Modelling Network Performance of End Hosts

Marat ZHANIKEEV†††,†,†††a, Member and Yoshiaki TANAKA†††, Fellow

SUMMARY In NGN standards, End Host, also referred to as Terminal Equipment (TE), holds an important place in end-to-end path performance. However, most researchers neglect TE performance when considering performance of end-to-end paths. As far as the authors’ knowledge goes, no previous study has proposed a model for TE performance. This paper proposes a method for measuring performance of TE and model extraction based on measurement data. The measurement was made possible with the use of a special NPU (Network Processing Unit) implemented as a programmable NIC. Along with the probing itself, a framework for removing the skew between the NPU and OS is developed in this paper. The multidimensional analysis includes method of probing, packet size and background traffic volume, and studies their effect on TE performance. A method for extracting a generic TE model is proposed. The outcome of this research can be used for modelling TE in simulations and in modelling end-to-end performance when considering QoS in NGN.

key words: end host, TE, Terminal Equipment, network performance, performance modelling, active probing

1. Introduction

The development of Next Generation Networks (NGN) is still a hot topic in the research community. The whole concept of NGN [1] is based on Quality of Service (QoS). In NGN, connecting two end hosts is not just a matter of finding a path between them but also and especially finding the path that satisfies user requirements in regard to QoS. Unlike the case of traditional network, end hosts of end-to-end paths cannot be ignored in NGN. Figure 1 shows the model of end-to-end QoS defined within NGN standards. While traditional model dealt with network-only performance, NGN also includes end hosts. This paper will use NGN’s terminology and will refer to end hosts as Terminal Equipment (TE) [2].

Traditionally, TE’s effect on end-to-end performance has been neglected. Only a few studies looked into this aspect. Most of them were carried out on Linux or another Unix-like platform [3].

End-to-end performance is well standardized. IPPM [4] defines a standard set of metrics of end-to-end network performance. Also, Y.1540 [5] recommendation contains practical advice on how to conduct measurements in order to obtain values for these metrics.

The objective of this paper is to study the network performance of TE under various conditions and propose a method for extracting a model for TE. The main target is to support the recognition of the importance of TE in end-to-end network performance. Outcome of this paper can serve as a reference for future TE performance standardization and modelling. Specific models proposed in this paper can also be used in simulations to model realistic TE systems.

This paper implements the following scope. Six distinct packet sizes (40, 64, 576, 1000, 1280, 1500 bytes) are used as the most frequent packet sizes in the Internet. Single-packet and multiple-packet probes are used to cover a wide range of communication patterns, including window-based TCP bulk transfer.

The implementation and the findings of this paper are platform-independent. Although this paper uses Windows operating system for implementation, the measurement model minimizes load both at application and network card, so, replacing them has minimum effect. It was concluded from data that performance models should not change much in other operating systems, given that implementations of UDP and TCP stacks is very similar on most systems.

The proposed probing and modelling method can be used to create models on any operating system on the fly. Unfortunately, NPUs are extremely rare, which is why the authors were restricted to a singular platform for implementation.

2. Related Works and Motivation

Performing measurement inside TE is a very difficult task, which partly explains the lack of studies on TE network performance. The study in [3] was implemented by connecting 2 PCs by cross-over cable. Delay between the two machines was measured. Also, source code of network stack in the
kernel (Linux OS) was analyzed in order to deduce delays inside TE. One of the findings in [3] was that context switching triggered by OS kernel caused large pauses in packet inter-transmission times, which made it impossible to transmit large trains of packets back-to-back and put a cap on maximum achievable throughput.

Another way to conduct such a measurement study is to use 2 short-circuited NICs [6], so the measurement can be done within the same machine. This removes the need for time synchronization because both receiving and transmitting processes can be on the same machine. The study in [6] deducted delays within network stack inside OS kernel by analyzing its source code, as well and also mentioned context switching as one of the main artifacts affecting throughput.

There is a problem with analysis of kernel source code. Few operating systems open their source code to public. For example, Windows OS, the most widely used by clients OS today, does not open its source code to public. In order to be able to measure TE based on any OS, there is a need for ability to perform measurement inside the host without having to access kernel source code. Also, the main target of this study is not to find out how kernel works, but to create a working model of how an OS kernel handles traffic under various conditions. Specifically, measurements in this paper will stress network stack inside kernel by subjecting it to cross traffic, and will measure how this stress is handled by the kernel.

Modelling based only on measurement results needs as high precision of measurement as possible. Earlier studies have connected two machines by a hub or created a loop back to the same machine. Such a measurement setup suffers from errors. When two machines are used, errors come from Ethernet frame collisions in the hub and from the need to synchronize the two machines. When both transmitting and receiving ends are inside the same machine, errors come from interference between these two processes.

