An Application-Oriented Synthetic Network Traffic Generator

Carsten Albrecht, Christoph Osterloh, Thilo Pionteck, Roman Koch, and Erik Maehle
Institute of Computer Engineering
University of Lübeck
Ratzeburger Allee 160
23538 Lübeck
Germany
{albrecht,osterloh,pionteck,koch,maehle}@iti.uni-luebeck.de

Abstract— **Design space exploration and detailed analysis in the field of hardware design applies simulation in many variants. A frequently used method is stochastic simulation where systems are stimulated by randomised input. Synthetic traffic traces mainly form the load for stochastic simulation of network computing devices. The generator presented here utilises two well-known models to meet the features of a majority of applications and traffic sources. Based on application-specific parameter sets, the traffic models stochastically generate packet flows which are merged to an aggregated stream. Nevertheless, all packets can always be identified and are not resolved to a data mass representing the load of a link.**

Keywords— **Traffic Generator, Packet Flow Models, Synthetic Workload Generation**

I. Introduction

Growing bandwidth and complexity of network applications and protocols, as well as issues of network security and integrity cause more and more complex packet processing tasks. They have to be performed particularly at the edges of sub-networks. The resulting need for increasingly powerful and flexible processing devices is, however, hard to satisfy, because the general increase in processing power is exhausted by the increase in utilised bandwidth. In order to exploit the evolving bandwidth potential, it is therefore inevitable to investigate architectures tailored to specific application areas. Simulation of hardware designs to examine their potential concerning performance and power is a suitable and wide-spread technique for design-space exploration and evaluation of the applied algorithms.

Simulation results highly depend on the quality of the stimulation. In the field of network computing, packet traces forming the data to be processed within a device are needed as stimulations. With regard to the scope of application the examined device has to be tested and investigated by realistic but also challenging stimuli to show behaviour and performance characteristics in heavy-load scenarios.

Since a decade, hardware devices such as FPGAs (Field Programmable Gate Arrays) provide more flexibility. While they were just reprogrammable case-by-case they are now even partially reprogrammable while they execute a certain task. These run-time reconfigurable systems are able to adapt their system behaviour and to outperfom traditional systems [Compton and Hauck, 2002]. Applied to network computing as a discipline of hardware design new features have to be tested and evaluated. That is why usual benchmark suites and packet generators are not sufficient anymore to stress the devices for any purpose. Beside usual requirements on data rates, and protocol stacks, the adaptive features of run-time reconfigurable network devices need certain conditions that exercise and test their capabilities. In particular, the traffic traces demanded here have to be realistic and have to include changes over long and short terms. Additionally, it is necessary to distinguish traffic requirements on a rather fine-granular per-flow basis to adapt the system to it. A sample system that requires these features of a traffic generator tool for analysis is described in [Albrecht et al., 2006], [Pionteck et al., 2006].

The traffic generator presented in the following uses well-known traffic models with application-specific parameter sets for generating flows. The term *flow* is defined here as 5-tuple of used protocol within the Internet Protocol (IP) packet, source address, destination address, source port, and destination port. A fixed time-out value T_0 describes the longest possible period between two packets of the same flow [Hohn et al., 2005]. The ports are given by the application originating the traffic.

The paper is organised as follows: First, the basic packet models are introduced in Section II and then are mapped to applications in Section III. Based on that, the concept and user interface of the synthetic traffic generator are described in Section IV. Section V shows generated traffic and sketches its application in the system-design phase. Before Section VII finalises this work and gives final remarks some existing and frequently used packet generators are described and compared in Section VI.

II. Packet Flow Models

Network traffic is still frequently modelled using the Poisson model although many examinations have shown that the packet inter-arrival time is in general not exponentially distributed [Paxson and Floyd, 1995]. Consequently, there are numerous models for different kinds of network traffic. They have at least two

Proceedings 22nd European Conference on Modelling and Simulation ©ECMS Loucas S. Louca, Yiorgos Chrysanthou, Zuzana Oplatková, Khalid Al-Begain (Editors)
ISBN: 978-0-9553018-5-8 / ISBN: 978-0-9553018-6-5 (CD)

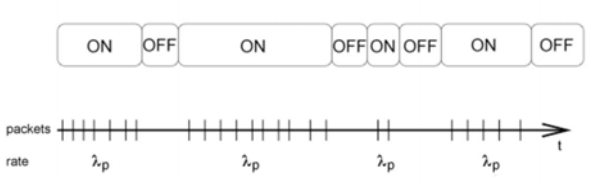


Fig. 1. Flow generated by On/Off model.

common features: self-similarity and long-range dependency. Self-similarity in the area of network traffic is the quality of keeping similar structures independent of scale and section. The requirement for self-similarity is shown in [Leland et al., 1994]. Long-range dependency is the property that there is a high dependency within long time intervals [Babic et al., 1998] and, thus, have high auto-correlation values over long periods of time. Further on, as a third property of the generated traffic which is closely related to long-range dependency, traffic should be heavy tailed. This leads to the Pareto principle where a huge number of small values add less to the average than a few large values. This effect is also known as $80 - 20$ law. The Pareto distribution is the most famous heavy-tailed distribution. Beside others the following two models provide the necessary properties and are able to cover a large number of applications generating network traffic.

