

TPEFD: an SDN-based efficient elephant flow detection method

TIAN Yu¹, LIU Jing^{1,2,3}, LAI Ying-xu^{1,2,3}, BAO Zhen-shan¹, ZHANG Wen-bo¹

(1. Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China;

2. Key Laboratory of Trusted Computing, Beijing 100124, China;

3. Information Security Rank Protection Key Technology National Engineering Laboratory, Beijing 100124, China)

Abstract: Software-defined networking (SDN) is a new approach to configure and operate programmable switches of the networks (especially the data center networks) through a centralized software controller. Elephant flows normally exist in data center networks and take up a large amount of network bandwidth, so the elephant flow detection is very important to ease network congestion. A two-phase real-time detection (TPEFD) method was proposed to detect the elephant flows in the SDN-based network. First, the controller obtained aggregated statistics and shrank sample scope until it was small enough, packets were sampled in the scope in switches. In order to identify the elephant flows, the sFlow sampling results were compared with the dynamic threshold. If the sampling value exceeded the threshold value, the flow was recognized as an elephant flow. The efficiency of our method in an SDN experimental environment was evaluated. The experimental results indicated that the proposed method was feasible and the detection time was efficient.

Key words: OpenFlow, elephant flows, flow detection, data center

doi: 10.11959/j.issn.2096-109x.2017.00167

1 Introduction

SDN technology based on OpenFlow^[1] protocol, an open standard designed for SDN can separate the control plane and data plane. We can use SDN for traffic engineering and QoS management. SDN enables network programmability and provides a new solution for the development and management of the network.

The network traffic is generally quantified by flows which collect packets with the same features. Network flows are generally identified by five tuples in the packet header. The five tuples contain source IP, destination IP, source port number, destination port number and IP protocol. The flow collection is usually defined as an aggregation flow. Elephant flows carry most of the network traffic in a period of time, so it may have a long delay com-

pared with a small flow. In data center networks^[2-4], although elephant flows only account for 2% of the number of flows, it holds 90% of the network traffic. Data center traffic has the characteristics of heavy-tailed distribution^[5,6], the number of mice flows is large while only account for little traffic. This phenomenon is very important to improve the network performance. The competition of resources between elephant flows and mice flows will make it very difficult for mice flows to obtain sufficient bandwidth. It is not necessary for controller to operate all flows. The controller only needs to pay attention to the elephant flows that impact network performance when network traffic managing, thus elephant flow detection is very important to ease network congestion.

ESHSP^[7] classifies flow table entry according to two dimensions {source IP, destination IP}, the

Received date: 2017-01-23, **Revised date:** 2017-02-15. **Corresponding author:** TIAN Yu, tianyu91@emails.bjut.edu.cn

Foundation Items: The Natural Science Foundation of Beijing (No.4162006), The Natural Science Foundation of Qinghai Province (No.2017-ZJ-912)

controller obtains aggregated statistics and shrinks the IP range until it is small enough, then obtains individual statistics. The method ignores the source port number, the destination port number and the IP. According to the randomness of identification field in IP packet, the work^[8] uses certain mask to realize distributed packet sampling.

In this paper, we developed an SDN-based efficient elephant flow detection method for data center networks. There is a conflict between the accuracy and the instantaneity of traffic detection. In a data center network, the arrival of flows are very quickly, the elephant flow identification must be high precision. In flow analysis, the failure to detect elephant flows have more serious consequences than mice flow. In our experiment, we chose Ryu^[9] as the controller. This paper mainly involves the following points.

1) We put forward a new two-phase elephant flow detection method that can reduce the burden of the control plane and rapidly identify elephant flows. At the first phase, we obtain source IP range includes the elephant flows. It only focuses on aggregate statistics while the individual statistics are no longer needed. The controller queries statistics of the flows from the switch supported OpenFlow protocol. We need analyse the relationship between statistical values and flow features in data center.

2) We calculate self-adaptive threshold in the second phase. We count threshold regarding not only accuracy, but also fast, thus we choose sFlow^[10,11] to sample packets. The sampling technique based on sFlow need to send all the flow to analyzer, which then determines the existence of the elephant flows, while we sample packets based on the filtered data packets in first stage. If the sampling counts in a flow exceed the threshold value that we calculate, we identify the flow as an elephant flow.

