traffic and further processing. WireShark and TCPdump tools are passive software-based packet capturing mechanisms.

There are diverse approaches to capture packets, which can be classified as active data collection methods and passive collection methods. The active data collection approaches often inject test data into traffic and waits for responses to achieve the purpose of network quality measurement, while the passive data collection mechanisms

Table 2
Performance of packet-based data collection mechanisms.

Research Work		System Performance												Information Security				
		IN		EF		EX		NP		RO	SC	SS	AD	II	CO	AV	TR	AU
		LS	*HS*	*LS*	*HS*	*DE*	*ME*	*IT*	*ET*									
SNMP Mechanisms	(Yu et al., 2008)	✓		✓		✓		×	✓	✓	*	*	×	*	*	✓	*	*
	(Hillbrecht and Bona, 2012)	✓		✓		✓		×	✓	✓	*	*	*	*	*	✓	*	*
WBEM Mechanisms	(Sundaram et al., 2006)	✓		*		*		*		*	✓	*	✓	✓	✓	✓	×	×
	(Hutter et al., 2009)	✓		*		✓		*		×	*	*	✓	✓	✓	✓	×	×
Normal Passive Packet Mechanisms	(Qadeer et al., 2010)	✓	×	✓	×	✓		✓	×	×	×	×	×	×	✓	×	*	*
	(Papadogiannakis et al., 2007)	✓	×	✓	✓	✓		✓	×	×	×	*	*	*	*	*	*	*
	(Ficara et al., 2008)	✓	×	✓	✓	×	*	✓	✓	×	*	*	✓	✓	*	✓	*	*
DPI Mechanisms	(Meiners et al., 2012; Turner et al., 2013)	✓		✓		*		*		×	✓	*	*	✓	✓	✓	*	*
Sampling Mechanisms	(Zhang et al., 2009; Kamiyama and Mori, 2006)	✓		*		✓		✓	*	✓	*	*	×	*	*	*	*	*
	(Zhao et al., 2007)	✓		✓		✓		✓	*	✓	*	*	×	*	*	*	*	*
	(Ji et al., 2009)	✓		✓		✓		✓	*	✓	*	*	✓	*	*	*	*	*
Distributed Mechanisms	(Morariu and Stiller, 2008)	✓		×		*		✓	*	*	✓	*	✓	*	*	*	*	*

IN: Instantaneity; LS: Low-Speed networks; HS: High-Speed network; EF: Effectiveness; EX: Expense; DE: Development Expense; ME: Maintanance Expense; NP: Network Performance; IT: Invasive Taffic; ET: Extra Transmission; RO: Resource Occupation; SC: Scalability; SS: System Security; AD: Adaptability; II: Integrity; CO: Confidentiality; AV: Availability; TR: Tracability; AU: Authenticity.
√: corresponding mechanism supports this property.
×: corresponding mechanism does not support this property.
*: corresponding mechanism does not mention or consider this property or its support is not sure.

usually monitor network traffic with monitoring tools either software or hardware. Most existing approaches deploy appliances in networks, which could affect original traffic.

4.1.1. *Active packet probe mechanisms*

Active packet probe is an active data collection mechanism for packet capturing. It injects test traffic into normal network traffic for network quality measurement. Then, the quality of networking can be judged and evaluated according to the response of the network. It is a kind of convenient and controllable approach. "Trace out" and "Ping" commands are two non-cooperative methods frequently used. However, the most prevalent active mechanism is to use some network management protocols as reviewed below.

4.1.1.1. SNMP based collection mechanisms. Simple Network Management Protocol (SNMP) based methodology is a typical active packet-based data collection mechanism. Besides, it is one of the most influential protocols in the Internet management. As shown in Fig. 2, a regular SNMP architecture usually consists of a number of management agents, Management Information Bases (MIBs), a management station and a database. When a user is desired to acquire the data in a specific management agent, the management station transports User Datagram Protocol (UDP) packets to an inherent port. Typically, port 161 is used in the SNMP protocol. When a SNMP data stream flows in, the management agent executes the processes of decoding, verification and querying the managed variable in MIB trees. Accordingly, the agent responds the request from the user. Therefore, the management station can get the requested data after acquiring the responding information (Goncalves et al., 2009).

Yu et al. implemented a mechanism to detect traffic flooding attacks by collecting SNMP MIB information (Yu et al., 2008). The MIB data collector acquires the data with active SNMP messages. In this way, a collector can easily identify attacks from normal traffic by using their implied features. In this mechanism, the collector does not collect data all the time. Only when a user asks for reply, the system needs to re-sponse it. It greatly reduces the overload of collectors. According to experimental results, its accuracy of detection rate can reach 99.40%. Thus, it fulfills EF. And the time to detect attacks is very short, thus satisfying IN. But this approach is unwieldy due to heavy dependence on MIB items, which are previously added and hard to be adaptively

Table 3
Performance of flow-based data collection mechanisms.

Research Work		System Performance												Information Security				
		IN		EF		EX		NP		RO	SC	SS	AD	II	CO	AV	TR	AU
		LS	*HS*	*LS*	*HS*	*DE*	*ME*	*IT*	*ET*									
Five-tuples Mechanisms	(Bajpai and Schönwälder, 2017; Sekar et al., 2008)	✓		✓		✓		✓		×	*	*	*	*	*	*	*	*
	(Bo et al., 2005)	✓		*		✓		✓		✓	*	*	*	*	*	*	*	*
Ingress and Egress Filtering Mechanisms	(Davis and Clark, 2011; Early and Brodley, 2006)	✓		×	✓	*		✓		×	*	*	*	*	*	*	*	*
DFI Mechanisms	(Xia and Song, 2009)	✓		✓		✓		*		*	*	*	*	✓	✓	✓	*	*
OpenFlow Mechanisms	(Foster et al., 2011)	✓		*		*		*		*	✓	*	*	*	*	✓	✓	*
	(Bari et al., 2013)	✓		✓		*		*		✓	✓	*	✓	✓	✓	✓	✓	*

IN: Instantaneity; LS: Low-Speed networks; HS: High-Speed network; EF: Effectiveness; EX: Expense; DE: Development Expense; ME: Maintanance Expense; NP: Network Performance; IT: Invasive Taffic; ET: Extra Transmission; RO: Resource Occupation; SC: Scalability; SS: System Security; AD: Adaptability; II: Integrity; CO: Confidentiality; AV: Availability; TR: Tracability; AU: Authenticity.
√: corresponding mechanism supports this property.
×: corresponding mechanism does not support this property.
*: corresponding mechanism does not mention or consider this property or its support is not sure.

Table 4
Performance of log-based data collection mechanisms.

Research Work		System Performance										Information Security						
		IN		EF		EX		NP		RO	SC	SS	AD	II	CO	AV	TR	AU
		LS	*HS*	*LS*	*HS*	*DE*	*ME*	*IT*	*ET*									
Pattern Matching Mechanisms	(Yan et al., 2012)	*		×		✓	×	✓	×	×	×	*	*	*	*	*	*	*
Machine Learning Mechanisms	(Qiu et al., 2010)	×		✓		×	✓	×	✓	×	✓	*	✓	*	*	*	*	*
Combing Mechanisms	(Moh et al., 2016; Asselin et al., 2016)	×		✓		×		×	✓	×	*	*	*	*	*	*	*	*
IA Mechanisms	(Shi et al., 2012)	×		✓		*	×	×	✓	×	*	*	*	*	*	*	*	*
SDN Logs Mechanisms	(Siniarski et al., 2016)	×		✓	*	✓	×	✓		×	✓	*	*	*	*	*	*	*

IN: Instantaneity; LS: Low-Speed networks; HS: High-Speed network; EF: Effectiveness; EX: Expense; DE: Development Expense; ME: Maintanance Expense; NP: Network Performance; IT: Invasive Taffic; ET: Extra Transmission; RO: Resource Occupation; SC: Scalability; SS: System Security; AD: Adaptability; II: Integrity; CO: Confidentiality; AV: Availability; TR: Tracability; AU: Authenticity.
√: corresponding mechanism supports this property.
×: corresponding mechanism does not support this property.
*: corresponding mechanism does not mention or consider this property or its support is not sure.

used in different scenarios. If an attack is unknown for databases, it will be hard to be detected. Besides, the mechanism may impact network performance since it increases the overhead of network.