A. On/Off *Model*

The On/Off model, also called packet-train model [Jain and Routhier, 1986], generates self-similar and long-range dependent traffic. The model bases on a finite-state machine with two states, On and Off. Packets are generated in bursts during the On phases. On phases last a time that follows a general distribution whereas the durations of Off phases are exponentially distributed.

Figure 1 shows bursts generated by an On/Off model. The first row shows an alternating sequence of states, the second one the time of packet generation. The packets are basically transferred with the infinite transfer rate λ_p [Siriwong et al., 2007]. The parameters concerning number and size of packets as well as the distribution of On and Off phases depend on the application generating the traffic.

An extension to this single stream model is the N-burst model [Lipsky et al., 2001]. Here, multiple sources produce packets which are loss-less buffered in a router. The router is modelled as a service station with a Markov-modulated Poisson process.

B. *b-Model*

Another well suitable model for generating traffic with the properties described above is the b-model [Wang et. al., 2002]. In contrast to the On/Off model, the b-model can be applied to generate aggregated traffic streams such as background traffic for network studies, e.g. [Thid et al., 2006]. Furthermore, single sources such as file accesses can also be modelled.

The algorithm is a divide-and-conquer mechanism

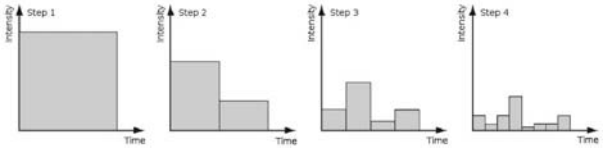


Fig. 2. Flow generation using b-model.

that splits the amount D of data into two parts. b acts as a weight so that one part gets $b \times D$ and the other $(1 - b) \times D$. The splitting is repeated until the required granularity is achieved. Figure 2 depicts the process of generating a b-modelled stream.

III. Applications

Network traffic is generated by various applications. Figure 3 shows the distribution of the applications which are responsible for most of the traffic. The distribution of applications is given in [Luo and Marin, 2005]. http, ftp, ssh, pop3 and smtp form more than half of the TCP (Transmission Control Protocol) traffic. Combining this statistic with daily network statistics of e.g. `www.caida.org`, the set of applications and protocols has to be extended by IPsec which is here included in the 'others' section. Nowadays, IPsec achieves a top rank in the statistics. Many institutions and companies provide secure access to their infrastructure by applying so-called IPsec tunnels. These tunnels contain again a lot of the previously named applications.

For modelling a significant part of usual network traffic all these applications have to be mapped to models utilisable for generating synthetic traffic streams. Although the degree and frequency of user interaction and so their parameter sets and generated traffic profile differ almost all those applications can basically be mapped to a single model: the On/Off model. For generating IPsec traffic the b-model is chosen because it is also designed for aggregated traffic. Of course it would be possible to mix tunnel traffic from a set of sources, too, but in this case the b-model reduces the computing effort.

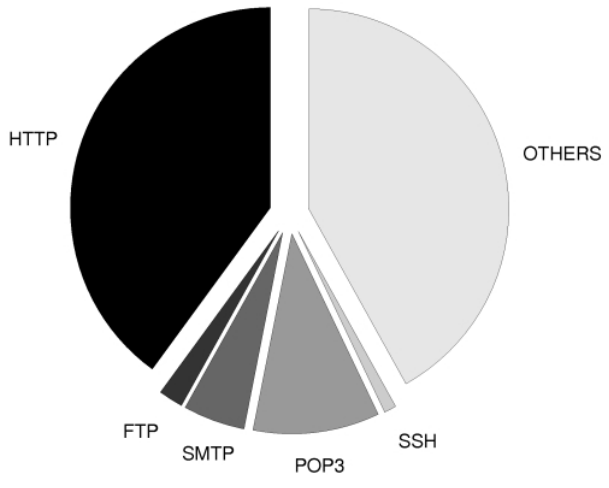


Fig. 3. Application-level traffic-source distribution of the TCP protocol.