3) We validate the usability on the trace of data center traffic with Mininet^[12], which uses linux network namespace and constructs virtual networks of

hosts and switches. Performance assessment indicate that our method can quickly detect the elephant flows.

2 Related work

There are many ways to detect the elephant flows in previous work. These can be roughly categorized into two major types, based on the network detection and host-based detection.

1) Based on the network detection

Although Estan's^[13] method concerns the elephant flows and ignores the small flows, it needs to handle each packet and increases the operating burden. A sFlow agent installs in the switch to collect traffic information, it monitors each flow that occupies a large link capacity. DevoFlow^[14] classifies elephant flow and handles most small flows in the data plane, it compares the number of sampling packets with the fixed threshold. It will lead to a large number of false positives and negative mistakes, due to ignore the network characteristics. Multistage filter^[15] takes up less space but can lead to overestimating the actual traffic of the elephant flow and has the characteristics of the hardware implementation complexity. It's difficult to be applied in real time measurement. Hedera^[16] utilizes OpenFlow protocol to distinguish elephant flows through continuous polling, the controller needs to collect all the aggregate statistics and the individual statistics. The statistical information between the controller and the switches increases the burden of the network. The data center networks flow transmission rate is usually very fast, so it is likely to lose a lot of packets when polling.

2) Host-based detection

Mahout^[17] proposed a method, identifying elephant flows in the terminal hosts through a shim layer to reduce network bandwidth consumption, but it needs to change each host operating system in data center, however the end hosts can't always be modified. Benson^[18] uses a specific server to collect data and predicts a busy link, it needs to modify the

end hosts slightly.

Our work need to sample packets continually. In our method, in the switch kernel space, we filter packets according to one dimension {source IP}. In our approach, we will analyze 5-tuple of elephant flows through sFlow sampling and provide adjustable sample threshold.

3 Method and implementation

In this section, we mainly propose the elephant flow detection structure and explain the threshold calculation formula. The following is a detailed description about the TPEFD method.

3.1 Elephant flow detection structure

Our two-phase elephant flow detection structure is shown in Figure1. In order to reduce the control bandwidth and processing overhead, our TPEFD method mainly includes two phases: one is to obtain the source IP address range that may exist the elephant flows, the other is to calculate dynamic threshold for the elephant flow identification. If the threshold using static configuration, it will ignore the variation characteristics of the network.

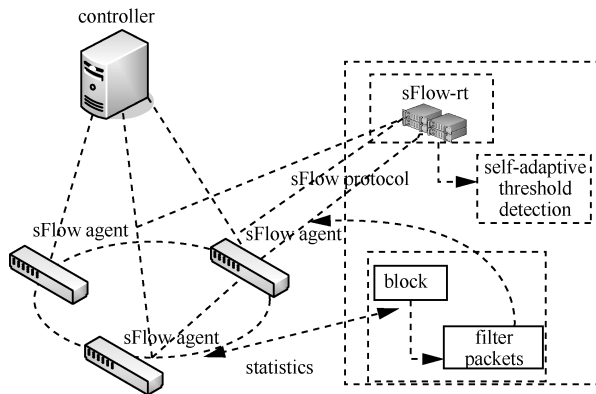


Figure1 Two-phase elephant flow detection structure

Firstly, the SDN controller uses RESTful API^[19] to continually collect aggregate statistics from switches, we use the value for the following analysis. Table 1 and Table 2 show the URI of obtaining the aggregate statistics of switches and the response message body. The response message include a packet count, byte count, and flow count.

Table 1 The URI of aggregating statistics request

Method	Get
URL	/stats/aggregateflow/<dpid>

Table 2 Aggregating statistics response body

Attribute	Description
dpid	datapath ID
packet_count	number of packets in flows
byte_count	number of bytes in flows
flow_count	number of flows

Secondly, we only sample packets within the source IP range obtained from the first phase. The openvswitch communication module is illustrated in Figure 2. The vswitchd component is a daemon, it exists in switch user space and realizes OpenFlow switch core functionality, the datapath module is in the switch kernel module. They coordinate the work to process and forward packets in switch. Meanwhile, the sFlow-agent continuously samples packets. When sampling, we need to modify datapath kernel module and use SFLOW_UPCALL protocol to communicate the kernel module with vswitchd. This does not affect the normal packets forwarding in switches, but only change the number of sample package. First of all, we filter packets according to source IP address, packets with different source IP address are blocked, then compare switch aggregate statistics with θ , which is defined as 20% of the link