With the development of cloud computing and Network Functions Virtualization (NFV), extra functions are required to support virtual machines. In (Hillbrecht and Bona, 2012), Ricardo et al. proposed a SNMP-based virtual machine management interface. It provides a solution for network managers to perform both monitoring and controlling, such as create, delete, restart, turn on, pause and shut down virtual machines. It helps operators to collect data in virtual environments. The same as Yu's mechanism, this mechanism also has good IN and poor NP due to the characters of SNMP. The authors conducted a set of experiments to evaluate the performance. The result shows that this mechanisms takes low RO in terms of memory and CPU.

4.1.1.2. WBEM-based collection mechanisms. As a novel standard of network management system, Web-Based Enterprise Management (WBEM) implements a cross-platform and resource-independent architecture to collect data for network performance and QoS evaluation. As show in Table 1, this protocol generally comprises of several components: Common Information Model (CIM), a CIM database named CIM Object Manager (CIMOM), eXtensible Markup Language (XML) API, Managed Object Format (MOF) and a standard module to stratify and manage functional domains. A CIM mechanism adopts an object-oriented technology to operate variables and manage sources flexibly. In order to realize the capability of communications between distributed and heterogeneous network devices, a standardized schema is applied. A WBEM client, a device that can also be considered as an individual manager, can conveniently get the operability to remote devices if permitted by headquarters. The

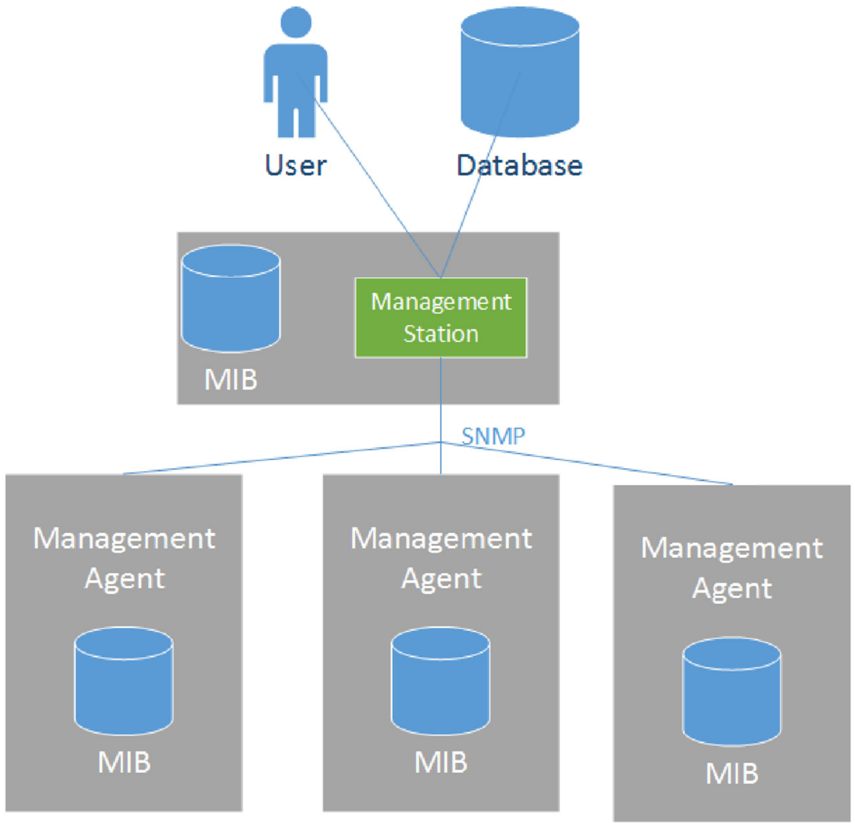

Fig. 2. A general SNMP model.

manager generates corresponding MOFs for the association between classes according to wanted data. Then the manager needs to search dynamic providers for classes that require dynamic data updating. If providers are found, the manager imports MOF into CIMOM and implants static data required by class instances. Finally, the operator visits managed data by an API of CIMOM (Goncalves et al., 2009; Michaut and Lepage, 2005).

Sundaram et al. (2006) proposed a WBEM-based network management system for inter-AS (Autonomous System) traffic collection. When two ASs are communicating, there are two agent nodes, which also perform as normal clients, distributed in each AS respectively for centralized transmission. Thus, two distant clients can exchange data across several ASs. In this process, each AS is controlled by a manager. It is convenient to collect data from the managers and the destination clients accordingly. Because WBEM stores information in some open source websites, II, CO and AV are easily satisfied. While the online platform does not care about the source of the information, TR and AU are hardly satisfied in this mechanism. According to the results of experimental evaluation, the number of instances enumerated seldom influences running time, which implies great SC and AD of the system. This mechanism also spends little time in connection establishment and indication notification, so IN can be easily guaranteed.

The WBEM method is not only applied to manage large heterogeneous networks, but also suited for miniature embedded networks. Hutter et al. (2009) proposed an embedded WBEM-based data collection and management system. The on-chip prototype platform requires low resource occupation due to limited resources of clients. The same as Sundaram's solution, this mechanism satisfies in II, CO and AV, while it does not perform well in terms of TR and AU due to the characters of WBEM. According to the result of the performance evaluation, the mechanism performs great with regard to IN, AD and requires low memory, because the mechanism performs even better than Personal Computer (PC) in CIM operations, which requires only 900 kB of non-volatile memory. However, it is likely to perform badly in terms of RO because its inefficient network driver may be completely blocked by requests.

Obviously, active probe mechanisms are convenient and effective to acquire network data and statistical information. It is also a cheap and available data collection mechanism that can be widely used in small scale networks. However, with rapid increase of the number of network devices and the high demand of network speed, the active data collection mechanism becomes improper. A lot of extra traffic is produced by the monitors to acquire responses. It is not necessary and intolerable to take up a tremendous amount of bandwidth. Thus, this approach is seldom used in high-speed networks. How to erase or reduce network influence caused by active data collection mechanisms becomes an urgent issue to solve.

4.1.2. Passive packet collection mechanisms

Generally, packet-based mechanisms use sniffers to implement network data collection through centralized management. The most famous packet capture tools WireShark and TCPdump are both packet sniffers. Normally, a packet can be only received by a Network Interface Card (NIC) when the destination Media Access Control (MAC) address of the packet matches the host of NIC. However, when an NIC works in a promiscuous mode, the host receives all packets, even when they are not intended to be sent to it. These kinds of data collection mechanisms are effective and convenient in some situations, especially for individual hosts.

In (Qadeer et al., 2010), Qadeer et al. proposed a network traffic monitoring and analyzing approach using a packet sniffer in host NIC, which is a passive monitoring approach to capture packets. Through Address Resolution Protocol (ARP) spoofing or MAC flooding, the sniffer breaks through LAN restrictions and has the capability to capture packets in the Internet. This approach is very convenient to be deployed in an individual host. Thus, it is not an expensive method. In a small-scale network, it can perform well although immersed probe is harmful for NP. Besides, due to periodical data collection in a large-scale network, RO and IN become obvious challenges in this approach. As a result, II and AV of collected data cannot be guaranteed.

Papadogiannakis et al. proposed an improved mechanism by utilizing locality buffering (Papadogiannakis et al., 2007). This mechanism distinguishes packet gathering phase and packet delivery phase. In a packet buffer, packets are sorted, divided and transmitted based on detected destination ports in an alternate way. As a result, this method mitigates the network influence of sniffer and decreases packet loss. Accordingly, it performs well in terms of EF. But it overly depends on ports that are convenient to be changed. So, SC cannot be guaranteed. Besides, the test experiment shows when traffic speed reaches 100 Mbit/s, the mechanism still has a good performance.

A sniffer is commonly regarded as a convenient and efficient tool to detect traffic and capture packets. Nevertheless, when the Internet users allows network management equipment to mirror their traffic without proper detection methods of sniffer, infiltrators need to perform the same way as a network management administrator so that illegal infiltrators can acquire private information such as user name, password and even more. It is an obvious threat for users. Thus, it is essential to classify intruders and network administrators when both of them extract packets from users.

The aforementioned methods cannot absolutely overcome the problems of NP. An excellent component named cable splitter achieves non-destructive impact on divert traffic in networks. It solves prior problems in terms of NP.

