IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

IMPROVE THE PERFORMANCE OF NETWORK BASED ON THE PACKET GENERATION

[1] Mr. N. Nanthini, [2] Mrs. P. Alaguthai, [3] T. Kanniyaadevi, [4]N. Poomalai@Tharani

[1, 2]Assistant Professor, [3, 4] Scholar

1, 2, 3, 4 Dept. of Computer Science, Sakthi College of Arts and Science for Women, Oddanchatram, TamilNadu, India.

ABSTRACT

Packet generators are crucial components in performance evaluations of network equipment, but can often be very expensive tools for companies in need of these to invest in. This creates a need for a more cost-effective solution to be able to scale the performance testing into the everyday life of the developers at companies developing these products. This report investigates this by looking at how open-source packet generator software can be used together with off-the-shelf hardware to decrease costs compared to buying commercial systems using purpose-built hardware. The state of the art of packet generation is reviewed and the most promising solutions are evaluated in more detail. An implementation of a prototype system for packet generation using open-source software and commodity hardware is also presented to show both challenges and possibilities with this approach. A conclusion is reached that while the commercial products using purpose-built hardware still have use cases when they are the preferred choice, solutions using open-source software together with off-the-shelf hardware can fill the need for scalable and cost-effective packet generation in other situations.

I. INTRODUCTION

The world is moving fast into the zetta byte era with the annual global IP traffic expected to surpass the zetta byte threshold in 2016 [4], meaning that more than 1021 bytes of data is predicted to be sent during the year. The growth is not predicted to slow down and will probably reach annual levels twice as high as this in only a few years. More and more people and devices are constantly being connected to the Internet and the amount of data each person generates is increasing every year. With the increased use of connected devices in the everyday life of many people, these devices have become essential parts of how we live. With this development, high demands are made by the users.

Everything is expected to work whenever and wherever people need them which puts a lot of pressure on the developers and manufacturers of the network equipment that should make it possible for people to use their devices in the way they want. With this ever increasing demand for performance and stability in network equipment, there is a need to be able to reliably verify the performance on a day-to-day basis in a scalable and cost-effective manner both during the development and when products are finished. For this, researchers and manufacturers of hardware and software network components rely on being able to generate traffic and send it into networks in a controlled manner.

This is done to perform quality of service testing and performance evaluations of the different components in a computer network to validate that the equipment is working as intended. High performance traffic generation has for a long time only been possible to achieve by using expensive commercial products with dedicated hardware and software that is made to work only on this specific hardware.

This is however starting to change as commodity hardware equipment is getting more powerful and new software techniques are being developed and improved. This report investigates if it is possible to use open-source software components and common off-the-shelf hardware to create a trustworthy packet generator with performance high enough to be comparable with purpose-built network test equipment. An overview of existing methods, tools and technologies for packet generation today is presented, including a closer evaluation of two of the existing systems.

A deeper look at how one of these systems works is also presented and the development of a prototype for a packet generator test tool is described together with discussions about a few challenges faced when developing this kind of system. An evaluation is then presented to assess the usability of a tool developed using these techniques, comparing it to purpose-built equipment.

IMPLEMENTATION GOALS

To be able to answer the research questions specified in the previous section, a prototype of a packet generator test tool were to be developed together with studying literature. A few implementation goals for this prototype were decided upon at the beginning of the project, specifying and prioritizing different parts that would be interesting to investigate. These implementation goals were the following: – Functionalities for the packet generator to investigate if and how they can be implemented, prioritized in a list where 1-6 were decided to be high-priority goals and 7-11 were secondary goals to investigate further if there was time left for it:

1. IPv4 UDP packet generation (uni- and bidirectional) at specified rate and packet size

2. Multiple client emulation

3. Script support

4. Measuring latency

5. IPv6 UDP packet generation

6. Generation of packets with VLAN headers

7. Generation of packets with different configurations simultaneously (multiple streams of packets)

8. TCP packet generation

9. Measuring setup rate

10. Replay of previously captured traffic

11. Packet generation using IPsec

– Construct controls for the execution of the packet generator – Gather statistics from test runs (real-time and total results after tests) – Command line client – Web user interface (low priority) – Ability to control multiple packet generators (low priority)

II. LITERATURE REVIEW

[1] STEFANO AVALLONE, DONATO EMMA, ANTONIO PESCAP`E, AND GIORGIO VENTRE. A DISTRIBUTED MULTIPLATFORM ARCHITECTURE FOR TRAFFIC GENERATION.