It is possible to achieve a higher measurement setup if a measurement point exists at NIC level. In this case, both measurement ends are inside the same machine but they do not interfere with each other. Special network interface cards are required for such a measurement setup. Such cards are called Network Processing Units (NPU). NPUs allow for running code onboard the NIC, where it does not interfere either with software running at application level of OS or code running inside OS kernel.

One example of NPU is DAG card (Data Acquisition and Generation card) [7]. It is a NIC that allows packet processing onboard and carries a high precision clock. In [8], for example, it was used to monitor traffic and to classify and filter the captured traffic on the card itself. However, due to its price, it is mainly used for applications that require extremely high clock precision, which was the case in the studies in [9] and [10], which exploited the DAG cards specifically because of the high precision of its timestamps.

The measurement in this paper is based on a NIC similar to the DAG card but with slightly inferior clock precision. However, as will be shown later in this paper, this inferior clock precision is still within the acceptable range given the target of the measurement.

3. NPU, Skew, and Measurement Setup

The smart NIC (NPU) used in this paper is Killer NIC [11], which is a network card specifically designed by Bigfoot Networks for network games. In fact, it is not just a network card but also carries a small operating system optimized for delay and performance. In this paper the term NPU will always refer to Killer NIC.

In order to probe TE, two measurement points are essential: application level measurement point and NIC level measurement point. However, in a conventional OS the NIC level is not accessible from user space in OS. Since NPU offers the opportunity to run applications on the NIC itself, and given that the Linux version running onboard NPU inflicts a negligible one way delay (the order of a few microseconds) compared to the one way delay of the probing itself, the use of NPU as a measurement point at NIC level is justified. One way delay (OWD), the properties of which are studied in this paper, is defined as one way delay between application layer in OS and NPU. This is the desired definition, because network between these ends includes only network stack code inside OS kernel.

The most important objective in such a measurement is to free both measurement ends from interference with other processes. The measurement setup explained below is created with this objective in mind. Both measurement points, one at application layer of the machine and the other one inside NPU, run 2 processes each, one for skew synchronization between the ends and the other for the main measurement tasks. Skew synchronization does very little work and runs at long intervals, which causes minimal interference. This means that each end runs only one intensive application at all times, which follows that interference is minimal.

The measurement setup using NPU is illustrated in Fig. 2. TE used for the measurement study is a DELL Inspiron 530 Core 2 Duo machine with 2.66 GHz clock and 3 GB RAM, running under Windows XP. NPU is Killer NIC K1 with 333 MHz clock and 64 MB RAM.

There are three separate packet streams as follows. First, there is a continuous skew measurement which is done round trip from NPU to OS and back to NPU. The main measurement process is conducted one-way from OS to NPU. Finally, it is useful to be able to test TE’s network performance under varying conditions in background traffic.
Background traffic packet stream originates at several other machines and enters the host through another NIC distinct from NPU. This guarantees that only network stack in TE’s OS is under stress from background traffic, while NPU gets no additional stress. No receiver is running in TE’s application space for background traffic, which guarantees that background traffic stresses only the network stack within OS. However, the absence of receiver does not stop the background traffic flowing in from the outside to be proposed by OS kernel. This is the intended feature, we need to stress OS with background traffic but we do not want additional interference at application layer.

Skew measurement is required because NPU is decoupled from OS and the clocks between the two measurement points skew with time. The clock offset is composed of the initial clock drift and the ageing as defined in [4] and [12]. The ageing can be calculated from the clock skew and the elapsed time since the time reference. Given that the proposed analysis requires values in sub-millisecond range, the skew between the clocks should be managed in this range.

In [13], the initial OWD between the two clocks is estimated by linear fitting. The chosen initial delay corresponds to the pair of skew and initial minimum delay, corresponding respectively to the slope and y-intercept of the closest line under the data points. A similar approach is used for skew removal in this paper.

Figure 3 shows the basic algorithm used for skew measurement. In the figure, \( t_{NIC} \) is the departure time at NPU, \( t_{NIC2} \) is the arrival time at NPU after round trip, and \( t_{OS} \) is the arrival time at OS.

Skew probes are implemented as UDP packets and are sent continuously. If we denote time span between two \( t_{NIC1} \) times as \( \Delta t_{NIC} \), and time span between two \( t_{OS} \) times as \( \Delta t_{OS} \), then, clock skew can be calculated as:

\[
s = \frac{\Delta t_{NIC} - \Delta t_{OS}}{\Delta t_{OS}}. \tag{1}
\]

Also, minimum OWD delay between NPU and OS can be found as:

\[
minOWD = \min\{0.5(t_{NIC2} - t_{NIC1})\}. \tag{2}
\]

Figure 4 shows an example set of consecutive 1000 skew measurements. Absolute OWD value gradually increases as clocks between OS and NPU deviate further from each other. The trend, however, is linear, which justifies the uses of linear fitting explained above. In fact, the linear trend of the skew is good news because non-linear trend would require a more complex version of fitting for raw measurement samples.