In (Ficara et al., 2008), a cooperative PC/Network-Processor architecture was advocated by using an optical splitter. It describes a mechanism where the two flows in a Gigabit Ethernet optical fiber are both divided into two optical signals: the first signal is scattered to an output fiber while the second passes through the optical splitter. Thus, there is another output signal that copies from original traffic but actually never affects the original one. As shown in Fig. 3, the architecture consists of a network processor, two gigabit Ethernet switches (one for output and the other for input) and a Personal Computer (PC) cluster. The network processor is the core module of the architecture that directly links with the optical splitter. When the optical splitter copies network traffic, the related network processor records the arrival time of each packet, classifies these packets and filters unwanted traffic. Then, it stripes the packet's header to get only necessary information and combines several packets into a batch frame. The batch frame will be sent to the PC cluster. The PC cluster is a group of distributed packet tools for further analyzing traffic data. This paper proposed a scheme to collect data without influencing normal traffic. Through the control of the network processor, this approach gets accurate packet timestamp, simplifies packet header and filters useless packets, which helps PC easily deal with the packets. Comparing with other approaches, this approach shows a great improvement with regard to packet loss rate. Thus, EF and IN can be satisfied. And the network processor can verify II and AV of received packets. However, because of the huge volume of traffic intended for the network processor, its performance becomes a challenging problem. In many cases, it could be too expensive to provide such a network processor.

Although normal mechanisms have many advantages in terms of conciseness, they cannot perfectly meet the requirements of the current complex network environment. Thus, some improved and alternative schemes are considered to improve traditional methods. Sampling mechanisms, distributed mechanisms and Deep Packet Inspection (DPI) mechanisms are three kinds of mechanisms that the investigators and network operators usually apply into network practice. We review these mechanisms as below.

4.1.2.1. Sampling mechanisms. To mitigate the large collection volume of traffic and speed up the acquisition rate in gigabit networks, sampling mechanisms were proposed by some researchers. The

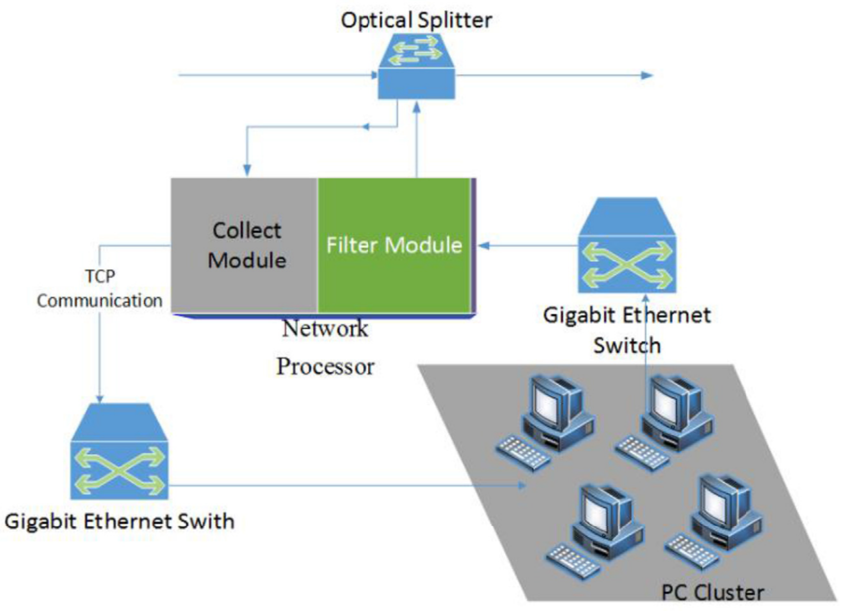

Fig. 3. PC/Network-Processor architecture.

sampling mechanism is a flexible and widely used mechanism in current data collection circumstances. This mechanism commonly utilizes some mathematical statistics theories and rules of traffic forecast to implement the function of filtration. Then, some policies can be applied to the process of real-time data collection. As a result, the mechanism can substantially cut the volume of collected data.

Traditional sampling methods such as simple sampling (Zhang et al., 2009; Kamiyama and Mori, 2006) have been widely used in packet-based data collection. Zhang et al. (2009) proposed a simple random sampling mechanism. It enhances the performance in terms of IN and RO. This mechanism samples data flows in an absolute random way. Obviously, the performance of this mechanism is very accidental. It must tolerate the risk that sampled data can be too anamorphic to reflect reality of original data.

Some improved approaches were investigated. A stratified sampling mechanism (Zhao et al., 2007) proposed by Zhao et al. provides a refined and reasonable packet sampling process. According to calculated data size, the mechanism decides stratums and then samples packets successively. Through corresponding data testing, a lot of attacks can be successfully detected even in high speed backbone networks. So, this mechanism can ensure IN. Besides, the mechanism has less false alarm rate than traditional mechanisms and the ways without sampling. Thus, the mechanism is effective (EF), too. However, like other sampling mechanisms, the mechanism mainly cares about efficiency, but pays little attention to other properties, such as II, CO and AV, etc.

In (Ji et al., 2009), a two-dimensional data collection approach divides data collection process into target data collection and detail data collection. According to the frequency characteristic of a flow, the collector adjusts its collection frequency adaptively. Detailed recourse occupation and network context are considered to decide the final sampling frequency. The same as other sampling mechanisms, this mechanism preforms well in terms of IN and RO. Meanwhile, AD is considered in this mechanism, sampling becomes more intelligent and regular rather than accidental. Accordingly, EF is improved.

Although the information collected is not complete, the sampling mechanism provides a thrifty and available method in high-speed networks. But at the same time, the approach must bear the risk of high error probability, which is considered as the most serious weakness of this type of mechanisms.

4.1.2.2. Distributed mechanisms. Centralized packet capturing mechanisms are effective for low-speed networks. However, with the rapid development of the Internet, gigabits level traffic becomes very common. Accordingly, packet capture capabilities could be easily limited by the storage and CPU speed of capturing nodes. And what's worse, packet delay and rate of packet loss could be increased due to resource limitation. Meanwhile, packet capturing mechanisms often execute their operations via threads, applications, and even different devices. As a result, accuracy and synchronization become challenges in gigabit traffic networks. These problems seriously hinder the performance of packet capturing.

Considering the above concrete problems and the limitation of the traditional network data collection mechanisms, some distributed packet capturing mechanisms become popular. The main idea of this type of mechanism is capturing packets distributedly and concurrently in different nodes. As a result, it can hardly exhaust the CPU capacity of the network. In general, a distributed packet capture system consists of capture modules, packet analysis modules, coordinator modules and some accessory modules.

Morariu et al. (Morariu and Stiller, 2008) depicted a distributed and typical architecture named DiCAP for network data collection, shown in Fig. 4. In the figure, the mirroring device is the core component of the packet capturing architecture. In this architecture, network traffic is mirrored in gigabit Ethernet and sent to each distributed capture nodes. By parallelizing the capture nodes, network traffic is scattered to different capture nodes so that the capturing work load of each single node is decreased. In distributed nodes, a libpcap based packet analyzer is regularly deployed for local analysis in each capture node. When distributed capturers acquire packet data, these analyzers receive data from capturers in order to filter unwanted traffic. Besides, some central analyzers may be arranged for further and precise analysis. A node coordinator is another essential component in this architecture. Control rules are set by node coordinators for controlling the capture processes in capture node clusters. This control can be either distributed for controlling multiple clusters or central for common controlling. According to a set of performance tests, DiCAP architecture performs much better than normal libpcap based mechanisms, especially in a high-speed networking environment. Distributed capture nodes can be easily added into or deleted from the capture system, so SC is well satisfied. However, due to incomplete round-robin selection approaches, capture nodes cannot be led to a perfect load balance. This decreases EF and should be further improved.

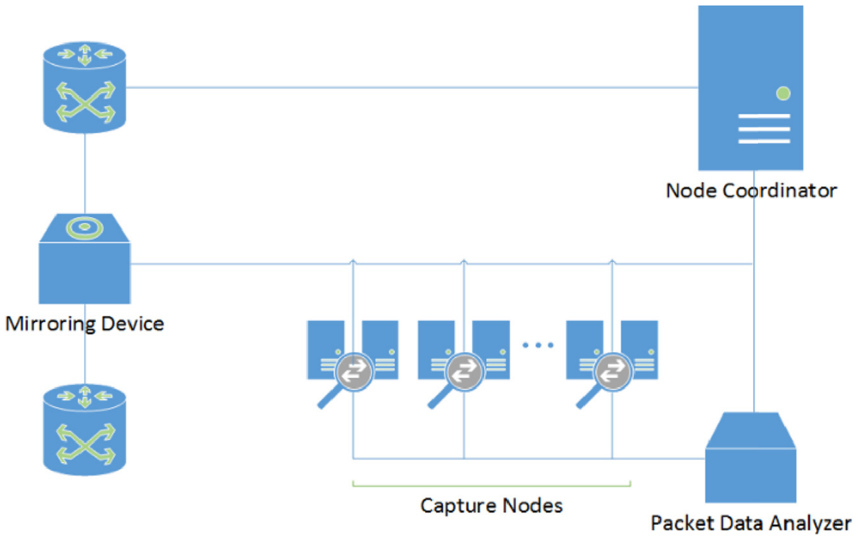

Fig. 4. DiCAP distribute architecture.