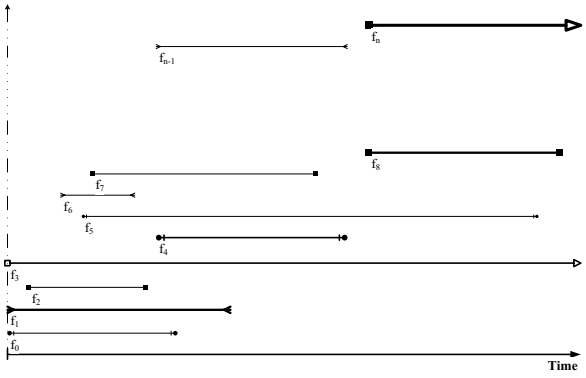


Fig. 4. Typed flow live-cycle diagram.

The parameter sets for the On/Off model to generate application-specific traffic can be taken from [Luo and Marin, 2005].

IV. Synthetic Traffic Generator

Based on the models and applications of Sections II and III a synthetic traffic generator is built. Its aim is to produce an aggregated traffic stream composed of multiple sources with the opportunity to identify any packet origin. So, the result is more than only a bunch of data that has to be put on a link in a certain time period.

A. Concept

Figure 4 shows the principle of the generator. For the creation of a traffic stream multiple flows f_i can be defined. Each flow f_i has to be parametrised by a tuple $(A_{f_i}, B_{f_i}^{max}, D_{f_i}, t_{f_i})$ the elements of which are listed in Table I.

TABLE I: Flow parameters and their description.

A_{f_i}	- application or type of flow
$B_{f_i}^{max}$	- link limit or maximum bandwidth
D_{f_i}	- amount of data
t_{f_i}	- starting point, i.e. offset to t_0

The application sets up the model. The link limit or maximum bandwidth of the flow represents the history of the flow. If it has used links with small capacities such as DSL or WLAN its bandwidth is generally reduced. Further on, the total amount of data transferred within this flow is defined. D combined with B_{max} delivers the lower boundary of flow live time. Finally, each flow has its own starting point t_{f_i} as an offset to the origin of time t_0.

The path from application level to packet level in the flow generation requires a packet size distribution. Some examinations of applications provide histograms for packet size distributions. For general TCP/IP traffic the major peaks of packet sizes are frequently at 40 Bytes, the smallest possible value, at about 576 Bytes and at 1500 Bytes. 1500 Bytes is a usual value of the maximum transfer unit (MTU) of an TCP/IP scenario [Sprint, 2008], [CAIDA, 2008]. Gaps can be closed by linearly interpolated data. In case that an application lacks a packet size distribution a uniform distribution for packet sizes is assumed.

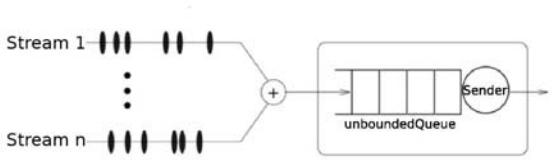
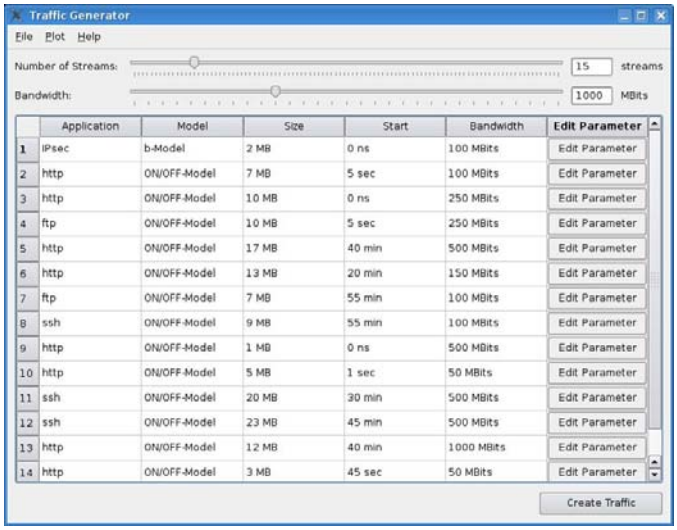


Fig. 5. Aggregation scheme for multiple flow sources.