Since the early days of computing, the demand of large scale scientific applications on more powerful hardware platforms has been a driving force for the development of advanced software environments including dedicated, machine optimized libraries for scientific computing. As more and more hardware

platforms emerge, the adaption of scientific applications codes to each new computer architecture requires significant programming efforts. Some of these efforts can be alleviated by employing restructuring compilers to restructure existing application codes. However, restructuring compilers need relatively 'clean code' without many programming tricks that is often exploited by human programmers.

Otherwise, automatic restructuring of a code for different hardware architecture can be an unsolved problem for those applications developed with specific target architecture in mind; in such an implementation, part of the knowledge content of the model is lost. This knowledge may prove to be invaluable with respect to an efficient implementation of the model on different computer architecture. While code (rewriting by hand is common practice in the field of application development, it will be evident that the (semi-) automatic generation of code directly from a high level language description of a problem provides a fast, robust, and, therefore, attractive alternative.

In this respect, a problem-specific code generator, also known as application driver, should translate the problem into near-optimal code for a given target computer architecture. In this paper the CTADEL Code-generation Tool for Applications based on Differential Equations using high-level Language specifications will be presented. Currently, a prototype system has been developed. This prototype system adopts explicit finite difference methods as numerical solution method and, for parallel architectures, adopts a domain splitting method to decompose the problem into sub problems that can be solved independently on each processor.

Furthermore, CTADEL incorporates a global common sub expression eliminator that acts in concert with an algebraic simplifier to reduce the computational complexity of the problem on a global scale. These techniques provide the necessary means within the CTADEL system to generate high-performance codes for various computer architectures automatically from a high-level language description of a model. We used the prototype CTADEL system as an application driver for the HIRLAM system, an operational limited area numerical weather forecast system [16]. We will present recent results of this prototype system for generating codes for the HIRLAM system.

Related Process

Many attempts have been made to solve scientific or physical models numerically or symbolically using computers. Today, a large collection of libraries, tools, and Problem Solving Environments (PSES) have been developed for solving such problems. The current and future role of these environments is discussed in [12]. Well-known PSES for solving PDEs, see [II] for an extensive overview, are ELLPACK [19] and DEQSOL [22], and in the field of parallel computing //ELLPACK [15].

The computational kernels of these PSES consist of a large library containing routines for many numerical solution met hods. These rout lines form the templates for the resulting code. The numerical knowledge a

different approach to library-based PSES consists of a collection of tools that generate code based on a problem specification without the use of a library. The numerical l&owledge is determined by the expressiveness of the problem specification language and the underlying translation techniques; the system employs a set of refinement steps to the problem by which the algorithms and data structures are refined from an abstract level into a concrete implementation.

Examples are the ALPAL system [7], the SINAPSE system [17], the SUSPENSE transformation system [20], and the DPML Data Parallel scientific Modelling Language [10]. The CTADEL system should also be considered as a system of the last type. However, a novel approach is taken in which the description of the problem is transformed into code by taking target computer architecture characteristics into account. This way, the main objective of the CTADEL system, the generation of efficient codes, can be achieved.

The advantage of this approach is that by exploiting specific hardware characteristics of the target computer architecture there is a complete absence of portability and codeconsistency related problems. For each machine, an efficient hardware-specific version of the code can be generated. All versions of the code will be consistent with each other and can incorporate the latest improvements to the model or problem specification.

