CLASSIFICATION OF NETWORK COMPUTERS BASED ON DISTRIBUTIONS OF ICMP-ECHO ROUND-TRIP TIMES

Julian M. Kunkel
Department of Computer Science, University of Hamburg, 22527 Hamburg, Germany
kunkel@informatik.uni-hamburg.de

Jan C. Neddermeyer
Institute of Applied Mathematics, University of Heidelberg, INF 294, 69120 Heidelberg, Germany
neddermeyer@statlab.uni-heidelberg.de

Thomas Ludwig
Department of Computer Science, University of Hamburg, 22527 Hamburg, Germany
ludwig@informatik.uni-hamburg.de

Keywords: Network Discovery, Fingerprinting, Host Classification

Abstract: Classification of network hosts into groups of similar hosts allows an attacker to transfer knowledge gathered from one host of a group to others. In this paper we demonstrate that it is possible to classify hosts by inspecting the distributions of the response times from ICMP echo requests. In particular, it is shown that the response time of a host is like a fingerprint covering components inside the network, the host software as well as some hardware aspects of the target. This allows to identify nodes consisting of similar hardware and OS. Instances of virtual machines hosted on a single physical hardware can be detected in the same way. To understand the influence of hardware and software components a simple model is built and the quantitative contribution of each component to the round-trip time is briefly evaluated. Several experiments show the successful application of the classifier inside an Ethernet LAN and over the Internet.

1 INTRODUCTION

When a person plans to attack a network host it tries to exploit security flaws in the target network and especially in the host owning a particular IP address. By inferring details about the operating system and hosted services an attacker can gear the attack toward security flaws.

A strategy to characterize a network host is to analyze its communication. This can be done by reading network traffic in a passive way or by requesting some service and interpreting the response. In the latter case a vulnerable software might respond to a particular request in a well-known way.

By identifying the software an attacker knowing a security issue can pick the appropriate exploit. In the worst case it can even get full control of the host.

Intrusion detection systems deployed on the host or network try to detect suspicious network activity and artificial communication and might even counter an attack. Therefore, an attacker must be careful and disguise additional traffic as network noise.

Another countermeasure for attacks from the Internet is to fake hosts inside the network (or outside on the gateway). These so-called honeypots seem like vulnerable hosts for potential attackers. However, they are only virtual systems on which an attacker might break in but can not do any harm. On one hand an attacker might not find the real gateway among all hosts, on the other hand they might get distracted trying to break into the fake computer systems. In particular automated attacks can be deceived by honeypots. During such an attempt an administrator can even analyze the strategy of the attacker.

Consequently, for an outsider every piece of information about a system can be useful. The classification of hosts inside a network into groups of similar characteristics might open further social engineering. Knowledge that an IP belongs to a specific ensemble of hardware and software allows further to identify instances of a virtual machine hosted on the same hardware or even honeypots. Once an attacker obtains more information about one host this knowledge might be transfered to other hosts with the same characteristics.

In this paper, we present an approach to classify hosts by inspecting the distribution of the response times from ICMP echo requests. The response time of a host is like a fingerprint covering components inside the network, the host software as well as some

hardware aspects.

The paper is organized as follows: First an overview of methods to fingerprint hosts is given and our approach is compared to existing ones. Then we discuss the contribution to the round-trip time and provide a simple model. Next the classification of hosts based on a statistical analysis of measured round-trip times is described. Several experiments evaluate the potential of such a classification. Finally, our research is concluded and an outlook to future work is given.

2 STATE-OF-THE-ART

First, existing mechanisms are discussed which identify (i.e. fingerprint) operating systems (OS). Typically an OS deals with malformed, i.e. not standard conform, packets it receives in its own way because the specification of the protocol does not cover all types of invalid communication. Amongst other tools the versatile security analysis utility *NMAP* (Lyon, 2009) allows to send specifically modified TCP or UDP packets to remote hosts in order to characterize the operating system (Spangler, 2003). There are many tests which examine the reaction of an OS for a specially constructed packet. An approach for fingerprinting is to send six special requests over TCP to probe for regularity in generated sequence number, set TCP options and handling of TCP window size.

Another fingerprinting method requires a larger number of common packets. Background of this method is that an OS should generate TCP sequence numbers for a new connection randomly. However, depending on the OS the random numbers show dissimilar patterns because of distinct pseudo-random number generators used. Therefore, by characterizing the target's random number generator one can identify the OS. In detail one can look at the order of the generated numbers. Some older OS, for instance, increment the sequence number in multiples of 64,000. Another method is to characterise the probability distribution itself. For instance, the probability for drawing a number between 0 and 10,000 might be higher than for other values (Zalewski, 2001). In many cases firewalls allow to respond to typical echo requests, therefore this test can be performed effectively.

The strategies mentioned so far are *active*, i.e. they require to send probes to the target host. Network intrusion detection or firewalls could detect the malformed probes and could alert the user. There are also tools like *p0f* (Zalewski, 2006) which only read network traffic on the attacking host. The OS is identified based on network communication from or to a target host which is analyzed passively. Therefore, an attacker might get information about the target OS by using regular services on a remote host.

