Assessment of Accounting Meters with Dynamic Traffic Generation based on Classification Rules

Georg Carle, Jens Tiemann, Tanja Zseby

GMD FOKUS
Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany
[carle, tiemann, zseby]@fokus.gmd.de

***Abstract*-- Charging provides an effective control mechanism for resource allocation in multi-service networks. In many cases fair charging requires the metering of used resources. Meter requirements heavily depend on the expected traffic characteristics, the charging scheme and the QoS provisioning technique. Meters for accounting are deployed in various scenarios. For a given meter architecture, performance is mainly influenced by two factors: the characteristics of the traffic mix and the classifier rules. We present a meter assessment method with a flexible testbed that allows to measure meter performance for different rule sets and a variety of traffic mixes. Our measurement setup allows to derive traffic generation from the specification of classifier rules. Our measurement results show the influence of traffic mixes and classifier rules onto needed processing resources.**

Index terms-- **IP metering, classification, accounting, traffic generation**

I. Introduction

Technicians and economists have proposed a wide variety of charging schemes for Internet services. Proposed solutions for measuring techniques include moving the measurement to the edge of the network [ShCE96] or integrating specialized modules into switches [DeWD97]. Different measurement granularity levels, e.g. virtual connections [CoKW97], or TCP flows, also have been considered.

Charging for QoS-enhanced IP Services based on resource reservations gives incentives against wasting of resources. As accurate estimation of traffic profiles in advance may be difficult for certain applications, tariffs that reflect the actual usage of resources can increase fairness. For determining resource usage, metering is required. As metering costs should not exceed a small fraction of the service provisioning costs, it should be performed efficiently for requiring little additional resources and causing little overhead.

Metering requires the classification of packets. Classification differs from filtering as follows: the result of a filtering process is a match or no-match statement while the result of a classification is an identifier for the class to which the packet matches.

For the development of accounting meters two different approaches can be distinguished. One approach is to capture all traffic with a fine granularity and to extract and group the relevant data later. An alternative approach is to provide for a flexible classification within the meter, and to meter with required granularity only the relevant flows. Both approaches have advantages and disadvantages. Capturing of all flows with a high granularity produces a high amount of accounting data that has to be sent to the accounting process and that might be stored in databases. At the same time the meter needed can be much simpler.

Meters that capture only the relevant data can reduce the amount of accounting data significantly. On the other hand they have to be configurable and must support a more complex classification process.

Cisco Netflow [Cisc99] provides a meter of the first type. All flows are captured. Classification is based on a number of header fields, which allows to achieve a fine granularity.

The NeTraMet meter [RFC2123] supports both concepts. It can be configured to explicitly classify a certain subset of the existing flows. Additionally it can be run in a modus that captures all flows. The desired granularity is controlled by an arbitrary set of configurable attributes.

A direct performance comparison of NeTraMet and Netflow is difficult because Netflow runs on Cisco IOS only. NeTraMet can be used on different platforms. Typically it resides on a dedicated workstation or PC.

In order to assess resource consumption of the metering process, the particular accounting scenario has to be considered. Meter performance depends on traffic mix, number of flow classes and number of attributes used to distinguish the classes. In our tests we investigate how the number of rules affects needed CPU resources and compare the two different metering approaches.

II. Assessment of Accounting Meters

For assessment of accounting meters different network scenarios (technique for QoS support, number of customers, charging scheme etc.) have to be considered. Meters can be assessed by applying different traffic mixes against a given set of classifier rules. Techniques for assessment of classifiers vary from the usage of real packet traces, either against standard rules defined for testing purposes (c.f. [BeMG99, McJa93, BaGP94]) or for randomly picked header field combinations [BoSS99], to the analysis of classifier rules from real ISPs and enterprise networks [GuMc99].

Structured meter testing requires not only the testing of various classifier rules, but also the definition of an appropriate traffic mix depending on the loaded rules and the objectives of testing (e.g. worst case, standard/average situation, special load conditions to certain rules, etc.). This allows a selective testing of various conditions and scenarios. Additionally it is desirable to use self generated test traffic as well as recorded real packet traces.