[2] JOHN W. LOCKWOOD; NICK MCKEOWN; GREG WATSON; GLEN GIBB; PAUL HARTKE; JAD NAOUS; RAMANAN "NETFPGA--AN OPEN PLATFORM FOR GIGABIT-RATE NETWORK SWITCHING AND ROUTING"

The NetFPGA platform enables students and researchers to build high-performance networking systems in hardware. A new version of the NetFPGA platform has been developed and is available for use by the academic community. The NetFPGA 2.1 platform now has interfaces that can be parameterized, therefore enabling development of modular hardware designs with varied word sizes. It also includes more logic and faster memory than the previous platform. Field Programmable Gate Array (FPGA) logic is used to implement the core data processing functions while software running on embedded cores within the FPGA and/or programs running on an attached host computer implements only control functions. Reference designs and component libraries have been developed for the CS344 course at Stanford University. Open-source Verilog code is available for download from the project website.

The NetFPGA Packet Generator operates similar to that of the software program, tcpreplay. We ran two experiments with the tcpreplay and the Packet Generator. We configured a computer system that contained a AMD dual core processor running at 2.5 GHz and an Intel dual port e1000 PCI-express x4 NIC. We then used a PCAP file that contained 43 packets and 25383 bytes. Using tcpreplay we tested the average rate that the system could achieve when playing this file on one port and also on two ports simultaneously. The

Packet Generator was then used to play the same PCAP file with both one port and two ports simultaneously. Each experimental setup (i.e. one port, and two ports) were run ten times and the

Being able to reliably repeat an experiment is a key feature of the NetFPGA packet generator. When using software programs running on a PC, it is difficult to ensure that inter-packet delays are always the same between experiments. Figure 3 shows the variation of packet arrival time of tcpreplay as compared to the NetFPGA Packet Generator.

When tcpreplay is run in software, most packets obtain a -2 to -18 microsecond delay from when there were expected to arrive (left of zero). A few dozen packets were sent +2 to +48 microsecond earlier than expected (right of zero), presumably due to variations of time caused by the kernel and/or the implementation of time delays by the TCP Replay software application. Only a few packets were are within plus or minus two microseconds of the expected arrival time. These variations in packet arrival time limits the precision experimenters can achieve. Using the NetFPGA, we can ensure that the delay between packets and the rate of sending is the same every time an experiment is run.

III SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Packet generators are crucial components in performance testing and validation of network hardware and software. With an ever increasing demand for performance in network equipment there is a need to be able to reliably verify the performance on a day-to-day basis in a scalable and cost-effective manner. Recent developments in software technologies and multi-core hardware architectures allow for development of tools and methods to perform these types of tests in a much more cost-efficient manner than ever before. This chapter briefly describes the purpose of this project and specifies the different questions that are to be answered in this report. Besides this, a short presentation of related work is also given, together with a look at how the project was planned and executed.

3.2 PROPOSED SYSTEM

The main issue that this report covers is to investigate how common off-the-shelf hardware and open-source software components can be used to build a cost-effective high-performance packet generator to be used for performance evaluation of network equipment. The objective is to investigate how to achieve highly efficient and trustworthy packet generation and to evaluate methods for performance testing and validation as well as implementing software using these methods. The selected methods will need to support packet generation of custom configurable packets at configured rates, as well as providing key indicators such as, for example, the number of sent and received packets during a test run and latencies for the sent packets. Two different types of goals were specified at the beginning of the project, research questions and

implementation goals. The research questions were chosen to specify the kind of knowledge that this project tries to gain and the areas that are to be investigated. The implementation goals acted as guidelines to lead the work and to prioritize the different areas that were interesting to investigate further.