With all tests mentioned one can gather information about the OS but one cannot identify if two hosts are equipped with the same hardware or if they are even identical. Of course, by comparing the OS and firewall behavior a hint is given whether the machines are configured in the same way.

Yoshi Kohno et al. developed a method to characterize behavior of remote clocks (Kohno et al., 2005). Clocks in computers are inaccurate and they have an individual behavior regarding drift. With their strategy it is possible to fingerprint remote hosts, thus, it is possible to distinct two different physical devices. In this way, virtual machines or honey-pots can be identified.

In contrast to the mentioned strategies, we propose a method to classify hosts solely based on the distribution of ICMP echo response times. This method is generally applicable and does not rely on any specific characteristics of the target system. To motivate our approach, we present and discuss a simple model for the response time which depends, among other things, on the network activity.

Many papers exist which investigate and propose models for network jitter and latency. The literature provides evidence that the empirical distribution of round-trip times can be well approximated by shifted gamma distributions (Mota and Fonseca, 2000), (Li and Mason, 2007), (Chen et al., 2009). In addition, some studies found long-range dependencies (Chen et al., 2009).

3 BACKGROUND OF OUR CLASSIFIER

The goal is to classify network hosts into groups with similar characteristics. Therefore, a *fingerprint* of each host is created and compared to the profile of other hosts. The comparison can be done either through visual inspection or through a statistical test. For the moment we focus on visual inspection, however, in Appendix B we briefly discuss a more profound statistical analysis. First, we need to understand the influence of the hardware and software on the observed round-trip time. Then, the attack is described and discussed.

3.1 Modeling Round-Trip Time

Let's assume a source (S) sends an ICMP echo request to a remote computer (R). To transfer the echo

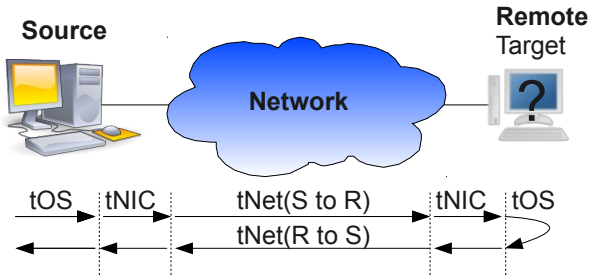


Figure 1: Model Illustration.

request to the remote host several intermediate steps are required.

A simplified transfer chain from S to R computer is shown in figure 1. First, the process must be scheduled to generate an ICMP echo request, then the OS constructs a network packet and orders the network interface card (NIC) to transfer the packet to the remote host. Now, the local area network transfers the packet to the remote host. The LAN packets might be transported by multiple switches. If the remote host is in a wide area network or the Internet, the packet is transferred via multiple intermediate routers and it might even cross administrative domains. The number of intermediate routers is called *hops*. Packet delay jitter may result from packets taking different paths to their destination to avoid congested areas or failed links. However, jitter is primarily caused by varying queuing delays because network packets compete with other network traffic at routers. Finally, the NIC of the remote host receives the packet. Once the packet is received the OS is notified by an interrupt.

The OS responds to the request by constructing an ICMP response. Depending on the OS, currently scheduled processes might be interrupted while the response is created. The response is submitted to the NIC and transferred back to the source via the network infrastructure. Once the packet is received on the source, the NIC announces the packet reception to the OS. The OS copies the response to the process which issued the ping and schedules the process. Finally, the process is dispatched and receives the response.

The mentioned processes depend on the current utilization of a component, on the software and hardware architecture, as well as on the timer accuracy. For instance, timing of the round-trip time on the source host is limited because of the precision of computer clocks and the frequency in which the OS schedules processes.

Consequently, the observed round-trip time can be modelled as:

$$
\begin{aligned}
t_{rtt} &= t_{ping} + t_{pong} \\
t_{ping} &= t_{NIC}(S) + t_{NET}(S\ to\ R) + t_{NIC}(R) + t_{OS}(R) \\
t_{pong} &= t_{NIC}(R) + t_{NET}(R\ to\ S) + t_{NIC}(S) + t_{OS}(S)
\end{aligned}
$$

All times can be seen as random variables. For the observation of one round-trip time we can assume that all variables (except $t_{NET}(S\ to\ R)$ and $t_{NET}(R\ to\ S)$) are (stochastically) independent because of disjoint hardware components. The times $t_{NET}(S\ to\ R)$ and $t_{NET}(R\ to\ S)$ might be correlated or nonlinearly dependent. In the intermediate network, phases of high or low activity can occur, for instance, caused by downloads issued by a user. This leads to longer queues in network switches which, in turn, delays transfer to and from the remote host.

The influence of the software and hardware on the round-trip time and the dependency between multiple transfers on a component are discussed in section 7.2.