The right side of the figure shows the frequency of values for \( s \). In this example sets of 1000 skew measurements, the value 0.0076, which corresponds to a skew of 7.6 ms/s, appears more than 800 times, which is the overwhelming majority. Moreover, despite the fact that the actual value of the skew is large, its drift is stable since only a fluctuation of 0.2% was observed in data. Also, distribution of values for \( s \) has a nearly perfect bell-shape curve in Fig. 4(b), which means that although skew can randomly deviate from its central mode, these deviations are random and located on both sides of the mode, successfully cancelling each other in the long run. These results show that the precision of time synchronization between NPU and OS is \( 10^{-4} \) seconds, which is sufficient for the current study.

Although Fig. 4 only works with 1000 samples, in main experiments skew measurement is performed continuously in order to provide continuous skew reference for main probing and background traffic.

During continuous skew probing, probe intervals are picked randomly between 5 and 10 seconds apart to satisfy PASTA property [14] and avoid regular artifacts in traffic. PASTA is a simple property which states that a random process with unknown model is best sampled by a random process. As was shown in [14], any form of randomization of sampling interval is acceptable as long as sampling intervals are not strictly periodic.

4. Single Packets

This section explains how main probing with single packets is implemented and its data analyzed. Main probing is performed in parallel with continuous skew probing.

Estimation of the actual OWD is a complicated task given the de-coupled system between OS and NPU. Applying wrong values for skew will result in appearance of false trends in OWD values.

The algorithm for single packets is as follows. For a given \( t_{NIC} \) of a given single packet, all skew measurement records between \( t_{NIC} - 20x \) and \( t_{NIC} + 20x \) are extracted, resulting in between 4 and 8 skew probes. To de-skew the OWD set from main probing, \( s \) and \( minOWD \) are calculated according to (1) and (2). All OWD values from main probing are skewed by \( s \), then \( minOWD \) is subtracted. The
resulting dataset contains OWD samples relative to global minOWD.

In modelling further in this paper, single packet probing will be considered in the following contexts:

- various levels of background traffic at the time of probing;
- various packet size used for main probing.

The raw data for analysis in all these cases will be the de-skewed and offset OWD data in accordance to the algorithm above.

5. Packet Bursts

This section details the method used to implement multiple-packet probes and analyze OWD data from such measurement. All bursts in experiments are 100 packets long.

Skew for bursts is also calculated according to (1) and (2), i.e. all skew samples between $t_{NIC} - 20s$ from the first and $t_{NIC} + 20s$ from the last packet in burst are picked used to calculate $s$ and minOWD. We get de-skewed and offset list of OWD values for all packets of the burst.

For the case of bursts, packets are sent back-to-back, with no space in between. The network socket is reset after the transmission of each burst to make sure that network resources are used independently for each packet burst.

In this paper, the burst size is measured in number of packets. However, it cannot be guaranteed that OS will dedicate all of its resources to transmission of the burst. From time to time, especially if transmission takes a relatively long time, OS will forcefully reclaim control from the sending process to execute other actions. The control will only be yielded back to the burst-sending process after the completion of those actions. In OS terminology, this is called context switching.

Consequently, the packets immediately after the context switching will have a very high OWD compared to others. Those abnormally high OWD values are referred to as peaks. And depending on how long the sending process lasts, such context switching might happen several times during one burst. It was discovered from experiments that the number of peaks in a burst and when they appear depends solely on the packet size. Therefore, in the analysis further in this paper, peaks and periods between them, where the period is measured in number of packets, will be considered.

Context switching was found to cause long delays by other research before. Both [3] and [6] came to conclusion that context switching is a major cause of long delays which can interrupt transmission of packet streams. Also, context switching is a logical explanation for peaks because a context switch will always interrupt a session of continuous transmission of packets.

Here is some example findings based on measurement data. For 64-byte packets, for example, the entire burst of 100 packets can be sent in about 8 ms, which is fast enough not to encounter context switching within OS, and consequently, most of the time no peaks are found in data. Probes with bigger packet size, such as 1500 bytes, will require about 10s to be sent, which is long enough for the sending process to be interrupted several times, which is why there may be several peaks in data.

In order to investigate the nature of peaks, we debugged the code in higher detail. It was confirmed that the gaps in inter-transmission time happened at the time when the code calls send() OS system call. By definition, system call is when code from application layer can call a routine which exists inside kernel. Also, when a system call is executed, code in application layer willingly yields control to kernel and blocks until system call returns. Through detailed debugging of our software, it was confirmed that the gaps in inter-transmission time came from prolonged blocking times between calling the send() system call and getting the control back from the kernel.