IV SYSTEM ARCHITECTURE

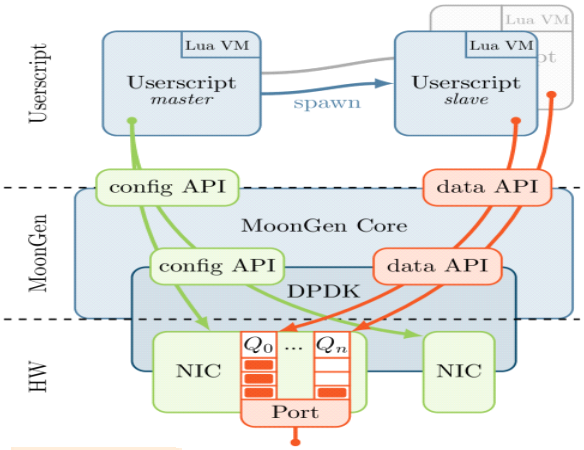

Figure 4.1 Architecture

V SYSTEM IMPLEMENTATION

5.1 SYSTEM METHODOLOGY

As for hardware, Intel 82599 10 GbE [15] network interface cards have been used. These were chosen because they are very common and fairly modern with different newer functionalities available, such as hardware filters, configurable multi-queue sending and receiving of packets, and hardware support for different protocols and calculations. Clavister was also using these network interface cards in many of their products, which meant that they were easily available for experimentation during the development of the prototype.

This chapter presents the development of this prototype. The design is first presented together with a short description about how it works. After that follows a few discussions about different challenges faced during the development of the packet generator test tool including various solutions to these.

Packets will be mixed during transmission when packets from multiple streams are sent simultaneously. When receiving packets at the destination, the stream that these packets belong to has to be identified to be able to present useful statistics and other results from the test run. A scenario where this is important is, for example, when evaluating traffic shaping. In this situation, a device under test will modify the flow of packets differently depending on the packets' configuration, i.e. which stream it belongs to. It is then important to be able to count the number of sent and received packets from each stream to be able to calculate loss percentages and similar statistics.

Modern Intel NICs have a feature called Flow Director filters [14], which can be used to sort packets into different receive queues based on a flexible 2-byte tuple anywhere in the first 64 bytes of the packet. Packets can be put into different queues and hardware statistics counters can be used to count the number of received packets at each queue. On the Intel 82599 NIC, there are however only 16 registers available for queue statistics, which means that the number of different streams is limited to a maximum of 16 on this NIC when hardware statistic counters are used. The filters at the NIC have also been seen to be a bit unreliable in some situations when used together with Intel DPDK. This, and the limited amount of possible streams, is two limitations with this approach, but it works well in most common situations.

5.2. EXPERIMENTATIONS & RESULTS

To evaluate the usability of a packet generator constructed using common off-the-shelf hard- ware and open-source software components, this chapter contains a comparison of the per- formance and functionality of the implemented prototype and commercial purpose-built packet generators. An important characteristic that will be evaluated is how cost-effective an open-source based solution using commodity hardware can be compared to the commer- cial systems, to see whether the extra functionality provided by the commercial products are worth their costs compared to other solutions.

Ixia1, Spirent2 and Xena3 are a few examples of companies providing purpose-built packet generators. All of them have packet generator solutions using interfaces ranging from 10 to 100 GbE. 40 GbE NICs are currently starting to be introduced to the market for commodity hardware, while 100 GbE NICs are not yet available [8]. This means that a packet generator using commodity hardware and open-source software currently has to use 10 GbE NICs, which means that the possible maximum throughput per interface will be a lot lower on these systems than on the commercial products. Fortunately, multiple 10 GbE interfaces can be combined to together reach the same amount of generated traffic as the interfaces with higher capacity.

It has been shown in this project that eight 10 GbE interfaces can be used to generate traffic at 80 Gb/s and Emmerich et al. have tested this for up to 120 Gb/s [8]. No indication has been seen that it would not be possible to scale this to even higher throughputs by combining even more 10 GbE interfaces, as long as the other requirements for being able to produce the packets are fulfilled, such as that the needed CPU capacity is available, etc.

The main drawback with the open-source solutions regarding the throughput that is possible to achieve is that the generated traffic might be divided among more interfaces instead of being focused at only a few as it could be on purpose-built systems. The available commodity hardware is constantly improving though, and 40 and 100 GbE interfaces will probably be more common on the market in the coming years. The technique used when sending the generated packets from multiple NICs at the same time is architecturally the same as if the packets would be sent to multiple queues at a single NIC instead. This would indicate that

the same technique used today in open-source systems using multiple 10 GbE interfaces could be possible to use when sending packets from 40 or 100 GbE interfaces when they are available in the future.