3.2 Classification of Hosts by Round-Trip Time Distributions

Assume an attacker controls a host which can access the network of the hosts to be examined. By using this host the attacker has a well-defined environment (software and hardware) for probing remote hosts. The main idea is to use this computer to ping other resources in the network and to classify them with respect to the characteristics of the measured round-trip times. Our approach is based on the comparison of the (round-trip time) profiles of diverse remote hosts. In the case when they are similar we conclude that these hosts have similar characteristics in terms of hardware and software configuration.

In detail, we *ping* a set of target hosts repeatedly and collect the round-trip times. Based on these measurements we estimate the probability density function (PDF) of the round-trip time. The estimated PDF is considered as the fingerprint of a target host. For the estimation we use a kernel density estimator (see Appendix A for details). We emphasize that the PDF contains more information about the distribution of the round-trip time than simple statistics such as the mean, median, or variance.

An issue is that consecutively measured times are usually not independent as discussed earlier. For instance, applications on the target host might require much CPU which delays manufacturing of the response packets of consecutive pings. Events on the intermediate network or high network traffic issued by the target host for longer periods (e.g. big downloads or backup services) can also affect subsequent packet transfers.

The measured times can be considered as a time series (stochastic process) which is generally non-stationary. Non-stationarity means that the (unconditional) mean and variance of the round-trip times changes over time. The non-stationary behaviour may, for example, be caused by changing user and network activity. In addition, diurnal patterns (e.g. day/night activity) may be a source of non-stationarity.

Over short time intervals the processes can often be assumed to be stationary. Stationarity is desirable because it makes the estimated host profiles more reliable. However, if the interval of the measurement is very small then short term activity has high impact on the created profiles. By inspection of the time series of the round-trip times measured over a longer period it is possible to identify abnormal/extreme (short term) activity. The corresponding times can be treated as outliers and removed before the PDF is estimated. Comparison of the times series of the response times also allows to determine phases of host and intermediate network activity.

Obviously, it is desirable to gather as many round-trip times as possible to obtain a reliable PDF estimate. To reduce the influence of non-stationary and abnormal effects on our classifier we can sample round-trip times of the target hosts in parallel over a long time. Then, we can expect for example that the network activity similarly influences the estimated PDFs of all target hosts. Another strategy is to repeat the measurement at different epochs.

In this paper, we visually inspect estimated PDFs of different hosts in order to classify host similarity. A more profound statistical test for the comparison of different hosts is briefly described in Appendix B.

4 EXPERIMENTS

Major concerns of using the PDF of the round-trip times to fingerprint hosts are whether it is possible to distinguish different hardware and software configurations, and the robustness of such a strategy. To answer these questions a set of experiments is conducted.

To profile a target host we use `ping` on a Linux machine to issue a series of about 10,000 echo requests per target host. Typically the series is repeated three times, but the series are conducted in round-robin over all targets of one experiment, i.e. the first series of pings is issued to the first host then the next series to the second host.

By default `ping` issues one request to the target, then it basically sleeps for a second and issues the next request[1]. The program reports the round-trip time for each ICMP request with a maximum time resolution of $1\,\mu$. For round-trip times which are longer than $1\,ms$ only three digits are provided i.e. $125.32\,ms$ will be reported as $125\,ms$.

For our experiments various machines were used: a cluster with 9 identical nodes (hardware and software) - the machines are dual socket XEONs from 2003; several new workstations and laptops; a recent dual core XEN server from 2008 which hosts two virtual machines. Tests with Windows XP were conducted on two recent workstations. Some servers which are not administrated by us are used as reference.

We conducted many experiments. Only a selection of our results is presented here.

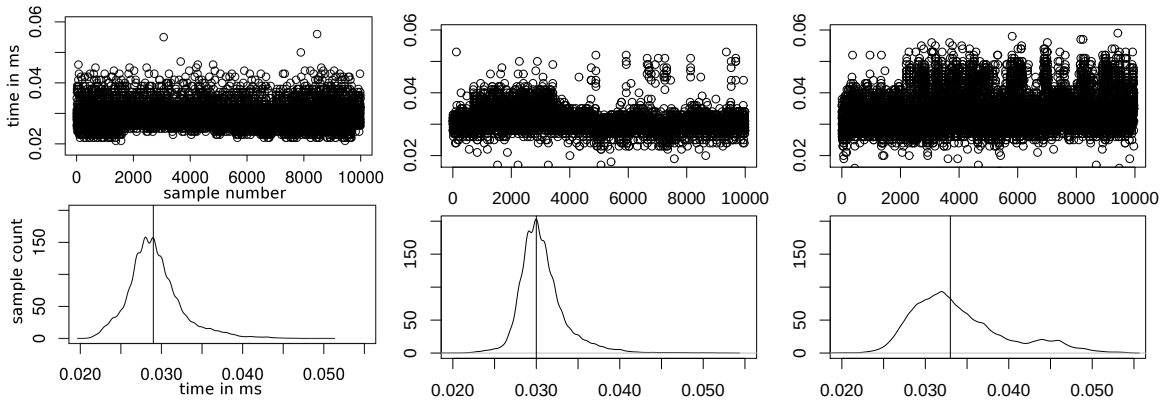
5 INFLUENCE OF HARDWARE AND SOFTWARE

In order to assess the influence of OS, network, and load on the target host a set of experiments is conducted which are described in the following.