The distributed mechanism is nearly an ideal mechanism for large scale data collection. It provides a fast, effective and cheap solution for big volumes of network data collection. However, there still exist some grave problems. For example, while a distributed system collects data in a sharply high speed, a tiny time error will mislead the collection process. Therefore, an accurate and robust time synchronization component is needed for the distributed mechanism, which may be not easy to realize in practice. Besides, central coordinators and managers are required to have strong abilities to manage distributed nodes and identify malicious traffic while remote nodes are prone to be compromised. These issues are worth further investigating.

4.1.2.3. Deep Packet Inspection mechanisms. Most of the packet-based data collection mechanisms are independent from packet contents. By only concentrating on the packet formats, a collector decides the packets that should be maintained. Unlike these approaches, *Deep Packet Inspection (*DPI) technology is a content-aware technology to identify traffic in network application layer (Sekar et al., 2010). This technology can be deployed as an application-level gateway identification mechanism applicable in the situations that control flow and traffic flow are divided. In this situation, it is insignificant to analyze traffic flow. Instead, control flow analysis is necessary to be conducted. In a traditional approach, monitor devices usually collect and detect traffic with ports. Many applications frequently change their ports in an automatic or manual way, hence, using the port information is not effective for collecting expected data.

The DPI mechanism has fine-grained partition for data collection and flow measurement. Generally, DPI architecture can be divided into preliminary pattern matching and feature inspection (Li et al., 2016). The former distinguishes different patterns with predefined rules. The most common technologies are Regular Expressions (REs) and Context-Free Grammars (CFGs). And the available patterns are usually protocol types. The features of protocols are various. They can be port numbers, character words and gateways in the application layer. As we have discussed above, port detection may be useless because application ports are frequently changed. Character words filtration provides a credible alternative policy when port detection mechanism is disabled. By this way, many modern network business and management applications can be deployed such as network billing, advertisement injection and application type identification. However, when the packet payload is cryptographically encrypted or protected, this approach cannot work.

In (Xu et al., 2016), Xu et al. made a survey on REs of DPI technology. They listed corresponding requirements that should be satisfied by DPI mechanisms. It's necessary for REs to exhaust data formats, otherwise some payload contents will be omitted to be recognized. However, hardware resource limitation, ever-increasing link speed and pattern scale are still challenging issues that should be solved.

Meiners et al. (2012) pointed out that RE and CFG mechanisms cannot effectively suit for the contents of different text constructs. Similar to RE approaches, CFG mechanisms are complex because of the limited capability of expression and the ambiguity of some words. In addition, these two mechanisms are both based on manual operations. This implies low reusability and easy inclination to errors due to the increasing diversity and complexity of payloads. Accordingly, they designed a counting automata architecture named FlowSifter to realize layer 7 field data extraction and collection. Their extraction grammar is generated by a grammar optimizer, which integrates information of protocol library and user-friendly extraction specification. This layer 7 field data collection mechanism overcomes the communal dilemma of tedious and inefficient manual processing and achieves the purpose to collect data automatically and adaptively, even the prior knowledge is not sufficient. The experimental tests took three factors: speed, memory and extractor definition complexity into consideration. According to the testing result, this mechanism has good performances in terms all of three factors. Thus, IN and EF can be supported. However, the proposed mechanism still cannot conquer the common problem of DPI, i.e., cryptographical protected traffic is still unreadable.

4.2. Flow-based data collection

Flow is defined as a set of packets having the same features and passing an observation point in the network during a certain time interval. Generally, the five-tuple features, including source and destination IP addresses, source and destination ports, and protocol types, are the most commonly used features of packets. The flow-based data collection mechanism, an alternative of packet-based mechanism, screens the flow so that evades the analysis of all packets. The specific flow can be filtered by a group of features. For example, when we need to collect related data to defend DoS attacks, the flows are filtered. Finally, the remaining flows are used in further analysis.

4.2.1. Five-tuple collection mechanisms

The five-tuple data collection mechanism is the most common method used in flow detection. Two packets can be identified with aforementioned tuples in the same flow. Holonomic NetFlow architecture, the most notable five-tuple flow protocol, includes three main components: flow exporter, flow collector and flow analyzer

(Thangapandiyan and Anand, 2016; Li et al., 2016). Initially, the flow exporter, also known as the observation point, gains access to raw packets on a monitored interface, extracts packets' headers and computes the hash values of the packets' five-tuples. Then, the packets are captured and pre-processed for further use. According to the hash values, packets are aggregated into same flow traffic in Synchronous Dynamic Random-Access Memory (SDRAM). If the flow is terminated or expired, the flow details are sent to the flow collector (Bajpai and Schönwälder, 2017; Babcock et al., 2002; Carli et al., 2009). Its export template is given according to the version of NetFlow protocol. Then, the flow collector captures flow traffic from the corresponding exporter and stores it in a storage backend. Afterwards, the flow analyzer extracts the information from the storage and automatically analyzes the traffic. The mechanism, especially the NetFLow mechanism, has become a working standard. But it silently loses some information because of flow filtering.

Bo et al. (2005) proposed an embedded flow collection mechanism. Embedded flow engines are deployed in routers and switches as exporters. Then, collected data are sent to collectors, i.e., some database servers. In this way, an inexpensive and efficient prototype can be developed in a high-speed networking environment. However, many other criteria, such as EF and AD, were not discussed in this mechanism.

Sekar et al. (2008) presented an improved scheme by calculating the features and the correlation between them. This scheme inherits the characteristics of IN and NP, and optimizes the EF of previous work. It has a great improvement on reducing the sharpness of flow so that the system can hardly be blocked. Thus, the worst-case processing overhead can be reduced and IN can be guaranteed. By comparing with existing packet sampling schemes through simulation tests, Sekar's mechanism performs much better in terms of bandwidth per connection, flow coverage and redundant flow reporting. Thus, NP can be satisfied. Bajpai and Schönwälder (2017) optimized IN and RO for the five-tuple collection mechanism. Comparing to prior mechanisms in data collection, this mechanism costs less time in each stage. But the above two mechanisms did not take information security into consideration.

It is generally believed that five tuples of a packet contain the main information of network links. Existing mechanisms usually have good efficiency and low resource occupation. In a large-scale data collection system, the performance of this kind of mechanisms is generally much better than the packet-based mechanisms. Thus, this kind of mechanisms is usually chosen for realistic data collection. However, this type of mechanisms only concentrates on specific tuples whatever the environment is, which is not so excellent in terms of adaptability.

4.2.2. *Ingress and egress filtering based collection mechanisms*

Flow collection is not only executed at network core elements, but also performed at network edge nodes, such as some network interfaces like firewalls and IDSs. Unlike core devices, these nodes are often deployed for specific terminals or LANs. Some features are not so meaningful due to their inherent relevance. such as a public IP address of hosts in the same LAN. Thus, five-tuple detection is not so valid since the information it contains is not adequate enough. Accordingly, some flow collection functions rely on ingress and egress filtering (Early and Brodley, 2006; Kim and Reddy, 2008). SNORT (Roesch, 1999), a light weight intrusion detection technique, is a simple prototype of flow collection implemented in this way.

A packet header that carries a packet's underlying information is needed in the stage of preprocessing. Packet Header Anomaly Detector (PHAD) (Garg and Maheshwari, 2016), a mechanism that can minimize the volume of preprocessing data, is essential to be deployed in order to mitigate threats of TCP/IP stack, Intrusion Detection System (IDS) evasion techniques, imperfect attack code, and anomalous traffic from victim machines. The basic feature of packet header can identify 33 attributes of each packet (Davis and Clark, 2011). Only when an attribute of the packet header performs in a bad or incorrect way, the mechanism can identify it from normal traffic immediately. This approach provides real-time flittering comparing to postmortem analysis of traffic. Rigorous real-time and on-line analysis is needed to satisfy data collection and analysis process in a gigabit network. Therefore, IN can be satisfied. And its EF is better than that of five-tuples mechanisms.