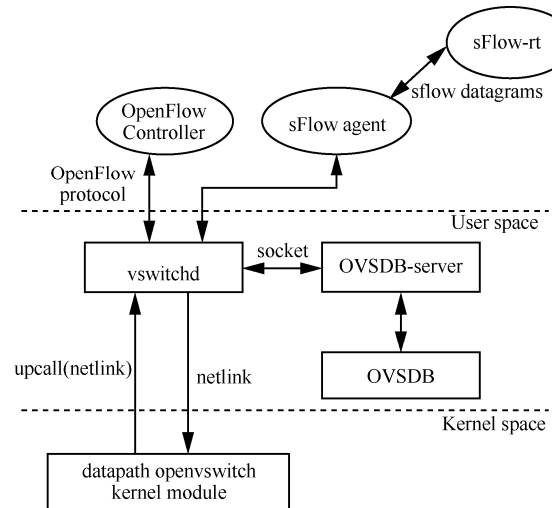


Figure 2 Openvswitch communication module

capacity. If it exceeds θ , there may contain elephant flows. Then we divide the source IP address range into 2^n ($n = 2, 4, 6, \dots$) blocks. We continue to divide blocks if statistics value is larger than θ . We set n according to IP address quantity, if exists 255 source IP address, then we set n equals to 8.

In the solution, the first stage obviously reduce the network traffic because sFlow-agent only collects a small number of packets to sFlow-rt for analysis.

3.2 Identifying elephant flows mechanism

In the first stage of identifying elephant flows, we need constantly to narrow the source IP address range with the purpose of finding the block that may exist elephant flows. The process that acquires the source IP range is as follows.

- 1) **input:** packets in flows
- 2) **output:** elephant flows
- 3) read packet arrives at the OpenFlow switch kernel module
- 4) restful API sends aggregate statistics request
- 5) while (statistics reply $> \theta$) then
- 6) shrink the source IP range & filter packets
- 7) end while
- 8) perform sample packet forwarding based on SFLOW_UPCALL protocol
- 9) sFlow-agent sample packets to collector
- 10) analyse packets & elephant flow detection formula

We use InMon's sFlow-agent in each switch to collect traffic statistics, only a small part of packets are sampled at the switches. A sFlow collector can monitor multiple agent. SFlow-rt collects sFlow datagrams from the agent for analysis after it has sufficient number of samples from the flow. In the process, packets are handled not in switch but on the server side. Sampling identifying elephant flows is expected 1 out of k packets. The sampling frequency should be set reasonable when we sample, the statistical interval is too long that it will lead to lose elephant flows, while short time will increase the

burden of memory. We calculate the self-adaptive threshold value σ in real time. In data center, network traffic follows the characteristics of heavy-tailed distribution, which shows most elements in the collection appear less frequently, but a small number of elements is very high. Common heavy-tailed model is Weibull model, Pareto model, their cumulative distribution curve is shown in Figure 3.

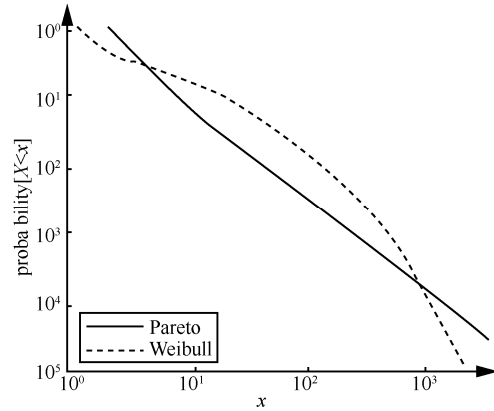


Figure 3 Heavy-tailed distribution function of cumulative distribution curve

In the following experiments, we use the Pareto distribution to calculate dynamic threshold value for convenience. The probability density function is $p(\beta^i = x) = \alpha k^\alpha x^{-\alpha-1} (x \geq k)$, which β^i be the number of packets belong to flow i and cumulative distribution function is

$$f(x) = p[\beta^i \leq x] = 1 - \left(\frac{k}{x}\right)^\alpha \quad (1)$$

The complementary cumulative distributions can be written as follows

$$\bar{f}(x) = 1 - f(x) = p[\beta^i > x] = \left(\frac{k}{x}\right)^\alpha \quad (2)$$