5.1 Local Operating System Noise

First we determine the jitter introduced by the OS, i.e. $t_{OS}(S)+t_{OS}(R)$. For this purpose, we ping the loopback device on different Linux hosts. Figure 2 shows the time series, the estimated PDFs, and the medians of the response times. Labels for the axes are specified for this figure and are the same for all following graphics. Observe that even local responses (figure (a)) vary between approximately $0.02\,ms$ and $0.04\,ms$ because of OS jitter. OS jitter might be caused, among others, by interrupts and the process scheduler (De et al., 2007). In addition, on a XEN virtual machine the hypervisor imposes overhead resulting in a higher average/median response time. On the other hand, we obtain a smaller variance, probably, as a result of the fact that the hardware performance of the XEN host is better than the one of the cluster nodes bought 2003. Figure (c) shows distractions on the virtual machine caused by a user working remotely on the system. The average (median) response time increases and the distribution becomes more heavy-tailed. In this case, 95% of the response times are smaller than approximately $0.046\,ms$. For comparison, in the case of figure (b) 95% of the times are smaller than approximately $0.036\,ms$.

[1] In fact, the default setting is more complicated, but for our experiments we can assume this behavior.



(a) Cluster node (b) VM (c) VM - working user

Figure 2: Time series and estimated PDFs: Localhost pinged by a node and a virtual machine. Vertical lines indicate the medians.

5.2 Local Area Network - Influence of Utilized Hardware

Results from experiments in a small LAN are shown in figure 3. In this case, we use two cluster nodes to ping each other. All nodes are equipped with the same hardware and software and interconnected with a GigE switch.

In these experiments we used only 1000 responses. Test (a) is performed without any user interaction on the network or nodes. Next, a CPU bound process is started with increasing duration and runs about half the time during fingerprinting. The influence of the additional process on the observed distribution (b) is visible but seem negligible compared to the PDFs obtained in figure 3 (c) and figure 4. Maybe this is because of the dual socket mainboard.

In order to demonstrate the influence of a utilized network interface of the target, a third host repeatedly benchmarks TCP/IP performance to the target with `iperf`. Similar to the experiment with the CPU intensive job we increase the duration of `iperf` benchmarks and `iperf` runs about half the experiment. The influence of the utilized network lifts the round-trip time two orders of magnitude, more precisely, from $0.2\,ms$ to $20\,ms$ (figure 3 (c)).

Because of the small number of observed events the PDF estimate in (c) is less precise. However, even with this small number of repeats the idle phases still lead to three observable modes which agree with those obtained in (a) and (b).

Observations for a data server in the institute are shown in figure 4. One can see phases with longer response times caused by a mix of network activity and utilized CPU resources for serving the files. Interestingly, the support of the PDF is separated into two disjoint intervals – no observations between $0.3\,ms$ and $0.43\,ms$ were obtained[2]. During fingerprinting one could remove phases with longer response time, or, depending on the change in response times, try to classify by the role of the server.

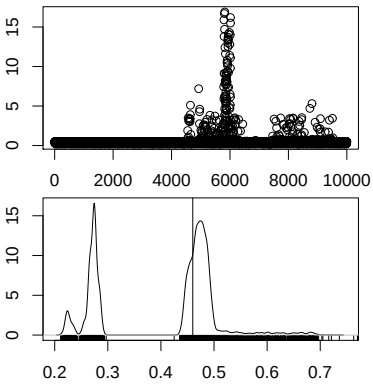
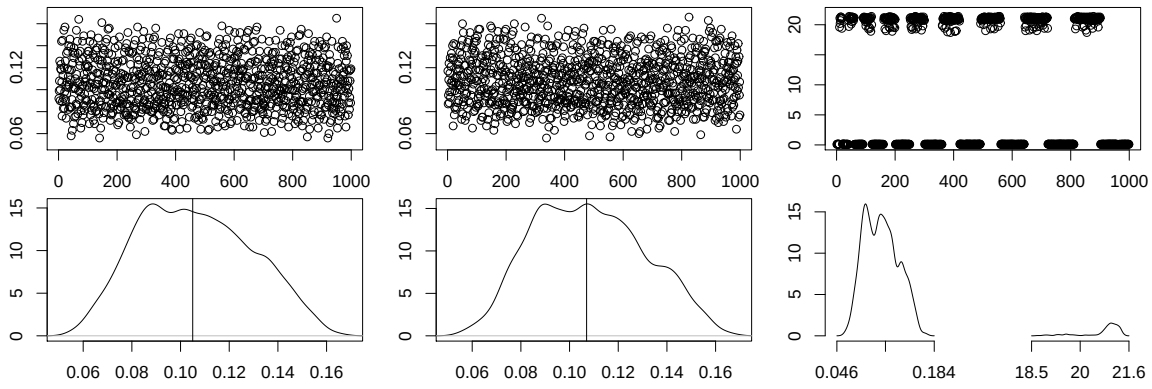


Figure 4: Time series and estimated PDF: A data server in a LAN.