III. Testbed for Meter Assessment

In order to assess how meters perform in different accounting scenarios, we built a testbed that couples the traffic generation process with the classification rules used for metering. A classifier specification notation (CSN) is used as generic input for classification and traffic generation in order to control and adjust the traffic mix to the loaded rules (Figure 1). This allows to investigate worst case scenarios as well as different average conditions.

For the classification process CSN instructions are compiled into the appropriate filter notation of the meter. For the traffic generation process CSN is used as a basis for defining traffic flow variations. CSN supports instrumentation by allowing to express additional settings, to control how frequently conditions of certain classification rules become true (i.e., the "load" of individual rules) and to define additional traffic characteristics (e.g. packet size, burstiness, remaining header fields). This allows to investigate the influence of different weighting strategies (e.g. "20% of all generated flows will be classified by the first rule"). In addition to the generation of artificial traffic, our generator has the capability to send previously captured real packet traces, and to generate new traffic mixes from recorded samples. The output of the meter (metering results and internal meter statistics) can be correlated with the instrumented CSN flow description.

A. *The Classifier Specification Notation (CSN)*

The NeTraMet package provides a Simple Ruleset Language (SRL) [Brow98, RFC2723] for the definition of rules. Despite its name, SRL is not simple enough for our purposes. SRL can contain nested if-then-else statements, which complicate the automatic editing of rules. Therefore we developed a even simpler notation, the Classifier Specification Notation (CSN). CSN can be used to express Traffic Classes for the classification in the meter and for traffic generation. It can be converted into different classifier rule representations used by different meters. Key objectives for the development of CSN was a good readability and the capability to allow automated adding and removing of rules. An example for a CSN file is given below.

```
# Classification based on portnumbers
define: tanya-gen= 192.168.75.106, bsd-meter =192.168.115.204;

set : class = 1;
addrule: scrip=tanya-gen, dstip= bsd-meter, proto= udp,
dstport =2000;

set : class = 2;
addrule: scrip=tanya-gen, dstip= bsd-meter, proto= udp,
dstport =2001;
```

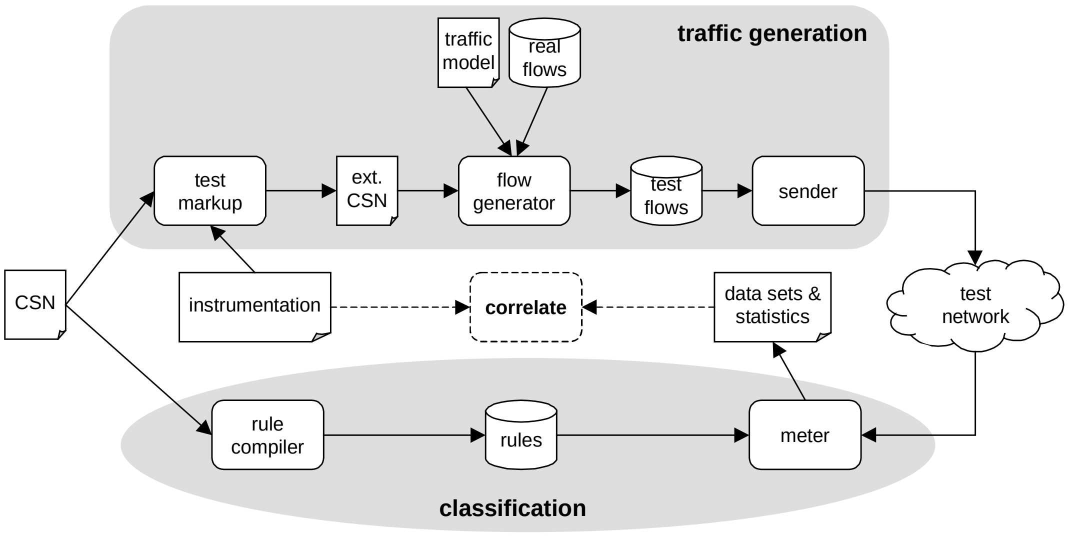

Figure 1: Test Method

CSN is used to control both, the classification process and the generation of appropriate test traffic. This allows to investigate the performance of the meter in different scenarios. With a rule compiler CSN files are translated into the appropriate rule notation which the classifier within the meter can understand. For traffic generation further information is required to define different conditions.