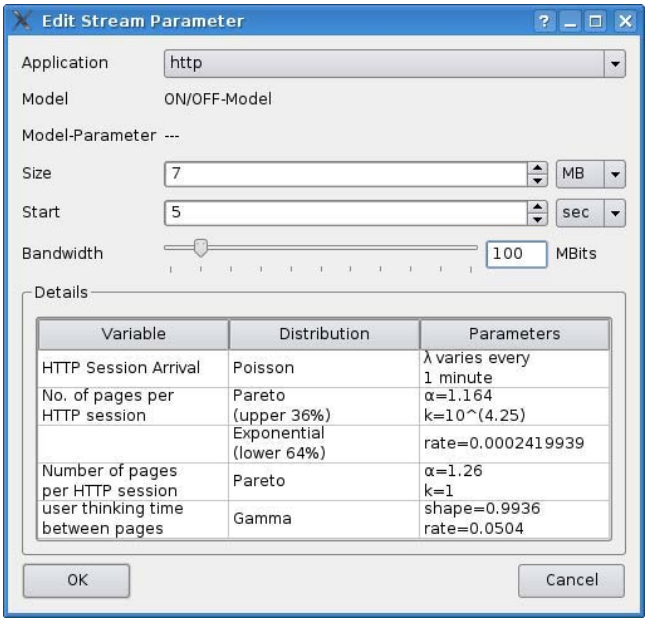
Another important issue is the combination of multiple flows to form an aggregated stream. [Hohn et al., 2005] uses an infinite queue for buffering and merging multiple streams. The procedure for merging used here is simply a merge sort. All packets get a global timestamp with respect to the stream offset t_0. It defines when they have to be scheduled to the buffer. So, the infinite buffer finally contains all packets in ascending order of their global timestamps. Figure 5 shows the assembly process. n streams are put into the buffer on the entrance site. Inside, the resulting aggregated stream is produced by emptying the queue over a defined link with a given bandwidth. Packet timestamps and global clock are synchronised so that packets do not leave the queue before their time is reached. This may lead to intermissions but also to congestion, i.e. packets leave the queue later than their timestamps announce. This procedure is realistic and simulates the processes that are performed on any system injecting data into a system. Large data bursts have always to be buffered and sent step-by-step over the target link. Congestion in real world scenarios could lead to packet loss. Since the packet flows are generated based on parameters derived from traffic monitored within a real network, congestion and packet loss is basically taken into account. Therefore, the ideal and loss-less assumptions have to be chosen. Moreover, this technique represents the processing procedure of routers and switches where packets are labelled with a unique and ascending identifier on the ingress side which comprises multiple sources or interfaces, respectively. On the egress side the packets are transmitted in the order of their identifiers so that the packet order is kept unchanged. Quality of service (QoS) features may break this order by classifying the traffic in at least two classes with different policies and bandwidth guarantees. Here, only the order of packets within the same class is kept. Packets of different classes can be rearranged to achieve the QoS requirements.

B. Software Tool

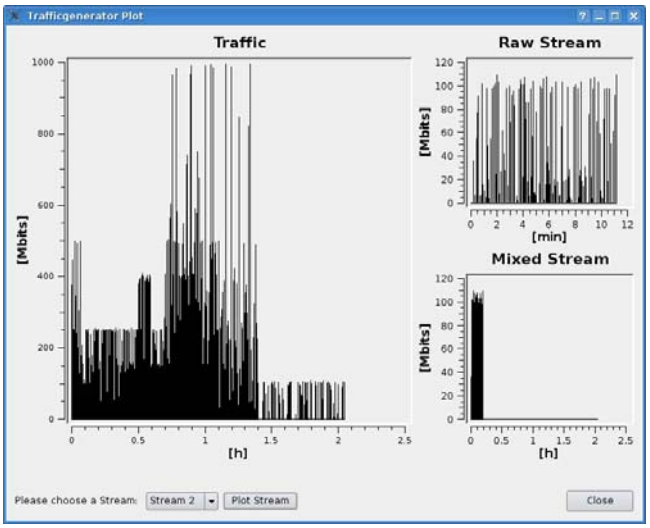
Based on this concept a tool is realised to generate aggregated or single-source traffic streams. The back-end is formed by the implementation of the models presented in Section II. For the implementation of these stochastic processes a pseudo-random number generator is used for making decisions stochastically. The implementation applies the Newran03 library [Davies, 2006] which is used in other established



(a)Main window provides overview of generated flows.



(b)Sub-menu for flow parametrisation including help on flow parameters and standard values.



(c)Plotting aggregated and individual flows before and after merge process.

Fig. 6. Views of Traffic Generator.