5.3 Influence of the Operating System

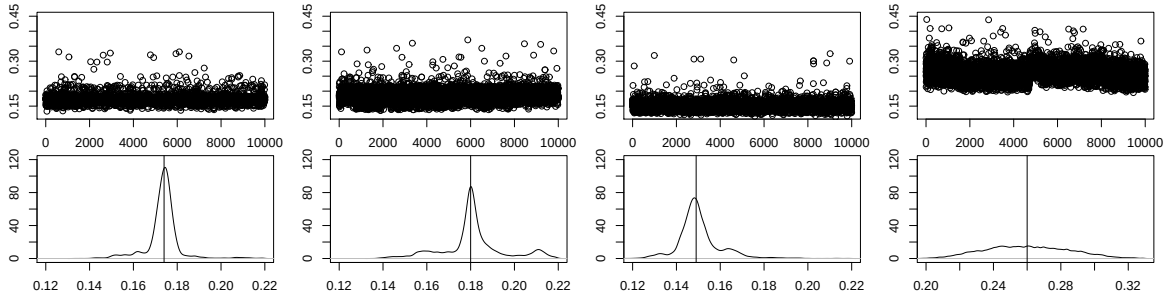
In this experiment, the impact of the OS on a given hardware configuration is analyzed briefly. Two desktop machines with diverse hardware configurations are booted once with Ubuntu 9.04 and once with the installed Windows XP SP2. A fixed third machine pings each of the targets. Figure 5 shows the results. It can be observed that Windows' response time is

[2]Note, all times larger than $0.7\,ms$ were treated as outliers and removed before the estimation of the PDF.



(a) Idle (b) Remote has one CPU intensive job (c) Remote NIC is 100% utilized

Figure 3: Time series and estimated PDFs of round-trip times between two cloned nodes in a LAN.



(a) Target one, Ubuntu (b) Target one, Windows (c) Target two, Ubuntu (d) Target two, Windows

Figure 5: Time series and estimated PDFs: A desktop with Linux and Windows in a LAN.

slower and that it has a higher variance. Comparing both, the machine response times of target two, which is even the newer PC, shows a higher variance. Summarizing, profiles of all four configurations vary and show different characteristics. The large variation between the OSs on target two might be a result of the differences in the network drivers shipped with Linux and Windows. First, we suspected the firewall is responsible, but with deactivated firewall the profile looks similar. It does not seem to be related to the Windows scheduler and interrupt handling itself because target one does not give such a flat PDF.

6 EXAMPLE CLASSIFICATION OF NETWORK HOSTS

To verify the attack several scenarios are evaluated. In the first experiment several identical nodes and the cluster frontend are fingerprinted. Another experiment shows the effectivity of the method by fingerprinting remote hosts over the Internet.

6.1 Fingerprinting in a Small LAN

In this experiment our cluster nodes node01 and node02 are used as source nodes which fingerprint either identical configured remote nodes – in our cluster we have nine nodes with the same hardware and software environment – or the cluster frontend. All nodes and the cluster frontend are directly interconnected by a GigE switch. Figure 6 shows the PDFs for three of the target nodes. In fact, for all nine nodes the profile looks very similar. Pinging the cluster frontend, on the other hand, resulted in a distinguishable fingerprint (figure 7). To show the influence of a variation of the source node two nodes were used as sources. The results of both sources are similar – for simplicity we plotted the obtained PDFs only for the cluster frontend.

When we repeated our tests a few months later a different PDF was obtained (see figure 8). It turned out that in the meantime the Linux kernel was updated by several versions. This indicates that the kernel version and network drivers lead to different fingerprints.

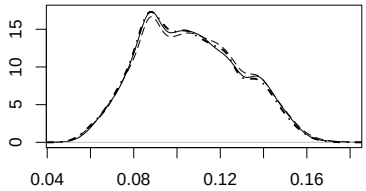


Figure 6: Estimated PDFs: Three different nodes pinged by node01 in the LAN.

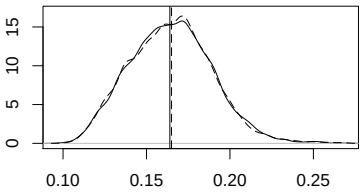


Figure 7: Estimated PDFs: The virtual machine (cluster frontend) pinged by two nodes: node01 (solid line), node02 (dashed line).