Then the logarithmic probability distribution function is

$$\log_b \bar{f}(x) = \alpha \log_b k - \alpha \log_b x \quad (3)$$

Pareto distribution function under the condition of the log-log is a straight line, the slope of the straight line is $-\alpha$, so the function with the increase of x exponential attenuation. From above equation (2) and equation (3), we estimate α and k using the least

squares method and let $\mu_i = \log_b x_i, v_i = \log_b \bar{f}(x_i)$, the computation formula can be shown as follows

$$\alpha = -\frac{\sum(\mu_i - \bar{\mu})(v_i - \bar{v})}{\sum(\mu_i - \bar{\mu})^2}, k = b^{\frac{\bar{v} - \alpha \bar{\mu}}{\alpha}} \quad (4)$$

If the sampling packet counts is greater than σ , we speculate the overall packet counts exceeds a certain value, thus the flow is an elephant flow. According to the power law distribution characteristics of data center networks. Let n be the total number of sampling packets and k^i be the number of randomly sample packets belong to flow i , which has m^i packets in the population. We suppose to have N packets in total during a period of time, thus we can define the sampling frequency $r_s = \frac{n}{N}$. We use the binomial distribution to calculate and the probability can be shown as follows

$$p_r(k^i|m^i) = \binom{n}{k^i} \rho^{k^i} (1-\rho)^{n-k^i} \quad (5)$$

In equation (5), let $\rho = r_s$. Only when it has enough packets from the same flow, the flow is divided into an elephant flow. However, the process of packet sampling may lead to the false positive rate (FPR) and false negative rate (FNR). FPR represents the error probability of identifying mice flows as elephant flows and FNR refers to the probability that elephant flows are not detected. Our method needs to minimize the error probability on the basis of the detection threshold rule. In order to find the optimum threshold σ , we should balance FNR and FPR. We use equation (6) to calculate FNR, FPR is calculated by equation (7). Here, TPR represents the rate of detected flows are elephant flows and TNR indicates the probability of non-elephant flows have not been mistakenly identified. We combine equation (6) and equation (7) together to calculate the intersection point of FPR and FNR. In the second step, these elephant flow will be confirmed if the sample packets exceeds a certain value. We calculate FPR and FNR by using the method mentioned in Mori^[20], the computation formula is shown as

follows

$$FNR(i) = 1 - TPR(i) = 1 - P_r[Y_i \geq k^i | X_i \geq m] \quad (6)$$

$$\begin{aligned} FPR(i) &= 1 - TNR(i) = 1 - P_r[X_i \geq m | Y_i \geq k^i] \\ &= 1 - \frac{\sum_{\varepsilon=m} P_r[Y_i \geq k^i | X_i = \varepsilon] P_r[X_i = \varepsilon]}{P_r[Y_i \geq k^i]} \end{aligned} \quad (7)$$

4 Results analysis and validation

We test the method in data center network. The implementation of TPEFD is built on Ryu, an open-source OpenFlow controller written by Python language. We use seven switches that support OpenFlow spec. v1.3. S2 and S4 are physical switches, which are CS6550 switches. The other switches are virtual switches, which are openvswitch. These switches are connected with controller respectively. Our controller leverages the Ryu platform to learn topology. We use 255 end hosts with different source IP in Mininet simulator. We install sFlow agents in switches and install sFlow-rt traffic analysis tool in another host. Figure 4 presents the experimental environment.

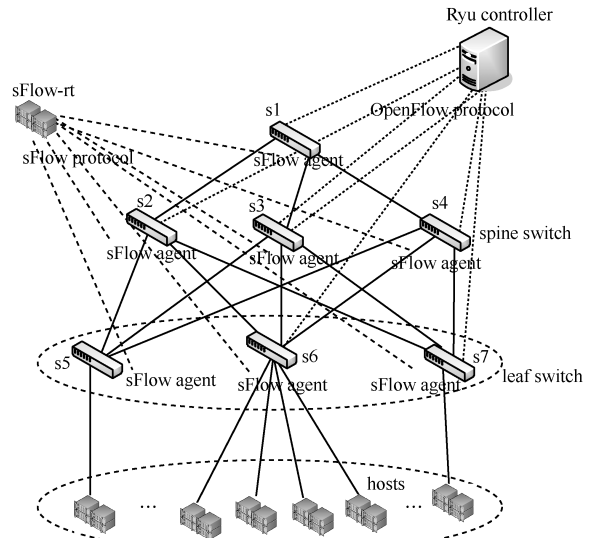


Figure 4 Experimental environment

In our method, we set sampling frequency is $r_s = \frac{1}{1000}$. In practice, misstatement have more serious problems than omission, thus in the circumstance of controlling omission probability, as far as