Figure 5 illustrates properties of a randomly selected burst which used 1500-byte packets. The peaks do not appear at the beginning and at the end of the burst. They only appear in the middle. In the beginning, packets are sent normally and interrupted rarely. But as transmission of the burst takes longer, the OS begins to treat it as a routine action. As a result, the packets in the buffer start to pile up. With the buffer continuously filling up, OS interrupts the process periodically, which results in the periodic nature of the peaks. Toward the end, packets stop arriving into the buffer, and the periodicity disappears until the entire buffer is emptied and socket is closed.

This figure alone is sufficient justification for studying TE’s effect on end-to-end networking performance. As is shown in Fig.5, Some peaks can be as large as 1s. This means that even if application needs to transmit a long packet train back-to-back, OS will regularly interrupt the application and will make it impossible to transmit an uninterrupted train of packets. It should be noted that both NPU and application layer were not under outside stress at the time of probing. As was explained earlier, measurement setup was created in such as way that stress is inflicted only on network stack inside OS, while both application layer of OS and code running on NPU have zero interference, meaning that receiving and transmitting programs are the only programs running on each end at a time.

Because of the presence of peak OWD in bursts, the analysis of the OWD of burst can be split into two parts:
analysis of the OWD without the peaks as function of packet size and background traffic, and a separate analysis of only peaks as function of packet size.

To establish the necessary terminology, the list of OWD values with peaks removed is called filtered OWD, or \( f_{OWD} \), as shown in Fig. 5, and its average is denoted as \( \mu_{f_{OWD}} \). Distance between peaks is referred to as peakPeriod and is measured in number of packets. Peaks are distinguished from the rest of the packets using three-sigma rule, where the threshold is set as three standard deviations away from the average. This threshold was tested over a large volume of raw data and proved to be very successful.

Figure 6 displays trends for \( f_{OWD} \) and the ratio between peaks and \( f_{OWD} \) for burst. To cover a wide range of OS conditions at the time of probing, each curve is average of over a hundred probes, all of which experienced the same level of background traffic. Figure 6 expresses background traffic as maximum background traffic received by the machine at the time the main probe was sent, or \( \text{maxBgRx} \) for short. The entire range of background utilization is tested. It looks like the trends in both metrics are retained regardless of the level of background traffic.

Both parts of Fig. 6 are useful for understanding how OS handles bursts. Let us consider the outcome walking in the order of increasing packet size.

First, for very small packet size, \( f_{OWD} \) shows higher values. However, at the same time, most probes of smallest packets have no gaps in inter-transmission time. This means that \( f_{OWD} \) values are in most cases not a subset, but the entire list of samples. Also, when there are zero or very few outliers with little absolute difference, it will naturally increase the value of average over all values. This is the reason why \( f_{OWD} \) for very small packet size appears higher. It is merely a statistical artifact coming from inability to properly separate peaks (outliers) from the main data.

For each next larger packet size, \( f_{OWD} \) and ratio of peaks to \( f_{OWD} \) are cleanly separated. Peaks for these packet sizes are substantially high and \( f_{OWD} \) subset of samples can be easily filtered from raw data. For most of higher packet sizes, the ratio between peaks and \( f_{OWD} \) is over 1000.

Packet size of 1280 behaved slightly unusually. For this packet size, not all peaks could be cleanly separated from \( f_{OWD} \) and as a result, \( f_{OWD} \) is much higher than for neighboring packet sizes. Naturally, average ratio between peaks and \( f_{OWD} \) is decreased as well. This is, again, a purely statistical problem coming from inability to cleanly separate all peaks from \( f_{OWD} \). However, if both parts of Fig. 6 are considered together, the peak for packet size 1280 in Fig. 6 (a) and dip for the same packet size in Fig. 6 (b) are independent. With this in mind, it can be stated that the overall trend in Fig. 6 (a) is slowly increasing with packet size. The overall trend in Fig. 6 (b) is rapidly increasing with packet size.

Figure 6 (a) has a somewhat discrete appearance, where the discrete unit is delay of 1 ms. In fact, the minimum OWD detected in data came out to be 1 ms. This does not however, mean that one way delay is always 1 ms. This artifact is due to two factors. First, skew measurement precision is \( 10^{-4} \) s, which allows for sufficient precision within one tenth of 1 ms. However, when skew is applied specifically to OWD, the process of de-skewing suffers from lower precision. When skew is measured, its timestamps both at OS and at NPU are recorded. Time precision at NPU is 1 \( \mu \) s but on OS side time advances in discrete 1 ms steps (was verified in practice). This can be visualized as a step-function, where NPU time advanced in 1000 times smaller intervals than OS time, where the latter advanced in a step-function fashion roughly one per 1000 changes in NPU time.

Therefore, when de-skewing time for OWD samples, the outcome for \( f_{OWD} \) shows discrete features, where it roughly falls within a multiplier of 1 ms. Values around 1.5 ms in Fig. 6 therefore come from roughly the same number of 1 ms and 2 ms values when calculating the average. 2 ms values roughly indicate that OS time advanced 2 times before the measurement was made.