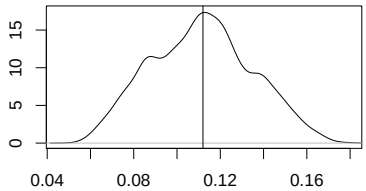


Figure 8: Estimated PDF: A cluster node after a kernel update.

6.2 Fingerprinting in University LAN

An example for profiles of hosts in a LAN consisting of multiple switches is given in figure 9. Packets are forwarded by the cluster frontend and several switches on the campus to a server in another building. Even in this case some structural information of the network is visible in the PDF. However, no intermediate router can be determined with `traceroute`. Therefore, the PDF might be a way to infer details about network topology.

Distinct PDFs are obtained by pinging five machines from an office PC: a XEN host, which deploys two VMs, and two NAS data servers (figure 10). The NAS data servers use identical hardware and a cloned software environment. The VMs are hosted on the same XEN host, however they use a different kernel (VM1 uses kernel 2.6.28.3 and VM2 an Ubuntu 2.6.24-19-xen kernel), and the VMs provide other services. As mentioned earlier we obtained three time series for each host. Unlike our other experiments, the profiles of the three repeated time series differed because of user interaction. Therefore, the time series with less interaction (most stationary) is visualized in the figures.

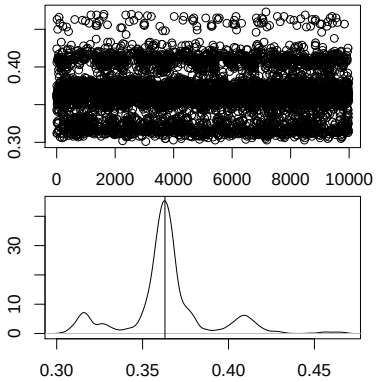


Figure 9: Time series and estimated PDF: An office PC pinged by a cluster node over the university LAN.

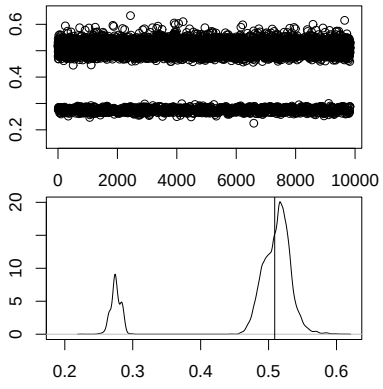
The obtained profiles of the NAS servers (figure (c)) look identical. The estimated PDFs for the two VMs show a different distribution (figure (e)) – this might be caused by the differences in the kernel versions.

6.3 Fingerprinting Internet Hosts

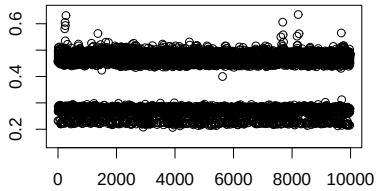
In this experiment an Internet connection from the T-Online ISP is used to ping the XEN host of the university and the two virtual machines we used in the previous LAN test. For comparison a web server hosted by T-Online is fingerprinted in the experiment as well. Time series for these hosts are recorded in round-robin during one session to T-Online. Compared to a LAN the response time of an Internet host is much slower. In our model (see section 3.1) the transport time between S and R dominates. Packet transport in the LAN resulted in a minimum round-trip time of $0.12\,ms$ where remote Internet hosts lead to $33\,ms$.

The T-Online server is chosen because the round-trip time is comparable to the university. However, packet transport to servers of the university needs at least 12 hops (the connection could not be traced after the university data center), whereas the web server of T-Online can be reached in 7 hops. The estimated PDFs are provided in figure 11. The two XEN VMs have a very similar fingerprint. In contrast, the XEN host and the T-Online web server reveal significantly different PDFs.

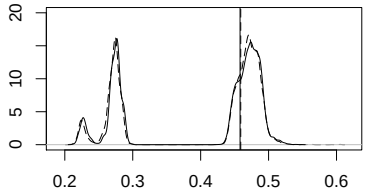
Compared to the LAN experiments it is not possible to distinguish the two XEN VMs, i.e. the software differences (kernel and services) are hidden behind network activity. However, it is possible to identify the VMs because of their PDFs' similarity.



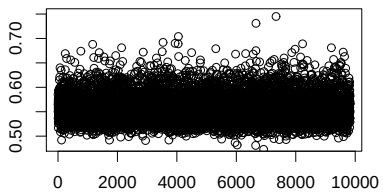
(a) Xen host



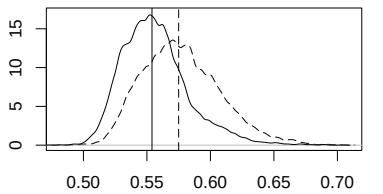
(b) NAS server 1



(c) NAS servers

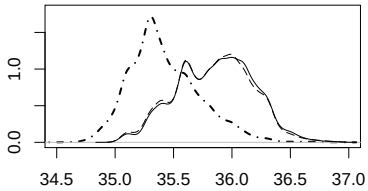


(d) VM server 1

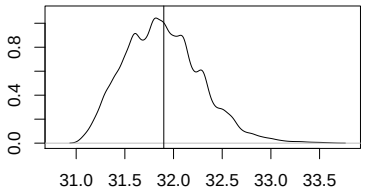


(e) VMs

Figure 10: Time series and estimated PDFs: Various hosts pinged by an office PC over the university LAN.