possible to minimize the misstatement probability. The probability of FPR is less than τ ($\tau=0.05$). Figure 5 is the description of the variation trend of FNR and FPR along with increasing of flow packet counts, here the optimal threshold is σ . If the number of sampling packets exceeds σ , we regard it as an elephant flow. Through our experiments, we find that the more the number of sampled packets a flow have, the more it is likely to be an elephant flow. In the below section, we will analyze and validate the performance of our method.

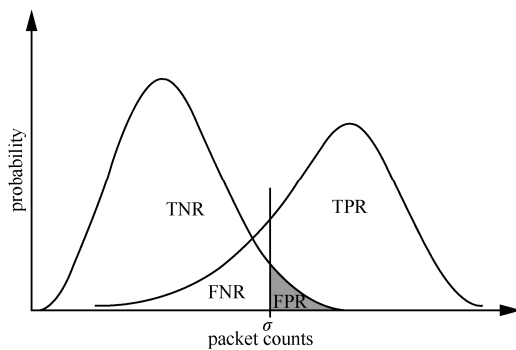


Figure 5 The description of the optimal threshold

To compare our methods with the other approaches, we built experiments on the data center data set. Figure 6 shows the delay time of three competing methods. In our system, the detection delay includes packet filter delay in the first stage and sFlow sample delay in the second stage.

We compare our method with Mahout^[17], ESHSP about the average detection time (ms) and do the experiment many times for different flow numbers and record the average detection time. In our method, the elephant flows can be detected with low overhead in the network. Figure 6 illustrates average elephant flow detection time. Mahout only takes a few milliseconds for detection, because it detects elephant flows in the terminal host. ESHSP needs to preserve the suspicious elephant flows constantly. In our method, by narrowing the scope of elephant flow to filtering sample packets in switch kernel space, which can reduce traffic, the controller doesn't need to process each sampled

packet. Our detection time only needs few milliseconds. Elephant flows usually survive for few seconds, so our method is efficient for elephant flow detection.

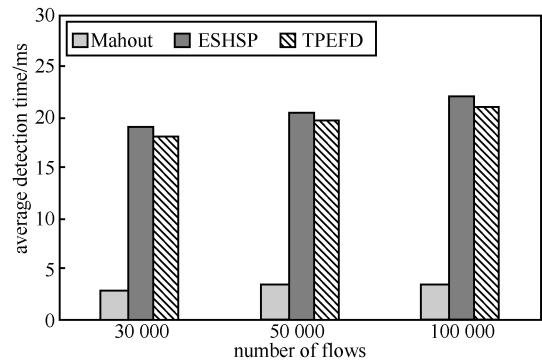


Figure 6 Elephant flow detection time

5 Conclusion

This paper proposed a two-phase, SDN-based elephant flow detection method in data center networks. We call it TPEFD method. The fact that the elephant flow accounts for most of the data center traffic promotes our work. We choose Ryu as the operating controller. The switch communicate with the controller in OpenFlow spec. v1.3. According to the traffic characteristics of data center networks, we use sampling mode to reduce the burden of the controller. In our method, we use sFlow to sample packets within a small range. sFlow needs to use equipment and important network resources to monitor each flow in the network, these lead to high overhead cost, so we filter sample packets in advance. We compare the sampling results with dynamically calculated threshold to detect elephant flow. If the sampling value exceeds the threshold value, we recognize the flow as an elephant flow. A large number of experiments and result analysis prove our new method is feasible and the detection time is efficient.

References:

- [1] AZODOLMOLKY S. Software defined networking with OpenFlow[M]. Birmingham: Packt Publishing, 2013.