The actual OWD may and should be much smaller than 1 ms for majority of samples. Given the above artifacts, lowest values for OWD happens to be displayed as 1 ms – the clock resolution of OS side. However, relative difference of OWD across packet size has a genuine trend. Even with the discrete states in Fig. 6 (a), the trend is still visible.

Some practical assumptions can be made based on that data. First, even with peaks removed, \( f_{OWD} \) exhibits increasing trend with packet size. This indicates that processing of traffic inside OS kernel depends on packet size. The most likely cause of this effect is checksumming, which in most operating systems is implemented in software and requires as many loops to process as there are bytes in a packet. In fact, it is an established fact that in high-end networking devices, checksumming is moved to hardware, specifically, to increase efficiency. However, in conventional client terminals, majority of OS implements checksumming in software. Unfortunately, these authors cannot confirm this assumption at this time.

Since OWD performance in bursts depends on packet size, it is logical to assume that throughput will also vary with packet size. One can imagine a trade-off relation between packet size and throughput. Although it is logical to assume that larger packet size automatically guarantees higher throughput, Fig. 6 shows that bursts with large pack-
ets suffer from long pauses (pick to average ratio) in the middle of bursts. Values for JOWD are the same for both extremes of packet size, so, it can be expected that these pauses play a crucial role in this trade-off.

Figure 7 shows that smaller packet size is the obvious winner in this trade-off. For small packet size, each packet processing is very fast and packets have almost no waiting time. Also, the absence of peaks for small packet sizes further improves the efficiency of transmission.

With packet size of 576 bytes and above, the throughput performance is roughly the same and is about 2-3 times lower than that of smaller packets. The packet size of 1280 shows up as a spike in this figure as well, but its throughput is not very distinct from neighbouring packet sizes.

### 6. Trends, Verification and Justification

Some of the results indicated in the two previous sections are not intuitive and require additional justification. In fact, as was mentioned before, one should normally expect that larger packet size should guarantee higher throughput regardless of environment. In order to verify the findings, additional tests were performed using a normal/conventional NIC instead of the NPU, with the same experimental setup as was used in [6]. Also, additional literature was found to justify some of the found artifacts. Based on that work, the following justification can be offered.

First, it is important to remember that throughput in this paper is a short-term property calculated based on a single burst of 100 UDP packets. No end-to-end communication control was used, but probes with dropped packets were excluded from data analysis. In short-term, Fig. 7 shows that bursts with small packets outperform those with large packets. In case of long-term communication, it is possible that the overhead from connection control (ACK packets, etc.) will play a considerable role in this trade-off. However, one can still imagine a long-term communication where short-term bursts are used as a basic unit of communication. In this case the results from Fig. 7 still stand as true.

Second major factor which is left out of scope of this paper is multi-hop communication. In fact, the main target of this paper is to model TE, not a multi-hop path including TE. In multihop communications, there are several new players at hand, such as congestion, packet size discrimination in routers, dependence of packet loss of packet size, etc. With these players, the above trade-off may change considerably. This research is envisioned by these authors as the next logical step in research on TE modelling and will be covered in future publications.

Finally, the effect of the NPU itself has to be ruled out. Two effects are obvious immediately. First, it is unclear why the NPU with its 1 Gbps physical interface is found to achieve at most 6 Mbps throughput. Secondly, it had to be verified whether experiments with a conventional NIC would indicate the same trends as found in Fig. 6 and Fig. 7. Experimental setup for conventional NIC was repeated directly from [6].

Virtually identical low figures for throughput were found with conventional NIC when communicating between two Windows machines. Linux to Windows and Linux to Linux communication showed drastically higher throughput, closer to 100 Mbps – the physical limit of the Layer 2 networks used for experiments. Windows to Linux has the same level low level of throughput as for Windows to Windows.

Experiments indicated that Windows OS was the main cause for the low throughput. Much better performance by Linux can be explained by the software realtime feature present in Linux kernels above 2.6.x, which substantially boosts efficiency of CPU utilization by kernel tasks, specifically interrupt processing.

CPU overloading by interrupt processing was found to play a major role in high-rate communications in [17]. In fact, the paper proves that using conventional NIC on a conventional OS practically guarantees inability to achieve 1 Gbps throughput. In fact, it was showed by that research that 1 Gbps NICs cause CPU overloading and interfere with applications.

Windows OS has 1 ms time slice. However, as was earlier measured and shown in Fig. 5, kernel may sometimes interrupt application layer for much longer, in some cases up to 1s. Since during this time only communication experiments described above are executed, it is valid to assume that kernel uses all this time to process networking-related tasks, including NIC or NPU interrupts.