VI CONCLUSION

The following three questions have been investigated throughout this project:

- How can highly efficient and trustworthy packet generation be achieved using common off-the-shelf hardware and open-source software components?
- How can the identified methods for packet generation be used to perform reliable performance testing of network equipment?
- How cost-effective are the identified methods for performance testing in comparison with purpose-built network test equipment?

To answer these questions, the state of the art of packet generation was reviewed and avail- able methods and techniques were evaluated and discussed. Different problems and limita- tions with software based packet generation compared to packet generators using purpose- built hardware were also identified to get a better understanding of what is possible to achieve using open-source software and off-the-shelf hardware.

The most promising methods for packet generation were evaluated in more detail and a prototype system for packet generation for performance testing of network equipment was implemented. This was done to show how it can be done and to find limitations and challenges with this approach. From the implementation goals specified in Chapter 3, how to implement functionalities 1-7 were investigated while 8-11 had to be left due to lack of time.

These last four functionalities are, however, probably also possible to achieve using similar methods as have been done for the first seven, but this is something that has to be investigated more in the future. The implemented prototype, together with the review of packet generation and the evaluation of the two selected systems, have shown that it is possible to achieve high-performance packet generation in a cost-efficient manner using open-source software and off-the-shelf hardware components.

There are, however, limitations to this approach, which are discussed in more detail in Chapter 6 and Chapter 7. The commercial packet generators using purpose-built hardware have their advantages and are still the preferred choice in certain situations, such as when complex traffic is wanted or when guarantees about the results are required. In other situations, such as when sending simpler traffic or when doing performance tests during the development of a product to get an indication of the performance, open-source solutions can be very useful.

The two low-priority goals specified in Chapter 3 have not been investigated in further detail during this project. The results have been visualized in the command line interface for the developed prototype and the testing can be automated by using the REST API, but these areas still require further work to find the best solutions to these two questions.

As have been shown in this report, the methods for performance testing using open-source software and off-the-shelf hardware components provide very cost-effective alternatives com- pared to commercial products in the cases when they can be used. They give the ability to scale the performance testing to levels not possible with commercial products and give the possibility to perform day-to-day testing during the development of network equipment without relying on expensive test tools being available. The commercial products still have important roles during the development of network equipment, for example for formal per- formance evaluations and more complex testing, but the software based packet generator solutions can fill the need for scalable performance testing in more basic situations.

VII FUTURE ENHANCEMENT

This report has shown that cost-efficient performance testing is possible using off-the-shelf hardware and open-source software components, but a lot more work can still be done in this area. The prototype that has been developed during this project can be developed further to make it a more complete tool for performance testing. This can be done by completing the low-priority implementation goals presented in Chapter 2, followed by continued testing and implementation of other needed functionalities. The generated traffic also has to be evaluated in more detail to make sure that the results from using the prototype really can be trusted and that the packet generator is delivering what it says.

How to generate packets using more complex protocols is another area that more work can be done in. Sending packets using stateful protocols, such as stateful application level protocols or TCP, requires more from the packet generator than what, for example, sending UDP packets does. These stateful protocols would be very useful to be able to generate when doing performance evaluations of network equipment and are therefore interesting areas to investigate further how they can be achieved.

The development of new hardware and software for fast packet processing is also a vital part in increasing the usefulness of packet generators using commodity hardware and software. As new techniques become available, how to best utilize these will be important questions to answer to further improve the performance of these systems and to make the open-source based packet generators running on off-the-shelf hardware into even more useful tools for performance evaluation of network equipment.

REFERENCES

1.Stefano Avallone, Donato Emma, Antonio Pescap`e, and Giorgio Ventre. A distributed multiplatform architecture for traffic generation. In International Symposium on Per- formance Evaluation of Computer and Telecommunication Systems, pages 659–670, 2004.

2.Alessio Botta, Alberto Dainotti, and Antonio Pescap´e. Do you trust your software- based traffic generator? IEEE Communications Magazine, 48(9):158–165, 2010.

3.Scott Bradner and Jim McQuaid. Benchmarking Methodology for Network Interconnect Devices. RFC 2544 (Informational), 1999.

4.Cisco VNI. Cisco visual networking index: Forecast and methodology, 2015-2020. CISCO White paper, 2016.