- [2] GREENBERG A. VL2: a scalable and flexible data center network[C]//ACM SIGCOMM Computer Communication Review. 2009:51-62.
- [3] KANDULA S, SENGUPTA S, GREENBERG A, et al. The nature of data center traffic: measurements & analysis[C]//ACM SIGCOMM Conference on Internet Measurement. 2009:202-208.
- [4] BENSON T, AKELLA A, MALTZ D A. Network traffic characteristics of data centers in the wild[C]//ACM SIGCOMM Conference on Internet Measurement. 2010:267-280.
- [5] MORI T, KAWAHARA R, NAITO S, et al. On the characteristics of internet traffic variability: spikes and elephants[C]//The International Symposium on Applications and the Internets. 2004: 99-106.
- [6] BENSON T, ANAND A, AKELLA A, et al. Understanding data center traffic characteristics[C]//ACM Workshop on Research on Enterprise Networking. 2009:65-72.
- [7] LIN C Y, CHEN C, CHANG J W, et al. Elephant flow detection in datacenters using OpenFlow-based hierarchical statistics pulling[C]//Global Communications Conference. 2014:2264-2269.
- [8] CHENG G, GONG J, DING W. Network traffic sampling model on packet identification[M]// Berlin: Springer, 2005:758-765.
- [9] SHALIMOV A, ZUIKOV D, ZIMARINA D, et al. Advanced study of SDN/OpenFlow controllers[C]//Central & Eastern European Software Engineering Conference. 2013:1-6.
- [10] PHAAL P, PANCHEN S, MCKEE N. InMon corporation's sFlow: a method for monitoring traffic in switched and routed networks[M]// RFC Editor, 2001.
- [11] HUANG L, ZHI X, GAO Q, et al. Design and implementation of multicast routing system over SDN and sFlow[C]//IEEE International Conference on Communication Software and Networks. 2016: 524-529.
- [12] EREL M, TEOMAN E, OZCEVIK Y, et al. Scalability analysis and flow admission control in mininet-based SDN environment[C]// Network Function Virtualization and Software Defined Network. 2015:18-19.
- [13] ESTAN C, VARGHESE G. New directions in traffic measurement and accounting[C]//ACM SIGCOMM Workshop on Internet Measurement. 2001:75-80.
- [14] CURTIS A R, MOGUL J C, TOURRILHES J, et al. DevoFlow: scaling flow management for high-performance networks[C]// ACM SIGCOMM 2011 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. 2011: 254-265.
- [15] WANG H, GONG Z H. Hits and holds: two algorithms for identifying the elephant flows[J]. Journal of Software, 2010, 21(6): 1391-1403.
- [16] AL-FARES M, RADHAKRISHNAN S, RAGHAVAN B, et al. Hedera: dynamic flow scheduling for data center networks[C]// Usenix Symposium on Networked Systems Design and Implementation. 2010:281-296.
- [17] CURTIS A R, KIM W, YALAGANDULA P. Mahout: low-overhead datacenter traffic management using end-host-based elephant detection[C]//Infocom IEEE. 2011:1629-1637.
- [18] BENSON T, ANAND A, AKELLA A, et al. MicroTE: fine grained traffic engineering for data centers[C]//CONEXT. 2011:8.
- [19] ZHOU W, LI L, LUO M, et al. Rest API design patterns for SDN northbound API[C]//The International Conference on Advanced Information Networking and Applications Workshops. 2014: 358-365.
- [20] MORI T, UCHIDA M, KAWAHARA R, et al. Identifying elephant flows through periodically sampled packets[C]//ACM SIGCOMM Conference on Internet Measurement. 2004:115-120.

About the authors:



TIAN Yu (1990-), born in Shandong. She is a master student in Beijing University of Technology. Her research interests include computer networks and security.



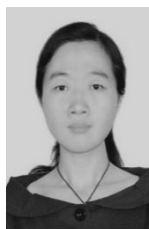
LIU Jing (1978-), born in Beijing. Master degree. She is a lecturer in Beijing University of Technology. Her research interests include network security, trusted computing.



LAI Yingxu (1973-), born in Liaoning. Ph.D degree. She is a professor in Beijing University of Technology. Her research interests include network access control, virus defense technology, network security engineering, trusted computing theory and applications.



BAO Zhenshan (1965-), born in Beijing. He is a vice professor in Beijing University of Technology. His research interests include embedded system, computer network, industrial control computer, ruggedized computer.



ZHANG Wenbo (1980-), born in Henan. Ph.D degree. She is a lecturer in Beijing University of Technology. Her research interests include heterogeneous computing, embedded systems, trusted computing theory and application.