Having performed a larger set of experiments with a conventional NIC, roughly the same trends were discovered. Specifically, bursts with the smallest packet size had a distinctly larger throughput. Bursts of larger packet sizes had large peaks (gaps in processing) and their throughput suffered in result. This proved that OS kernel is the main cause for the effects found in Fig. 6 and Fig. 7. In fact, measurement and modelling of these effects is the main objective of this paper. The particular values for metrics may vary across hardware and software, but the measurement and modelling methods proposed in this paper should remain valid.

More help in justifying the discovered trends was found in [6], both in theoretical and practical form. The paper split end-to-end path between two machines into several distinct stages. One of the stages was for processing which happens between the time an application sends a packet and the time the packet is physically transmitted by NIC. The paper
shows that delay in that stage depends on packet size. In fact, results in [6] indicate that the dependence is a non-linear function, where smallest packets experience a distinctly smaller delay than other packet sizes.

Unfortunately, [6] does not perform the same measurements in immediate sequences of packets, which would allow for the capture and analysis of gaps caused by context switching in kernel. Context switching is only mentioned by the paper but no actual measurement is performed. These authors could not find any literature which would study the effect of context switching on throughput by a practical setup.

One final note about the NPU’s 1 Gbps physical interface is required. As was explained earlier when introducing measurement setup, the NPU does not actually communicate to the outside world, so, its physical network interface is not actually used. NPU is simply used as a measurement point on the other side of the kernel, with application layer being on the opposite side.

7. A Model of TE

Given the multi-parametric analysis space in this paper, the results in previous sections only covered a subset of all possible parameter combinations. This section will present the rest of the data in form of a statistical model describing various performance properties of TE. Packet size, background traffic rate, and probing method are the three parameters constituting the analysis space.

Figure 8 shows the algorithm used to extract a model over the multi-parameter analysis space. First, there are three loops, each looping through a set of values for each parameter in the multidimensional parameter space. For probing method, single packets versus packet bursts are used. Packet sizes are 40, 64, 576, 1000, 1280, and 1500 bytes. Background levels are picked at 20, 40, 60, 80, and 100 Mbps.

The core of the algorithm involves probability distribution fitting which is where a set of samples is fitted with a given distribution in the result of which coefficients for the closest approximation of data by that distribution are obtained.

Fitting was performed separately for each combination of the three analysis parameters. Also, in order to eliminate noise and randomness in data, probing was conducted until over 10000 samples were collected for each combination of analysis parameters. Because fitting over that many samples may obliterate patterns in data, partitioning was performed as shown in Fig. 8, with coefficients for each partition aggregated as averages at the end of each loop.

The last remaining problem is the selection of the probability distribution model to use for each fitting. This is a difficult problem which requires not only multiple trials for each partition but also a mechanism to verify the validity of coefficients obtained by fitting. As shown in Fig. 8, the main idea is to select the most appropriate distribution for a given set of samples. Initially each partition is fitted using various commonly used distributions. Each fitting result then undergoes a Kolmogorov-Smirnov test, which evaluates the validity of the fitting. The test yields two parameters, the p-value which is used to determine if the distribution is valid or not, and the test statistic which represents the distance between the data from the partition and each sample generated from the fitted distribution.

After all the distributions in the set are tried, the test statistics are compared and the distribution with lowest test statistic is selected as the most appropriate for the partition. After the best distribution for each partition in a given set of samples has been determined, the most frequent distribution is chosen as the distribution most fit for that particular partition. The coefficients for the set of samples are obtained by calculating the average of the coefficients of all partitions.

The experiments lead to the result that bursts are best fitted with Weibull distribution while Gamma is best to fit samples from single-packet probing.

This paragraph provides an example of obtained model for TE, according to the configuration in this paper. Not all partitions can be fitted with a simple distribution. Some partitions are multimodal, which is not really a probability distribution model in itself, but is a complex random process. When a multimodal distribution has to be emulated in practice, it is normally done with a mixture of different distribution models running simultaneously. Statistical modelling using multimodal distribution is a complex subject and is beyond the scope of the present paper.

Since bursts are fitted with Weibull distribution, the distribution coefficients are shape and scale. Figure 9 shows the dependence of coefficients from fitting fjOWD on packet size and background traffic. Packet size has more influence on the fitting parameters than background traffic, as shown in Fig. 9.

For small packet size, due to the values of the shape, the distribution is closer to normal distribution. On the con-
tary, for large packet size the distribution is closer to exponential distribution. Small packet size has higher scale, which means that the distribution has a bigger spread. Consequently, the average loses its weight and different values of OWD have almost the same probability to occur in the burst. For larger packet, the distribution is less spread, meaning that $\alpha f_{OWD}$ has a very high probability to occur in the burst. This can be explained by the presence of maximum values in small packet size burst and the removal of peaks in larger packets, the remaining OWD have roughly the same values.