(a) XEN host (dot-dashed line), VM1 (solid line), VM2 (dashed line)



(b) T-Online web server

Figure 11: Estimated PDFs: Different Internet servers pinged.

7 DISCUSSION OF THE RESULTS

First, the accuracy of the classification approach is considered. Then, the obtained results are critically discussed in the context of the model described in section 3.1.

7.1 Accuracy of the Obtained PDF

For the comparison of the profiles of different hosts the accuracy of the estimated PDFs is an issue. If the observed times are stationary or even independent a small number of events suffices to obtain a precise estimate of the distribution.

However, in our case events in the LAN and computers required more than 1000 pings even for idle computers. In section 5.2 the idle process and the one with a CPU intense job differed slightly (see figure 3 (a) and (b)). A reason might be the low number of probes. The PDFs presented in figure 6 seem to be more accurate because multiple nodes and repeats resulted in almost the same PDF. In our experiments in LAN and Internet, where computers with the same hardware and software setting could be identified, 10,000 probes provided robust identification. Depending on the scenario of network and server activity more probes might be necessary.

To improve the speed of fingerprinting – a time series with 10,000 packets takes around 166 minutes – we tested the adaptive mode of `ping`. In adaptive mode `ping` sends the next ICMP request once it receives a response, but not faster than one packet per $200\,ms$. As superuser the next request is immediately

issued once the previous response is received i.e. the inter-packet interval is about round-trip time. In fact, the Internet results are obtained with adaptive mode i.e. the inter-packet interval is around $200\,ms$ compared to the normal $1\,s$ for the other tests, effectively reducing the time to 30 minutes. Still the results are promising.

However, network noise and non-stationary events caused wrong fingerprints when we used adaptive mode as superuser in the university LAN. Probably the short total time of $11\,s$ per time series is very sensitive to short term network and service noise. Note, in our university LAN the round-trip time is between 0.1 and $0.5\,ms$, but `ping` issues a request once per $1.1\,ms$ [3].

To neglect the influence of network events it is important to generate long time series. Outliers caused by short term dynamic processes or non-stationary influences could be identified within a longer time series and removed before the estimation of the PDF. We repeated the time series three times and used the round-trip time of 10,000 ICMP requests. Experiments involving multiple target hosts the time series were obtained by pinging the hosts in round-robin fashion. By comparing the three time series long-term stationary or random events influencing only one time-series are reduced and give us confidence in the correctness of the obtained PDFs.

7.2 Consistence with the Round-Trip Time Model

The experiments suggest that it is possible to classify hosts over the Internet to some extend. Keeping our simple round-trip time model from section 3.1 in mind we found this result surprising. In detail it was expected that network noise would hide details of the OS. This will be discussed in this paragraph.

To assess the latency the response times will be compared for pinging localhost, over university LAN and Internet. Minimum, mean[4], 3rd quartile, and 95% quantile for a XEN host and the VMs are provided in table 1[5].

First, the minimum round-trip time in LAN is about 6 times the time of localhost's 95% quantile. Therefore, the OS should be only a minor contribution within LAN traffic. The minimum round-trip time over the Internet is more than 50 times the response time of the LAN experiments' 95% quantile.

[3]Reported interframe gap

[4]After removal of outliers.

[5]Remember the source hosts used in these three experiments are different. Therefore, a comparison is only qualitative.

	Min	**Mean**	**75% q.**	**95% q.**	**Max**
Loopback					
VM	0.017	0.033	0.032	0.036	4.05
LAN					
Xen host	0.225	0.478	0.523	0.541	0.81
VM1	0.266	0.557	0.570	0.607	13.3
VM2	0.285	0.577	0.597	0.632	0.99
Internet					
Xen host	34.6	35.48	35.6	36.0	83.9
VM1	35.1	35.87	36.1	36.3	57.2
VM2	35.0	36.27	36.1	36.4	98.9

Table 1: Round-trip time statistics (in *ms*).

The variances of the LAN and Internet experiments are also increasing, compare the PDFs or the 3rd quartile and 95% quantile. For instance, over the Internet the difference between 3rd quartile and 95% quantile is between 0.2 and $0.4\,ms$, which is almost the average round-trip time for the LAN experiments.

Yet, it is possible to distinguish XEN host from the two VMs clearly. Within the LAN we could identify the VMs and also see differences probably caused by the variation of the kernel.