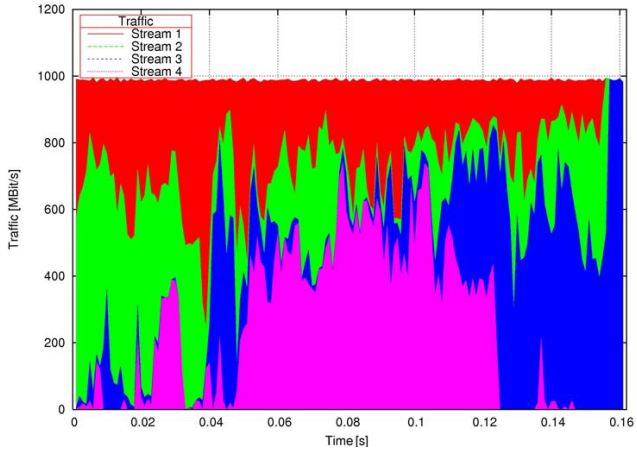
simulation tools, too.

The front end of the traffic generator is a graphical user interface to ease configuration and parametrisation of the generator system and its sub-systems. The graphical user interface utilises the Qt library.

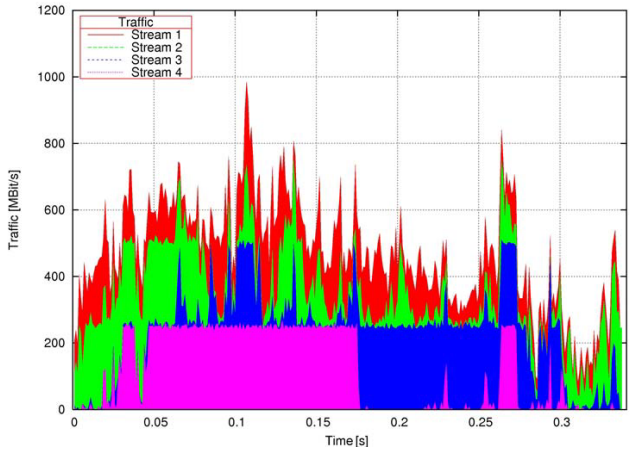
Figure 6 shows different user views on the trace configuration. In the root or main window shown in Figure 6(a) the tuple $(A_{f_i}, B_{f_i}^{max}, D_{f_i}, t_{f_i})$, see Table I, is defined. The first argument A_{f_i} includes a completely parametrised model so that it is not necessary but feasible to adapt the model parameters for own purposes. Model parameters are presented in a sub-menu with an own window shown in Figure 6(b). For user guidance the different parameters are described and the default values are given. The complete traffic configuration can be saved in a configuration file for later re-use. Because of the stochastic processes, different traces can be generated with the same settings. This is of course an important feature for the evaluation of stochastically driven simulations.

After generating a trace based on the given configuration the data distribution is visualised as shown in Figure 6(c). The large diagram on the left side of Figure 6(c) shows the cumulated data distribution of the aggregated traffic trace. For further analysis single streams can be depicted with more details on the right side. A stream is selected by the combobox [Trolltech AS, 2007] at the bottom. The diagram in the upper right corner shows the originally generated data distribution of the selected stream whereas in the lower right corner the embedded and adapted distribution is shown.

The output is given as a file in a currently proprietary format providing timestamp, application identifier, flow



(a) Aggregation of four flows consuming complete link capacity each.



(b) Aggregation of four flows with different bandwidth limits.

Fig. 7. Synthetic b-modeled traffic samples with and without bandwidth limits for certain flows.

identifier to re-assign any packet to its originating flow description, see Figure 6, and packet size. For an improved integration into other environments the output capabilities are extended to the `pcap` format and an XML scheme.

The configuration shown in Figure 6 is utilised in the following to create the aggregated stream of Figure 8.

V. Application Example

A sample system that requires specific traffic to examine all functionalities of the device is a run-time reconfigurable coprocessor for network processors utilised for, e.g., edge routers. It provides the opportunity to off-load time-consuming payload processing so that the network processor gains headroom to serve its tasks more quickly. The coprocessor [Albrecht et al., 2006] adapts its system configuration to the flow requirements. An examination of self-adaptation, for example, is given in [Albrecht et al., 2008].

Here, the coprocessor supports tunnel modes for IPsec partially combined with IPcomp by providing accelerators for the cryptographic algorithms AES, DES, and 3DES as well as the compression algorithm Deflate. Tunnels are aggregated flows just as the complete flow so that all streams of Figure 7(a) are generated by the *b*-model. Assuming that each stream is assigned to at least one of these algorithms, pipelining more than one is possible and required. It is easy to see that the requirements of the aggregated flow change over time. Just a short traffic trace is shown, but the significant difference is obvious.

Another feature of the generator is shown in Figure 7(b). Basically, these streams are the same as in Figure 7(a) but the bandwidth limit is reduced to a quarter of the maximum which is 1000 Mbit/s in this case. Peaks need now longer to be transferred so that the overall duration grows.