Apart from packet header basic features, Davis and Clark (2011) also proposed Single Connection Derived (SCD) features and Multiple Connection Derived (MCD) features to detect abnormal traffic from victim machines. Commonly, these approaches use SCD and MCD features to execute postmortem analysis rather than real-time analysis because of the complexity of feature aggregation. Statistical features per connection of single flow are equally aggregated for effective identification and secure data collection. Early and Brodley presented a data collection and analysis approach based on SCD features (Early and Brodley, 2006). With the respect of transmission of TCP connections, mean packet inter-arrival time, mean packet length, etc. are calculated as statistical information. Thereafter, these features are transmitted to a data analyzer to be clustered and classified. Thus, malicious traffic can be detected and eliminated from normal data. This approach is used to detect single and marked anomalous data. But as for complex data, it seems hard to be marked. Thus, MCD features are needed. MCD features, also called group features (Hofstede et al., 2014), are aggregated multiple flow features that are used to extract complex statistical flows. By utilizing foregone conclusions and domain knowledge, corresponding flows are classified into anomalous traffic. The mechanism based on SCD and MCD is more accurate to detect attacks. However, it must tolerate time delay that is caused by features aggregation.

The ingress and egress filtering based collection mechanism stretches the five-tuple mechanism. In the situation that researchers or network managers need additional features of the packets, the ingress and egress filtering mechanism is suitable to be applied. However, this mechanism is still a distortion method. How to extract appropriate features is worth further investigating.

4.2.3. *Deep Flow Inspection (DFI) mechanisms*

DFI technology is another type of flow monitoring approach. It identifies flow based on the behavior characteristics of flow (Liu et al., 2014). For example, packet length is an important and effective behavior feature of flow. In a VoIP connection, the packet length of a voice message is usually between 130 and 220 bytes and the active flow often stays for a long time. Thus, a DFI mechanism applies specific features compared to normal flow detection approaches. Afterwards, its latter process analyzes the content of data packages and compares them to attack features stored in a presupposed library. As a result, corresponding hardware or software modules control access rules and discard unexpected packets.

Xia and Song (2009) proposed a mechanism to collect Peer-to-Peer (P2P) traffic data based on DFI. In a P2P network, peer nodes communicate with others by using dynamic ports to prevent their flows from being identified. And their payload contents are encrypted for privacy reasons. After the packets are captured, an aggregator calculates the statistics based on feature database and divides the packets. Then, feature database is revised due to subsequent packets. According to statistical flow behaviors, useless flows can be discarded and valuable data can be collected. In this way, the author identifies most P2P services with high EF, such as Skype, BitComet and MSN. The detection rate can reach 86%. The rest services that cannot be identified are instant communications, etc. This mechanism ensures II, CO and AV of information.

Information entropy is an important concept in these mechanisms to select characteristics. There are lots of algorithms based on this concept. Biologically Inspired Feature (BIF) (Senliol et al., 2008) is an intuitive algorithm to select features. It chooses the characteristics of minimum information entropy from all candidates. Some other characteristic selection algorithms are based on BIF. The Fast Correlation Based Filter (FCBF) (Guo, 2011) is one of them. It mainly uses Markov blanket technology to calculate characteristic similarity, so as to rapidly aggregate valid characteristics and filter redundant characteristics. By testing diverse examples of different features, the mechanism has both good average recognition precision and low time-costs. This implies high EF and IN of this mechanism.

The DFI mechanism is very practical in the situations that original flow mechanisms cannot take effect. But this mechanism ignores the information contained by packet payloads and headers even when they are available, which may make it miss many valuable traffic features. Thus, it is a mechanism that supports restricted data collection.

4.2.4. *OpenFlow based mechanisms*

SDN is an emerging technology proposed to improve the complex structures of traditional networks. The architecture of SDN is comprised of data plane, control plane and application plane. After SDN operators develop a policy in the application plane, the controllers execute the policy in the control plane and SDN switches complete corresponding forwarding actions in the data plane. OpenFlow is the most notable southbound API to connect the data plane and the control plane. Every packet flow matches with flow tables to be dropped or classified into a specific class. Therefore, flow-based data collection mechanisms are easily accomplished in SDN networks. Besides, SDN network is prone to collect management and statistical data generated and aggregated from the control plan. Accordingly, there are abundant data collection mechanisms for collecting statistical data. Due to the convenience made by the application plane, application-aware data collection mechanisms are also applicable in the SDN networks.

Foster et al. (2011) proposed a data collection mechanism using a programming language for OpenFlow-based networks. It provides an interface that defines flow filtering rules for incoming traffic. According to the flow policy sent from flow tables, created policies can be executed in the data plane to react to network events. This mechanism provides flexibility on data collection configuration through programming, though it does not make up exact rules for data collection. In theory, because of the network-wide awareness of SDN, the mechanism can easily match TR and AV. The controller can detect the data flows through each network node. According to tests, this mechanism is more effective than NOX, an OpenFlow controller. It also satisfies SC because it can easily achieve the functions of spanning and can discover some types of DDoS.

An autonomic data collection architecture was presented by Bari et al. (2013) for collecting both network statistics and real-time traffic. In this architecture, collecting policies are adjusted according to the feedback from autonomic analyzers. It overcomes the shortcomings of real-time collection in previous work. However, the payload of controller, which manages and controls the traffic in a centralized way, may become a bottleneck. Through simulation comparison, this mechanism suffers less packet loss and link failure, which implies its great performance regarding II and EF in data collection. Moreover, it has more throughput and available bandwidth usage, so its RO is optimized.

The SDN-based data collection mechanism needs specific SDN routers to implement. However, SDN network is of low network penetration in current environments, which may cause the unavailability of the mechanisms. Besides, the linear flow tables in controllers have inadequate capabilities to create complex and interleaving rules.

4.3. *Log-based data collection*

The log-based data collection is a significant part of network security audit process (Turner et al., 2013). Hosts, mobile devices, routers, IDSs, different types of Web servers, data centers, and every node of network system contain log files. Besides, log files can be partitioned into different types, while each type of them records diverse information for further analyzing and auditing. According to their functions and locations where they are generated, they can be divided into three categories: operating system logs, Web logs and equipment logs (Yan et al., 2012). Generally, logs provide the information of registration, environmental information, system changes, etc. Dynamic information like memory usage and static information such as user name are contained in the logs. When a service is turned on, its log file is therewith created in a corresponding directory. While a routine is running, the log records its date, exact time, operator, involved parties and actions. Obviously, a log file customarily contains abundant information worth analyzing.

LogParser (Yan et al., 2012) is a log screening tool to accomplish an ordinary log collection and analysis task of several log files. It depends on a manual rule set. However, it cannot suit for a large-scale and adaptive analyzing process because of the difficulty to log inspection manually in large-scale circumstances. Meanwhile, individual analysis of logs is perhaps restricted because of the high correlation of the file and the lack of effective information. Accordingly, it is necessary to investigate a comprehensive analysis scheme to integrate multiple log contents without manual operations.

There are two main approaches to collect and analyze data from log files. Rule matching is a simple and effective approach for some log structures. This mechanism parses log features based on expert knowledge and then find similar segments in log files. The other approach is machine learning for deep analytic, which is not limited at the expression of log files. Bayesian methods, Support Vector Machine (SVM) methods customarily fall into this type of approaches.

Yan et al. (2012) held a viewpoint that single system log contents cannot be used for a comprehensive analysis in spite of almost all of shallow information of the system is covered. They presented a mechanism combining multiple log files to identify multiple candidate risk points from host logs. According to corresponding match rules, diverse features of malicious traffic are structured. In the sight of different types of attacks, their log files are generated into three categories: login log, log of uploading files, log generated by processing operations. For instance, if an invader crashes the operation system by creating a false identity, administrators will check login logs that contain user ID, user name, invasion time, etc. Equally, they can narrow the scope to identity attacks when the part of abnormal logs is assured. This mechanism is effective in attack detection. However, this data collection methodology heavily relies on expert knowledge so that complete and precise rules must be formulated. Otherwise, it is prone to obtain false-negative results. To mitigate this dilemma, operators must frequently revise the rules, which seems unrealistic and inefficient for a data collection system.