Single packets follow Gamma distribution. Usually, Gamma distribution is described with the parameters shape and scale. However, the terms in R [15], which was used for the fitting, the parameters of Gamma distribution are shape and rate. Due to the high volatility of single packets, not all the partitions could be fitted successfully. This means that data for small single packets oscillates between Gamma distribution and multimodal distribution.

The oscillation of single packets between Gamma and multimodal distribution decreases with packet size. For example, 1500-byte packets get very good validity test results when fitted with Gamma distribution. This means that they are less volatile as well. Since the processing of larger packets takes longer, they are less vulnerable to fluctuations in internal OS processes, resulting in a more steady performance.

When fitting parameters from multiple partitions were collected and probabilistic density curve was calculated for all combinations for the three analysis parameters, the following features were found for such curves:

1. When most partitions yield similar distribution parameters, the resulting density curve is smooth and narrow.
2. If distribution parameters vary, but majority of values are around a single mode, we can call such curve smooth and wide.
3. If distribution parameters cluster around multiple modes, such a curve can be called irregular, because it means that outcome for distributions parameters are hard to predict.

In summary, the logic presented in Fig. 10 can be used to model TE in practice.

When modelling single-packet probes, the following rules can be established:

1. TE performance can be modelled with common distributions, depending on the probing method, packet size and background traffic.
2. Weibull distribution should be used to model bursts and Gamma distribution should be used for single packets.
3. For small single packet probes, the process cannot be modelled by a single distribution model, instead a mixture of models has to be used to emulate multimodal distribution.
4. For large single packet probes, Gamma distribution parameters can be selected randomly from a narrow distribution with a single mode.

When modelling packet bursts, the following rules can be established:

1. Weibull distribution can be used to model packet bursts.
2. When packet size is small, distribution parameters have to be picked from an irregular random distribution, which can be emulated with 2-3 random pools each with a different mode.
3. Combination of small packets and low background traffic shrinks the range of random values for distribution parameters.
4. When packet size is large, distribution parameters are to be picked from a wide single-mode random distribution.
5. Large packet size is not majorly affected by background traffic, however, low background traffic may cause irregularity in distribution parameters while high background traffic will, in fact, promote stability in distribution parameters.

8. Conclusion

This paper created a measurement setup capable of detecting performance properties of TE, performed the measurements over a multi-parameter space of various packet sizes, background utilization level, and probing method, and finally extracted a statistical model based on collected results.

Measurement was made possible with the use of a programmable NIC with latency-optimized Linux onboard. In order to solve the clock skew problem, skew measurement
was performed continuously and independently from the actual probing itself.

The analysis based on collected measurement data was multidimensional in nature. Performance of TE was studied in two steps using varying probing method, packet size and intensity of background traffic. In the first step, the raw data was considered and particular performance properties were presented. In second step, a distribution based statistical processing was used to extract a model.

It was discovered that bursts are more predictable than single packets especially when single packets are of small size. On the other hand, several times during a burst packets encountered extremely high OWD, up to 1s, which is larger than the overall end-to-end delay of a network path, and therefore is a considerable performance artifact. Additionally, at low background traffic, statistical model is affected only by packet size.

The paper found direct evidence that probing is affected by processing within operating systems running on TE s. In fact, the following are the main candidates for the sources of these effects. They are platform-independent because they can be found in any operating system.

1. Checksumming: which results in the fact that processing large packet takes longer than smaller ones.
2. Context switching based on time slot resolution of OS: even if the resolution differs from one OS to another one, the value is still a finite one, which causes the appearance of peaks in bursts. And, in some cases, the time slot resolution might be related to checksum processing time.

Despite the fact that only one conventional OS was used to perform the measurement on which the model extraction in this paper is based, the model extraction itself can be generalized and it is safe to assume that it can be applied to any kind of OS. In fact, even if the OS differs, some common components remain, namely checksumming and context switching based on time resolution of OS. Consequently, the same main performance characteristics such as existence of peaks and higher OWD for larger packet size still hold, which means that the model extraction method can still be applied.

Although lack of access to source code of most operating systems makes it impossible to confirm whether checksumming and context switching are the main causes of extremely long processing delays, conclusions of this paper agree with conclusions made by a study in [6] before. Specifically, very long gaps were found in transmissions using large packet size, which points to context switching as the most likely cause. Also, processing time is slightly larger for larger packet size which indirectly points to checksumming as the most likely cause. In general, this paper found the same artifacts as were found in [3] and [6] before. However, the value of this paper is in being able to find the same artifacts via probing rather than inspection of source code.

In the future, the authors plan to complete the study by getting more results and more tests on other machines and OS types as availability of cross-platform programmable NICs improves. The current methodology can also be used in analysis of end-to-end QoS provisioning which takes into account performance of connection ends.