The only explanation we have is that short delays caused by the target hosts OS (or activity on the host) is propagated and possibly amplified by the network. This is possible if we assume dependency between consecutive network times, in particular between $t_{NET}(S\ to\ R)$ and $t_{NET}(R\ to\ S)$. Typically, the reversed route is used for the response of an ICMP request. Assume the situation on the intermediate network changes frequently, but depends on the old state i.e. for a short time frame it is quite similar (smooth transitions). Then, the longer the response takes to reach intermediate nodes the more variation is observable. Therefore, to some extent the network amplifies and hides local variance at the same time.

8 SUMMARY AND CONCLUSION

In this paper a method to fingerprint hosts solely based on round-trip time was introduced. Time series of thousands of ICMP echo requests are used to estimate the PDF of the round-trip time of a given host. This PDF can be compared to PDFs of other network hosts to group the hosts. Very similar PDFs are observable if hardware, software and network interconnect are identical.

Many experiments showed the application of the methodology. Over a local area network it was possi-

ble to identify virtual machines and cluster nodes. In addition, they could be distinguished from VM host and other computers. An experiment revealed that it is even possible to classify hosts over the Internet.

Gathering knowledge about remote hosts by this method enables us to investigate network hosts in a simple and non-intrusive way.

REFERENCES

Beran, R. (1988). Prepivoting test statistics: A bootstrap view of asymptotic refinements. *Journal of the American Statistical Association*, 83(403):687–697.

Chen, D., Fu, X., Ding, W., Li, H., Xi, N., and Wang, Y. (2009). Shifted gamma distribution and long-range prediction of round trip timedelay for internet-based teleoperation. *IEEE International Conference on Robotics and Biomimetics 2008*, pages 1261–1266.

De, P., Kothari, R., and Mann, V. (2007). Identifying sources of Operating System Jitter through fine-grained kernel instrumentation. In *Cluster Computing, 2007 IEEE International Conference on*, pages 331–340.

Hall, P. and Horowitz, J. (1996). Bootstrap critical values for tests based on generalized-method-of-moments estimators. *Econometrica*, 64(4):891–916.

Kohno, T., Broido, A., and Claffy, K. C. (2005). Remote physical device fingerprinting. *IEEE Trans. Dependable Secur. Comput.*, 2(2):93–108.

Li, H. and Mason, L. (2007). Synthesis of network delays for voice packets in service overlay networks. In *QSHINE '07: The Fourth International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness & Workshops*, pages 1–4, New York, NY, USA. ACM.

Lyon, G. F. (2009). *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure, USA.

Mota, A. and Fonseca, J. (2000). Dealing with jitter in systems modelling and identification. *Controlo 2000, Portugal*, pages 93–108.

Scott, D. W. (1992). *Multivariate Density Estimation*. New York: Wiley.

Spangler, R. (2003). Analysis of remote active operating system fingerprinting tools. `http://www.packetwatch.net`.

Zalewski, M. (2001). Strange attractors and tcp/ip sequence number analysis. `http://lcamtuf.coredump.cx/oldtcp/tcpseq.html`.

Zalewski, M. (2006). p0f 2 `http://lcamtuf.coredump.cx/p0f/README`.

Appendix A: Kernel Density Estimation

The kernel density estimator used to estimate the PDFs is given by

$$\hat{p}(x) = \frac{1}{Th} \sum_{i=1}^{T} K\left(\frac{x - t_{rtt}^i}{h}\right),$$

where h is the bandwidth, K is a kernel function, and t_{rtt}^i, $i = 1,\ldots,T$, are the measured round-trip times. The bandwidth is a parameter which affects the smoothness of the estimate. It was selected through cross-validation which is a standard approach for dependent data. For more details on kernel density estimation see for example (Scott, 1992). Finally, we remark that if the time series $t_{rtt}^1, t_{rtt}^2, \ldots, t_{rtt}^T$ is stationary then $\hat{p}$ estimates the (marginal) stationary distribution.

Appendix B: Statistical Classification Test

Suppose we have sets of measured round-trip times from two different target hosts. The aim is to test whether the round-trip time distributions of the two hosts differ. A standard test for this purpose is the (two-sample) Kolmogorov-Smirnov (KS) test. Let $F_{T_1}(x)$ and $F_{T_2}(x)$ be the empirical (cumulative) distribution functions computed from the times $t_{rtt,1}^i$, $i = 1,\ldots,T_1$, and $t_{rtt,2}^j$, $j = 1,\ldots,T_2$, respectively. The KS test statistic is defined through

$$D_{T_1,T_2} = \sup_x |F_{T_1}(x) - F_{T_2}(x)|.$$

The null hypothesis is rejected, that is the two distributions differ significantly, if $\sqrt{T_1 T_2/(T_1+T_2)} D_{T_1,T_2}$ is larger than some critical value. However, in our case the standard critical values do not apply because the measured times are not independent and identically distributed (which is a key assumption of the KS test). The test can still be applied but other critical values must be used. Adequate critical values can be obtained through a bootstrap method which accommodates the dependency of the data. More details on the computation of critical values based on bootstrapping can be found in (Beran, 1988) and (Hall and Horowitz, 1996).