In a network data collection process, router syslog messages are regarded as the most informative resources. Due to its extensive records, probing or collecting router syslog is considered as one of the most comprehensive mechanisms. However, the format of router syslog information is extremely optional with a simple structure (Qiu et al., 2010). Besides, raw router syslog messages are almost meaningless, which must be aggregated and processed in order to be utilized. There are some approaches by using rule matching and expert systems. However, they are over-reliant on domain knowledge. If experts cannot provide enough knowledge, high false-positive and low accuracy of log analysis result will happen. Moreover, the result could be deviated when experts give incorrect knowledge.

Qiu et al. (2010) presented a mechanism using machine learning with self-adaption capacity. The mechanism consists of an offline domain knowledge learning process and an online syslog digest process. In the offline domain knowledge learning process, its actual state is frequently changed due to the train of historical syslog data and router configuration files. Then, the trained collector is deployed into networks to collect data and identify abnormal status. Meanwhile, the collector has the capability of self-correction with acquired data. Thus, it can obtain desired effectiveness finally. Though the mechanism is of high EF, it requires complex file formats, high capability and large RO of analyzers. These factors restrict the scope of the mechanism's application because some of log collection devices do not have the capability to process complex file formats.

With the development of the Internet and increasing demand of cloud computing, web-based applications have become extremely prevalent nowadays. Hence, web log is regarded as another essential data source for network analysis. There are different kinds of logs, such as web server logs, error logs and application logs. Similar to the logs of network devices and hosts, web log formats are not uniform. Besides, the formats are usually modified and updated due to regular maintenance and periodic updates of applications. It seems really complicated for log analysis.

Asselin et al. (2016) proposed an anomaly detection model for web applications. By applying this model, trivial reports are filtered so that web server logs are reduced. Moh et al. (2016) proposed an approach by combining pattern matching and machine learning. Standardized log files are firstly sent to machine learning components and malicious attacks can be identified. Then, the decision results are used for further analysis at pattern matching components. In this approach, it mitigates the deviation caused by systematic error of machine learning mechanisms. For example, a Bayesian decision method assumes that all features are independent, which is hard to be satisfied customarily. Thus, the approach can achieve better performance than the scheme that only uses pattern matching or machine learning. However, it inherits their weakness at the same time. The approach must unify the formats and update the matching rules in order to keep an excellent performance.

Shi et al. (2012) used a concept called Intelligent Agent (IA). It is an agent entity having self-perception, analysis, and response capabilities. IA can replace users to realize complex work such as corresponding information filtering, querying, management and resource integration. Through utilizing IA entities, the authors implemented log collection agents on diverse devices such as applications, network devices and secure devices. Thereafter, the log collection and analysis system implements centralized storage and control. Thus, it's necessary to unify their log formats coming from different kinds of agents. By this way, available log information can easily be collected by the agents independently. Thus, the approach can get higher EF than normal log mechanisms without impacting network performance. However, how to design an efficient and feasible IA entity is an aporia. An IA entity often has diverse but untargeted characteristics, some of which are not obtained and meaningless for the system. Due to this, it's worth studying the resource consumption and the cost of IA entities in order to make such a mechanism practical.

Unlike packets, SDN logs cannot record overall information of networks. But these logs, such as newly defined information, can be used for network security measurement and traffic monitoring. Actually, a controller-based mechanism is the main approach of data collection in SDN networks, where SDN flow table becomes an alternative of the traditional flow.

The SDN logs record such main functions as flow table modification, network component connection status and topology characteristics (Siniarski et al., 2016; Henni et al., 2016). Log based data collection is also available for SDN architectures. Siniarski et al. (2016) proposed an architecture to monitor SDN in real-time by using non-invasive cloud-based logging platforms. There are many log collection agents distributed in different terminals and network devices. The agents automatically collect logs from controllers, switches and Virtual Machines (VMs). Then, the collected data are parsed into logical fields. Next, processed logs are forwarded to a remote log manager for centralized filtering and analysis. Finally, corresponding feedback such as potentially critical events probably affecting network performance, potential attacks and malicious behaviors, network health status are reported. Comparing to the logs in routine networks, SDN logs have relatively fine formats, especially in control plane due to the usage of simple and fine-organized protocols and flow formats. By using non-invasive mechanisms to collect log records, network load and performance are not affected. And union features extracted by multiple nodes in a remote analyzing platform can provide high accuracy to identify attacks and malicious traffic. However, due to rough filtering and lack of concerns of adaption, this kind of data collection mechanisms perhaps induces a large volume data in the process of transmission and storage. This could cause a bottleneck that limits availability and executive efficiency of the mechanism.

5. Open issues and future research directions

5.1. Open issues

Based on our survey, we identify some issues that still remain open for the purpose of direct future research in the area of network data collection.

First, effective and accurate data collection is missed for reducing the size of data collection. With the coming of big data era, large amount of data becomes the first thing to consider. In most of existing mechanisms, the network data flow is all collected by collectors. But, not all of them are needed for corresponding data process or analysis. Obviously, it's too heavy and no need to collect and manage useless data due to resource consumption and memory occupation at collectors, especially for Internet of things (IoT) (e.g. wireless sensors) with constrained capabilities. Some schemes we researched considered traffic forecasting and data sampling. But the literature still lacks a sampling scheme with high effectiveness and accuracy, as well as adaptability. Thus, the issues of data amount are still open for future researchers.

Second, the approaches to deal with encrypted data for ensuring data privacy is still an open issue. Due to the awareness of user privacy and the improvement of network systems, individual privacy has gotten more and more concerns. Nowadays, encrypted data become a maintain format to transmit and store. However, in most of current collection mechanism, it is usually given an assumption that collected data is plaintext so that operators can easily deal with it, though the assumption is not suitable nowadays. How to collect the encrypted data and support later data analysis and process while at the same time preserve data privacy become a practical challenge.

Third, proper access control is not well studied in the process of data collection, which is requested due to privacy leakage caused by collectors. The privacy protection of network data collection is an important and significant issue that should be considered (Zhang et al., 2017). However, cooperation among multiple operators for data collection in a heterogeneous network increases the risk of privacy leakage. Some information providers are pleasant to offer information to reliable collectors. However, they dislike their privacy leaked to public. Thus, essential access control should be studied in an inter-domain scenario with regard to network data collection.

Fourth, existing work lacks study on data authenticity verification. It is required to ensure that the collected data really reflect a networking environment. Data collectors can be deployed in remote unsupervised places, where various attackers such as session hijack and internal node compromise can easily change network routes or data traffic. Some meaningless or malicious data could be forged and be transmitted to collectors or servers. Accordingly, when the data are collected, it requires that the core components of data collection systems have the capability to authneticate the source of data for the purpose of identify compromised collectors.

Last but not the least, the security of data collection architecture has not been comprehensive investigated in prior arts. In reality, the performance of a data collection mechanism is heavily dependent on system security. The data collected by a comprised system is invalid for network security measurement and further analysis. Some researches have put forward varied attacks. However, not all of them are well solved. For example, in packet sniffer-based approaches in hosts, there is still no valid solutions to identify eligible managers from malicious intruders. Besides, a synthetic and comprehensive solution that can satisfy all evaluation criteria is still not available yet.

5.2. Future research directions

Based on the above specified open issues, we propose a number of future research directions that we think, are worth special efforts.

First, distributed data collection mechanisms are worth researching for feature investigators. The new mechanisms should have good performance in data collection in a large scale heterogeneous network. Although there are many architectures proposed by researchers, only some small prototypes and small-scale enterprise architectures are deployed in a real network environment. The security management and large-scale control across different networking domains are still open challenges for current network operators.

Second, the architecture based on SDN seems to be a good choice for comprehensive and optimized network data collection. SDN architecture has the awareness of complete network topology and can control a network system in a centralized way. Besides, the research of SDN is becoming mature, which can provide great convenience for prototyping and testing the performance of data collection and data analytics for various network management and security purposes. However, the compatibility between SDN and original networks is waiting to be solved.

Third, adaptive and extensible data collection is a significant research topic that should be investigated. Prior arts of data collection mostly collect all the data that pass through collecting nodes. However, this kind of mechanisms are forcible and ineffective. A flexible and intelligent mechanism should be proposed. It can intelligently collect data according to current needs, past experiences, data features, maybe based on traffic forecasting or smart and adaptive data sampling. Meanwhile, due to the heterogeneity of networks, adaptability and scalability become two important properties that should be supported by the next generation data collection mechanisms.