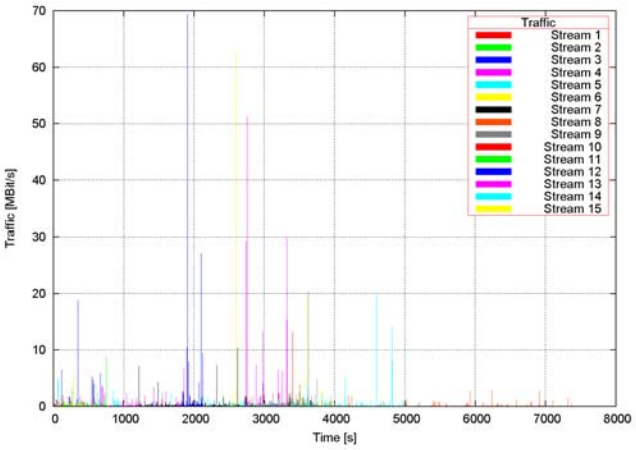
As mentioned before, traffic profiles change from daytime to daytime in composition and intensity. In Figure 8 a synthetic trace composed by 15 streams with different sources is shown. It is similar to daily profile with its increasing shape in the first half and a decreasing one in the second. All plots of Figure 8 show the same traffic using different time resolutions. This sequence of figures shows very well two important features of real Internet traffic which have to be met by a generator: self-similarity and burstiness. Starting with a time unit of $1s$ in Figure 8(a), time resolution is scaled up by a factor of 100 in Figure 8(b), and again by a factor of 10 in Figure 8(c). The aggregated traffic stream closely resembles itself in all three scales. Although the stream is depicted with a total scaling factor of 1000 averaging effects on the bandwidth in larger intervals are not visible. The shape including bursts and intermissions is kept over all time scales, i.e. its self-similar and bursty nature becomes obvious.

VI. Related Work

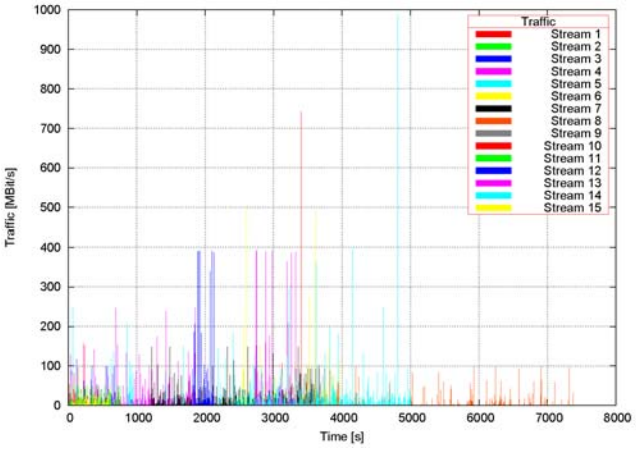
Synthetic traffic or workload generators are used in almost all fields of system and performance evaluation. So, the traffic generator presented here is of course not the first one. Aiml, granularity and, thus, applicability of the existing approaches, however, strongly differ.

[Vishwanath and Vahdat, 2006] presents the traffic generator Swing. It relies on a multi-layered model that represents user input or user interaction with the application, sessions of a single application, connections within a session, for example connections to different target hosts, and, finally, the network characteristics. Swing first monitors a packet link and evaluates the monitored data by applying this multi-layered model. Then, it starts to produce synthetic traffic based on the extracted information. The results meet well the real traffic characteristics. Nevertheless, the generator depends on an existing real environment or at least a test bed where data can be monitored.

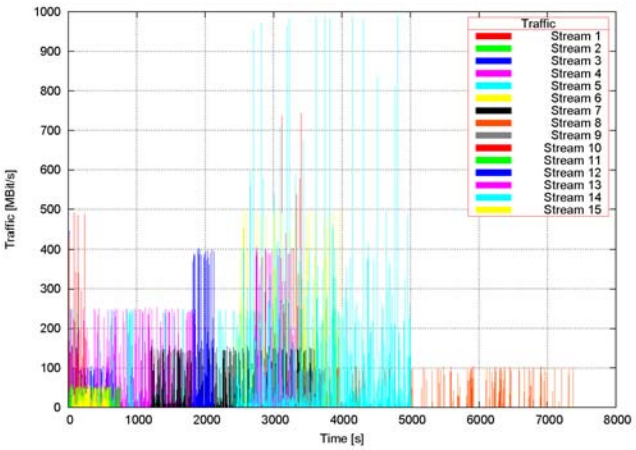
In contrast to the previous generator, Tmix [Weigle et. al., 2006] is targeted to the network simulator *ns-2*. Again the parameters for generating traffic are extracted from a monitored traffic trace that is recorded on a representative link. The model reduces application behaviour to connection behaviour, i.e. sub-parts of an application as described before. These connections are modelled by so-called epochs which is a triplet that de-



(a)Aggregated traffic with a time resolution of 1s.