References

[1] Next Generation Network (NGN), Part III: Standards Gap Analysis. Report by Alliance for Telecommunications Industry Solutions (ATIS), May 2006.
[2] Network Performance Objectives for IP-based Services. ITU-T Recommendation Y.1540, May 2006.
[3] X. Zhang, L. Bhuyan, and W. Feng, “Anatomy of UDP and M-VIA for cluster communication,” J. Parallel and Distributed Computing, Special Issue on Design and Performance of Networks for Super-, Cluster-, and Grid-Computing, vol.65, pp.1290–1298, Oct. 2005.
[4] Framework for IP Performance Metrics (IPPM). RFC 2330, May 1998.
[5] Internet Protocol Data Communication Service – IP Packet Transfer and Availability Performance Parameters. ITU-T Recommendation Y.1540, May 2006.
[6] G. Cena, I. Bertolotti, and A. Valenzano, “Experimental analysis of latencies in ethernet communications,” Proc. Workshop on Factory Communication Systems, pp.303–312, June 2006.
[7] Endace DAG Cards Datasheet. Available at: http://www.endace.com
[8] P. Zejdl, S. Ubik, V. Macek, and A. Oslebo, “Traffic classification for portable application with hardware support,” Proc. Workshop on Intelligent Solutions in Embedded Systems (WASOC), pp.1–9, July 2008.
[9] P. Arlos and M. Fiedler, “Influence of the packet size on the one-way delay in 3G networks,” Proc. Passive and Active Measurement Conference, vol.6932, Springer, pp.61–70, April 2010.
[10] K. Wac, P. Arlos, M. Fiedler, S. Chevul, L. Isaksson, and R. Bults, “Accuracy evaluation of application-level performance measurements,” Proc. 3rd Next Generation Internet Networks (EuroNGI), pp.1–5, May 2007.
[11] Bigfoot Networks Killer NIC K1. Available at: http://www.bigfootnetworks.com/killer-nic-k1/
[12] Definition and Terminology for Synchronization Networks. ITU-T Recommendation G.810, Aug. 1996.
[13] S.B. Moon, P. Skelly, and D. Towsley, “Estimation and removal of clock skew from network delay measurements,” Proc. INFOCOM, vol.1, pp.227–234, March 1999.
[14] F. Bacelli, S. Machiraju, D. Veitch, and J. Bolot, “The role of PASTA in network measurement,” ACM SIGCOMM Computer Communication Review, vol.36, pp.231–244, Oct. 2008.
[15] V. Ricci, “Fitting distribution with R.” J. Economy and Commerce, vol.1/2, pp.47–60, 2005.
[16] WAND Group, The DAG cards. Available at: http://dag.cs.waikato.ac.nz
[17] P. Gilfeather and T. Underwood, “Fragmentation and high performance IP,” Proc. 15th International Parallel and Distributed Processing Symposium (IPDPS), pp.165–173, 2001.
Marat Zhanikeev received M.S. and Ph.D. in global information and telecommunications studies from Waseda University in Tokyo, Japan, in 2003 and 2007, respectively. He has co-authored close to a 100 peer-reviewed papers, as well as several invited papers and tutorials. His current research interests include network measurement, network monitoring, and network management, and can roughly be embraced by the area of network performance in general. End-to-end network performance is specifically his immediate target at the time of this writing. He is presently an assistant professor in Tokyo University of Science.

Yoshiaki Tanaka received the B.E., M.E., and D.E. degrees in electrical engineering from the University of Tokyo, Tokyo, Japan, in 1974, 1976, and 1979, respectively. He became a staff at Department of Electrical Engineering, the University of Tokyo, in 1979, and has been engaged in teaching and researching in the fields of telecommunication networks, switching systems, and network security. He was a guest professor at Department of Communication Systems, Lund Institute of Technology, Sweden, from 1986 to 1987. He was also a visiting researcher at Institute for Posts and Telecommunications Policy, from 1988 to 1991, and at Institute for Monetary and Economic Studies, Bank of Japan, from 1994 to 1996. He is presently a professor at Global Information and Telecommunication Institute, Waseda University, and a visiting professor at National Institute of Informatics. He received the IEEE Outstanding Student Award in 1977, the Niwa Memorial Prize in 1980, the IEICE Achievement Award in 1980, the Okawa Publication Prize in 1994, the TAF Telecom System Technology Award in 1995 and in 2006, the IEICE Information Network Research Award in 1996, in 2001, in 2004, and in 2006, the IEICE Communications Society Activity Testimonial in 1997 and in 1998, the IEICE Switching System Research Award in 2001, the IEICE Best Paper Award in 2005, the IEICE Network System Research Award in 2006, in 2008, and in 2011, the IEICE Communications Society Activity Award in 2008, the Commendation by Minister for Internal Affairs and Communications in 2009, and the APNOMS Best Paper Award in 2009.