6. Conclusions

Data collection is the foundation of many network management services, such as intrusion detection, network situational awareness, and even network security measurement. The studies on it is particularly significant for solving existing security evaluation problems, facing the era of big data in cyber space. In this paper, we performed a thorough survey on the existing mechanisms of network data collection. We introduced the background of this study, proposed a number of criteria for evaluating the performance of network data collection, and reviewed existing work based on data carriers, i.e., packet-based data collection, flow-based data collection and log-based data collection. For each of reviewed data collection mechanism, we described its approach and commented its performance according to the proposed evaluation criteria. Finally, we indicate open issues and forecast future research directions, targeting at large scale heterogeneous networks.

Acknowledgment

This work is sponsored by the National Key Research and Development Program of China (grant 2016YFB0800704), the NSFC (grants 61672410 and U1536202), the Project Supported by Natural Science Basic Research Plan in Shaanxi Province of China (Program No. 2016ZDJC-06), the 111 project (grants B08038 and B16037), and Academy of Finland (grant No. 308087).

References

An, X., Liu, X., 2016. Packet capture and protocol analysis based on Winpcap. In: 2016 International Conference on Robots & Intelligent System (ICRIS), pp. 272–275.
Antichi, G., Giordano, S., Miller, D.J., Moore, A.W., 2012. Enabling open-source high speed network monitoring on NetFPGA. In: 2012 IEEE Network Operations and Management Symposium, pp. 1029–1035.
Asselin, E., Aguilar-Melchor, C., Jakllari, G., 2016. Anomaly detection for web server log reduction: a simple yet efficient crawling based approach. In: 2016 IEEE Conference on Communications and Network Security (CNS), pp. 586–590.
Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J., 2002. Models and issues in data stream systems. In: Presented at the Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 1–16.
Bajpai, V., Schönwälder, J., 2017. Network flow query language–design, implementation, performance, and applications. IEEE Trans. Netw. Serv. Manag. 14, 8–21.
Bari, M.F., Chowdhury, S.R., Ahmed, R., Boutaba, R., 2013. PolicyCop: an autonomic QoS policy enforcement framework for software defined networks. In: 2013 IEEE SDN for Future Networks and Services (SDN4FNS), pp. 1–7.
Bian, S.S., Zhang, P., Yan, Z., 2016. A survey on software-defined networking security. In: Presented at the Proceedings of the 9th EAI International Conference on Mobile Multimedia Communications, pp. 190–198.
Bo, Y., Yi, L., Yuehui, C., Runzhang, Y., 2005. A flow-based network monitoring system used for CSCW in design. In: Proceedings of the Ninth International Conference on Computer Supported Cooperative Work in Design, 2005, pp. 503–507.
Callado, A., Kamienski, C., Szabo, G., Gero, B.P., Kelner, J., Fernandes, S., et al., 2009. A survey on Internet traffic identification. IEEE Commun. Surv. Tutorials 11, 37–52.
Carli, L.D., Pan, Y., Kumar, A., Estan, C., Sankaralingam, K., 2009. PLUG: flexible lookup modules for rapid deployment of new protocols in high-speed routers. In: Presented at the Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication, Barcelona, Spain.
Das, R., Tuna, G., 2017. Packet tracing and analysis of network cameras with Wireshark. In: 2017 5th International Symposium on Digital Forensic and Security (ISDFS), pp. 1–6.
Davis, J.J., Clark, A.J., 2011. Data preprocessing for anomaly based network intrusion detection: a review. Comput. Secur. 30, 353–375.
Early, J.P., Brodley, C.E., 2006. Behavioral features for network anomaly detection. In: Machine Learning and Data Mining for Computer Security: Methods and Applications, pp. 107–124.
Elsen, L., Kohn, F., Decker, C., Wattenhofer, R., 2015. goProbe: a scalable distributed network monitoring solution. In: 2015 IEEE International Conference on Peer-to-peer Computing (P2P), pp. 1–10.
Ficara, D., Giordano, S., Oppedisano, F., Procissi, G., Vitucci, F., 2008. A cooperative PC/Network-Processor architecture for multi gigabit traffic analysis. In: 2008 4th International Telecommunication Networking Workshop on QoS in Multiservice IP Networks, pp. 123–128.
Foster, N., Harrison, R., Freedman, M.J., Monsanto, C., Rexford, J., 2011. "Frenetic: a network programming language. ACM Sigplan Not. 46, 279–291.
Garg, A., Maheshwari, P., 2016. PHAD: packet header anomaly detection. In: 2016 10th International Conference on Intelligent Systems and Control (ISCO), pp. 1–5.
Goncalves, P., Oliveira, J.L., Aguiar, R.L., 2009. An evaluation of network management protocols. In: 2009 IFIP/IEEE International Symposium on Integrated Network Management, pp. 537–544.
Guo, L., 2011. Smile expression classification using the improved BIF feature. In: 2011 Sixth International Conference on Image and Graphics, pp. 783–788.
He, L.M., Yan, Z., Atiquzzaman, M., 2018. LTE/LTE-A network security data collection and analysis for security measurement: a survey. IEEE Access 6 (1), 4220–4242.
Henni, D.E., Hadjaj-Aoul, Y., Ghomari, A., 2016. Probe-SDN: a smart monitoring framework for SDN-based networks. In: 2016 Global Information Infrastructure and Networking Symposium (GIIS), pp. 1–6.
Hillbrecht, R., Bona, L.C.E. d, 2012. A SNMP-based virtual machines management interface. In: 2012 IEEE Fifth International Conference on Utility and Cloud Computing, pp. 279–286.
Hofstede, R., Bartoš, V., Sperotto, A., Pras, A., 2013. Towards real-time intrusion detection for NetFlow and IPFIX. In: Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013), pp. 227–234.
Hofstede, R., Čeleda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A., et al., 2014. Flow monitoring explained: from packet capture to data analysis with NetFlow and IPFIX. IEEE Commun. Surv. Tutorials 16, 2037–2064.
Huang, S., Griffioen, J., Calvert, K.L., 2014. Network hypervisors: enhancing SDN infrastructure. Comput. Commun. 46, 87–96.
Hutter, M., Szekely, A., Wolkerstorfer, J., 2009. Embedded system management using WBEM. In: 2009 IFIP/IEEE International Symposium on Integrated Network Management, pp. 390–397.
Ji, Z., Kuang, Z., Ni, H., 2009. A novel two-dimension adaptive data collection method for network management. In: 2009 WRI International Conference on Communications and Mobile Computing, pp. 237–241.
Kamiyama, N., Mori, T., 2006. Simple and accurate identification of high-rate flows by packet sampling. In: Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications, pp. 1–13.
Kim, S.S., Reddy, A.L.N., 2008. Statistical techniques for detecting traffic anomalies through packet header data. IEEE/ACM Trans. Netw. 16, 562–575.
Kreutz, D., Ramos, F.M.V., Veríssimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S., 2015. "Software-Defined networking: a comprehensive survey. Proc. IEEE 103, 14–76.
Kundu, S.R., Pal, S., Basu, K., Das, S.K., 2009. An architectural framework for accurate characterization of network traffic. IEEE Trans. Parallel Distr. Syst. 20, 111–123.
Lee, S., Levanti, K., Kim, H.S., 2014. Network monitoring: present and future. Comput. Network. 65, 84–98.
Li, W., Liu, H., Zhang, X., 2016. A network data security analysis method based on DPI technology. In: 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), pp. 973–976.
Lin, H.Q., Yan, Z., Chen, Y., Zhang, L.F., 2018. A survey on network security-related data

collection technologies. IEEE Access. http://dx.doi.org/10.1109/ACCESS.2018.2817921.

Liu, A.X., Meiners, C.R., Norige, E., Torng, E., 2014. High-speed application protocol parsing and extraction for deep flow inspection. IEEE J. Sel. Area. Commun. 32, 1864–1880.

Liu, G., Yan, Z., Pedryczc, W., March 2018. Data collection for attack detection and security measurement in mobile ad hoc networks: a survey. J. Netw. Comput. Appl. 105, 105–122.

Mann, V., Vishnoi, A., Bidkar, S., 2013. Living on the edge: monitoring network flows at the edge in cloud data centers. In: 2013 Fifth International Conference on Communication Systems and Networks (COMSNETS), pp. 1–9.