(b)Aggregated traffic with a time resolution of 10μs.



(c)Aggregated traffic with a time resolution of 1μs.

Fig. 8. Traffic plot with different time resolutions.

scribes the amount of data in both directions and the pause until the next epoch starts. This model resembles the On/Off model chosen here. Applications that do not match have not a representative here.

Further on, there exist approaches that only generate traffic of a single-source application such as HTTP [Barford and Crovella, 1998] or FTP [Ishac, 2001]. All these approaches are able to reproduce traffic as a single stream with the characteristics of their input parameters. Unfortunately these approaches are limited in the sense of generating traffic with changes in the profile, e.g. daily data distributions.

VII. Conclusion

Simulation is one of the leading techniques to evaluate new hardware designs. Examinations highly depend on the quality of stimuli driving the simulation model. Therefore, traffic generators providing traffic streams as realistic as possible are needed for synthetic workload generation in the area of network computing. The generator presented here is based only on two models which are well examined. The tool allows the generation of packet traces as coarse-grained volumes of data as well as a combination of fine-grained flows with varying sources and parameters. The transition between these two characteristics is smooth. This variant of traffic generation is important to assemble link load providing characteristics to analyse the features of the device-under-test. In particular, the recently arising field of run-time reconfigurable devices demands workload patterns that stress their adaptivity. So, beside the available functionalities it is important to build traces with self-defined characteristics and combinations of sources. The traffic generator presented here is a suitable solution to close this gap.

VIII. Acknowledgement

This work was funded in part by the German Research Foundation (DFG) within priority programme 1148 under grant reference Ma 1412/5.

References

[Albrecht et al., 2006] Albrecht C., Foag J., Koch R., and Maehle E. 2006. *DynaCORE - A Dynamically Reconfigurable Coprocessor Architecture for Network Processors.* In Proceedings of the 14th Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP). IEEE Computer Society. Montbeliard-Sochaux, France.

[Albrecht et al., 2008] Albrecht C., Roß P., Koch R., Pionteck T., and Maehle E. 2008. *Performance Analysis of Bus-Based Interconnects for a Run-Time Reconfigurable Co-Processor Platform.* In Proceedings of the 16th Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP). IEEE Computer Society. Toulouse, France.

[Babic et al., 1998] Babic G., Vandalore B., and Jain, R. 1998. *Analysis and Modeling of Traffic in Modern Data Communication Networks.* In The Computing Research Repository, cs.NI/9809056, 1998.

[Barford and Crovella, 1998] Barford P., and Crovella M. 1998. *Generating representative Web workloads for network and server performance evaluation.* In ACM SIGMETRICS Performance Evaluation Review, Vol. 26, Issue 1, 151-160.

[CAIDA, 2008] Cooperative Association for Internet Data Analysis (CAIDA). 2008. `www.caida.org`.

[Compton and Hauck, 2002] Compton K., Hauck S. 2002. *Reconfigurable Computing: A Survey of Systems and Software.*

In ACM Computing Surveys, Vol. 34, No. 2, ACM Press, New York, NY, USA.

[Davies, 2006] Davies R. 2006. *Newran03 - a random number generator library* `http://www.robertnz.net`

[Hohn et al., 2005] Hohn N., Veitch D., and Ye T. 2005. *Splitting and Merging of Packet Traffic: Measurement and Modelling.* Performance Evaluation, Vol. 62, Issues 1-4, 164-177, 2005.

[Ishac, 2001] Ishac J. *FTP Traffic Generator.* Technical Report, Case Western Reserve University.

[Jain and Routhier, 1986] Jain R., and Routhier, S. A. 1986. *Packet Trains - Measurement and a New Model for Computer Network Traffic.* In IEEE Journal on Selected Areas in Communications, Vol. 4, No. 6, 986-995, 1986.

[Leland et al., 1994] Leland W. E., Taqqu M. S., Willinger W., and Wilson D. V. 1994. *On the self-similar nature of Ethernet traffic (extended version).* In IEEE/ACM Transactions on Networking, Vol. 2, No. 1, 1-15.

[Lipsky et al., 2001] Lipsky L., Jobmann M., Greiner M., and Schwefel, H.-P. *Comparison of the Analytic N-Burst Model with Other Approximations to Telecommunications Traffic.* In the Proceedings of the IEEE Symposium of Network Computing and Applications, 122-132, Cambridge, MA, USA.