Meiners, C., Norige, E., Liu, A.X., Torng, E., 2012. FlowSifter: a counting automata approach to layer 7 field extraction for deep flow inspection. In: 2012 Proceedings IEEE INFOCOM, pp. 1746–1754.

Michaut, F., Lepage, F., 2005. Application-oriented network metrology: metrics and active measurement tools. IEEE Commun. Surv. Tutorials 7, 2–24.

Moh, M., Pininti, S., Doddapaneni, S., Moh, T.S., 2016. Detecting web attacks using multi-stage log analysis. In: 2016 IEEE 6th International Conference on Advanced Computing (IACC), pp. 733–738.

Moindze, S.M., Konate, K., 2014. A survey of the distributed network management models and architectures: assessment and challenges. In: 2014 IEEE 6th International Conference on Adaptive Science and Technology (ICAST), pp. 1–8.

Morariu, C., Stiller, B., 2008. DiCAP: distributed Packet Capturing architecture for high-speed network links. In: 2008 33rd IEEE Conference on Local Computer Networks (LCN), pp. 168–175.

Oliner, A., Ganapathi, A., Xu, W., 2011. Advances and challenges in log analysis. Queue 9, 30–40.

Papadogiannakis, A., Antoniades, D., Polychronakis, M., Markatos, E.P., 2007. Improving the performance of passive network monitoring applications using locality buffering. In: 2007 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, pp. 151–157.

Qadeer, M.A., Iqbal, A., Zahid, M., Siddiqui, M.R., 2010. Network traffic analysis and intrusion detection using packet sniffer. In: 2010 Second International Conference on Communication Software and Networks, pp. 313–317.

Qiu, T., Ge, Z., Pei, D., Wang, J., Xu, J., 2010. What happened in my network: mining network events from router syslogs. In: Presented at the Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, pp. 472–484.

Roesch, M., 1999. Snort - Lightweight Intrusion Detection for Networks. Lisa. pp. 229–238.

Sekar, V., Reiter, M.K., Willinger, W., Zhang, H., Kompella, R.R., Andersen, D.G., 2008. CSAMP: a system for network-wide flow monitoring. In: Presented at the Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, pp. 233–246.

Sekar, V., Reiter, M.K., Zhang, H., 2010. Revisiting the case for a minimalist approach for network flow monitoring. In: Presented at the Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, pp. 328–341.

Senliol, B., Gulgezen, G., Yu, L., Cataltepe, Z., 2008. Fast correlation based filter (FCBF) with a different search strategy. In: 2008 23rd International Symposium on Computer and Information Sciences, pp. 1–4.

Shi, S., Shen, X., Zhao, J., Ma, X., 2012. Research on system logs collection and analysis model of the network and information security system by using multi-agent technology. In: 2012 Fourth International Conference on Multimedia Information Networking and Security, pp. 23–26.

Siniarski, B., Olariu, C., Perry, P., Parsons, T., Murphy, J., 2016. Real-time monitoring of SDN networks using non-invasive cloud-based logging platforms. In: 2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), pp. 1–6.

Sperotto, A., Schaffrath, G., Sadre, R., Morariu, C., Pras, A., Stiller, B., 2010. An overview of IP flow-based intrusion detection. IEEE Commun. Surv. Tutorials 12, 343–356.

Sundaram, S., Jong-Cheol, S., Abdurakhmanov, A., Young-Tak, K., 2006. Design and implementation of WBEM-based network management system for inter-as traffic engineering. In: Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06), pp. 162–170.

Thangapandiyan, M., Anand, P.M.R., 2016. An efficient botnet detection system for P2P botnet. In: 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), pp. 1217–1221.

Therdphapiyanak, J., Piromsopa, K., 2013. An analysis of suitable parameters for efficiently applying K-means clustering to large TCPdump data set using Hadoop framework. In: 2013 10th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, pp. 1–6.

Turner, D., Levchenko, K., Savage, S., Snoeren, A.C., 2013. A comparison of syslog and IS-IS for network failure analysis. In: Presented at the Proceedings of the 2013 Conference on Internet Measurement Conference, pp. 433–440.

Xia, T., Song, R., 2009. A method of P2P traffic identification on Internet based on the deep flow inspection. In: 2009 International Conference on Communication Software and Networks, pp. 777–780.

Xu, C., Chen, S., Su, J., Yiu, S.M., Hui, L.C.K., 2016. A survey on regular expression matching for deep packet inspection: applications, algorithms, and hardware platforms. IEEE Commun. Surv. Tutorials 18, 2991–3029.

Yan, D., Feng, R., Huang, J., Yang, F., 2012. Host scurity event track for complex network environments based on the analysis of log. In: 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems, pp. 807–811.

Yan, S., Aguado, A., Ou, Y., Wang, R., Nejabati, R., Simeonidou, D., 2017. Multilayer network analytics with SDN-based monitoring framework. IEEE/OSA J. Opt. Commun. Netw. 9, A271–A279.

Yao, Z., Yan, Z., 2016. Security in software-defined-networking: a survey. In: The 9th International Conference Security, Privacy, and Anonymity in Computation, Communication, and Storage, pp. 319–332.

Yao, Z., Yan, Z., 2018. A trust management framework for software-defined network applications. Concurrency Comput. Pract. Ex. e4518. http://dx.doi.org/10.1002/cpe.4518 (IF: 1.133).

Yu, J., Lee, H., Kim, M.-S., Park, D., 2008. Traffic flooding attack detection with SNMP MIB using SVM. Comput. Commun. 31, 4212–4219.

Zeng, W., Wang, Y., 2009. Design and implementation of server monitoring system based on SNMP. In: 2009 International Joint Conference on Artificial Intelligence, pp. 680–682.

Zhang, H., Yang, Z., Guo, W., 2009. Flow byte sizes estimation from simple random packet sampling. In: 2009 Fifth International Joint Conference on INC, IMS and IDC, pp. 92–96.

Zhang, P., Li, H., Hu, C., Hu, L., Xiong, L., 2016. Stick to the script: monitoring the policy compliance of SDN data plane. In: 2016 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), pp. 81–86.

Zhang, L.F., Yan, Z., Kantola, R., July 2017. Privacy-preserving trust management for unwanted traffic control. Future Generat. Comput. Syst. 72, 305–318.

Zhao, K., Zhang, M., Yang, K., Hu, L., 2007. Data collection for intrusion detection system based on stratified random sampling. In: 2007 IEEE International Conference on Networking, Sensing and Control, pp. 852–855.

Donghao Zhou received the B·Sc. degree in telecommunications engineering from Zhengzhou University, Zhengzhou, China, in 2016. He is currently pursuing the master degree in information security in Xidian University, Xi'an, China. His research interests are in security, Software-Defined-Network (SDN) and network security measurement.

Zheng Yan (M′06, SM′14) received the BEng degree in electrical engineering and the MEng degree in computer science and engineering from the Xi'an Jiaotong University, Xi'an, China in 1994 and 1997, respectively, the second MEng degree in information security from the National University of Singapore, Singapore in 2000, and the Licentiate of Science and the Doctor of Science in Technology in electrical engineering from Helsinki University of Technology, Helsinki, Finland in 2005 and 2007. She is currently a professor at the Xidian University, Xi'an, China and a visiting professor and Finnish Academy Research Fellow at the Aalto University, Espoo, Finland. Her research interests are in trust, security privacy, and data mining. Prof. Yan serves as an associate editor of Information Sciences, IEEE Internet of Things Journal, IEEE Access Journal, Information Fusion, JNCA, Security and Communication Networks, etc. She is a leading guest editor of many reputable journals including ACM TOMM, FGCS, IEEE Systems Journal, MONET, etc. She served as a steering, organization and program committee member for over 80 international conferences. She is a senior member of the IEEE.

Yulong Fu received his B·Sc. from Harbin Institute of Technology, M.A. Sc. degree from University of Bordeaux 1 (France), and Ph.D. degree from University of Pau (France) in 2014. He is currently a lecturer in Xidian University, China. His research interests include 5G Security, Formal Methods, Data Driven Security and Security Measurement.

Zhen Yao graduated from XiDian universitey in 2013. Now, he is pursuing master degree in the State Key Laboratory on Integrated Services Networks, Xidian University, China. His main research direction is Software-Defined-Network trust management system.