[Luo and Marin, 2005] Luo S., and Marin G. A. 2005. *Realistic Internet Traffic Simulation Through Mixture Modeling and a Case Study.* In the Proceedings of the 37th Conference on Winter simulation, 2408-2416.

[Paxson and Floyd, 1995] Paxson V., and Floyd S. 1995. *Wide Area Traffic: the Failure of Poisson Modeling.* In IEEE/ACM Transactions on Networking, Vol. 3, No. 3, 226-244.

[Pionteck et al., 2006] Pionteck T., Koch R., Albrecht, C. 2006. *Applying Partial Reconfiguration to Networks-on-Chips.* In Proceedings of 2006 International Conference on Field Programmable Logic and Applications. Madrid, Spain.

[Siriwong et al., 2007] Siriwong K., Lipsky L., and Ammar R. 2007. *Study of Bursty Internet Traffic.* In the Proceedings of the IEEE Symposium of Network Computing and Applications, 53-60, Cambridge, MA, USA.

[Sprint, 2008] Sprint Academic Research Group. 2008. `ipmon.sprint.com`.

[Thid et al., 2006] Thid R., Sander I., and Jantsch A. 2006. em Flexible Bus and NoC Performance Analysis with Configurable Synthetic Workloads. In Proceedings of the 9th EUROMICRO Conference on Digital System Design (DSD). 681-688. IEEE Computer Society. Cavtat, Croatia.

[Trolltech AS, 2007] Trolltech AS. 2007. *Qt 4.3.* Whitepaper. Norway.

[Vishwanath and Vahdat, 2006] Vishwanath K. V., and Vahdat A. 2006. *Realistic and Responsive Network Traffic Generation.* In Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols For Computer Communications, Pisa, Italy, 111-122. ACM, New York, NY, USA.

[Wang et. al., 2002] Wang M., Chan N. H., Papadimitriou S., Faloutsos C., and Madhyastha T. M. 2002. *Data Mining Meets Performance Evaluation: Fast Algorithms for Modeling Bursty Traffic.* In Proceedings of the 18th International Conference on Data Engineering, 507-516, IEEE Computer Society. Washington, DC, USA.

[Weigle et. al., 2006] Weigle M. C., Adurthi P., Hernández-Campos F., Jeffay K., and Smith F. D. 2006. *Tmix: a Tool for Generating Realistic TCP Application Workloads in NS-2.* SIGCOMM Computer Communication Review, Vol. 36, Issue 3, 65-76.

Author Biographies

Carsten Albrecht received his Diploma degree in Computer Science from the University of Lübeck in 2002. In the same year, he joined the Institute of Computer Engineering, University of Lübeck. His current research topics are application-specific management of dynamically reconfigurable systems and network-on-chips. Further research interests include multi-threaded processor design and network processor architectures.

Christoph Osterloh studies Computer Science at the University of Lübeck with the objective of receiving his Diploma degree this year. His current research topics are hardware design and performance analysis in the area of reconfigurable computing and robotics.

Dr.-Ing. Thilo Pionteck received his Diploma degree in Electrical Engineering in 1999 and his PhD from the Technical University of Darmstadt in 2005. In the same year, he joined the Institute of Computer Engineering, University of Lübeck where he leads the "Reconfigurable Computing" research group. His research interests include design of dynamically reconfigurable systems, topology adaptive network-on-chips and network processors.

Roman Koch received his Diploma degree in Computer Science from the University of Lübeck in 2003. He is currently a PhD student at the Institute of Computer Engineering, University of Lübeck where he holds a position as a research assistant. His current research topic are design and prototyping of dynamically reconfigurable systems.

Prof. Dr.-Ing. Erik Maehle received his Diploma and Doctoral degree in computer science from the University of Erlangen-Nürnberg in 1977 and 1982, respectively. After subsequent positions as a postdoc at the IBM Zurich Research Laboratory and the University of Erlangen-Nürnberg he became a Professor at the Universities of Augsburg in 1987 and Paderborn in 1989. Since 1994 he is Director of the Institute of Computer Engineering at the University of Lübeck. He is Vice-Chair of the IFIP Working Group 10.3 Concurrent Systems, steering committee member of the German Special Interest Group on Parallel Processing PARS, member of the Steering Committees of the German Informatics Society (GI) on Computer and System Architecture ARCS as well as Fault-Tolerance and Dependability VerFe and speaker of GI Specialist Area Computer Engineering. His research interests include parallel and fault-tolerant computing, reconfigurable computing, robotics and embedded systems.