B. Traffic Generation with CSN

The traffic flow generation is based on an ATM test system developed at GMD FOKUS. The test system includes a special hardware, the TANYA ATM test interface. The card offers as key feature an open programming interfaces. Together with several basic software modules for measurements and testing that are running on a workstation, this system is highly flexible and can be used for a variety of measurement purposes and conditions. The test hardware TANYA offers real time features like filtering and time-stamping of incoming traffic, which is not used within our meter assessment testbed. Our testbed uses programmable hardware (FPGA, Field Programmable Gate Array) of TANYA to send ATM cells within tight time constraints. The testbed uses software modules from our FAST Advanced Tool Set which offers (beside others) traffic generation of IP flows, AAL5 segmentation/reassembly, decoding of ATM/IP traffic and presentation of interarrival and bandwidth statistics. The mentioned functionality is needed for the low-level generation of the traffic and to check the correctness of the streams.
The architecture of the flow generator used for measuring meter performance is shown in Figure 2. The system consists of a SUN workstation running Solaris with up to four TANYA ATM test interfaces. Using one or several traffic models, the workstation generates in software a traffic flow or an aggregation of traffic flows. These models can run in real-time and send the traffic directly to the network. The traffic also can be constructed off-line and loaded into a FIFO (also used for decoupling the processes during real-time access) that can hold over 2000 complete ATM cells and play them repeatedly. The number of flows for this off-line traffic generation is limited (e.g. up to 200 flows for 425 Byte IP packets, the average packet length in the German research network B-WiN), but the play-out is completely done in hardware and introduces no load to the workstation. There are also some features of the TANYA/FAST system which allow an efficient real-time

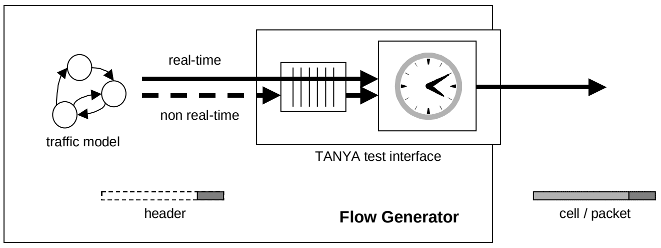

Figure 2: Traffic Generation with TANYA

generation of the traffic, e.g. the capability to generate an arbitrary IP test packet with two ATM cells (for IP header and AAL5 trailer). The timing of the transmitted stream is controlled by hardware by means of an external time-stamp.
The task of the traffic generator(s) is to define the contents of the ATM cells (in our case the IP headers with changing flow attributes) and an additional time-stamp for each cell to characterize the traffic profile. We are working with UDP flows. For the generation of TCP flows (or higher layer application flows based on TCP) an emulation is needed that runs in software on the workstation.
In our scenario we use an abstract description for the traffic flows based on CSN. The prototype traffic generator uses scripts and existing FAST tools for the following steps of flow generation:

- Extract flow information (e.g. number of flows, flow attributes) from CSN file
- Generate single UDP flows of IP packets in ATM cells (calculate traffic profile with line rate)
- Optionally generate additional traffic flow for background load
- Merge ATM cells into one file, download result for the transmitting process to TANYA
- Control the peak cell rate of the ATM stream for rate control of the IP packets.

Especially the second and third step of this process, the traffic generation of flows, require additional information which are not included in a normal meter rule file. This information, like packet length, data rates, attributes of additional/background flows, etc. is controlled via parameters of the used tools. A second version of the traffic generator will have a more integrative approach, where this additional information will be included in the CSN description and can be used by the traffic generator.

C. Meter Instrumentation with CSN

One idea behind the usage of a common notation for the representation of classification rules is to improve the comparability of measurement results. Nevertheless, meters uses different rule notations. Therefore it is required to translate rules represented in CSN into the appropriate notation for the meter under test. For this purpose we developed the compiler CSN2ruleset. This compiler takes CSN files as an input and generates the appropriate NeTraMet rulesets, which can be downloaded to the meter via SNMP by the NeTraMet manager.

IV. TEST SETUP

In the following section the test setup and conditions for the different tests are presented in detail. The results of the measurements are shown and interpreted in section V.

A. Test Goal

The goal of our first tests is to examine the influence of the number of rules onto meter performance. An additional goal is to investigate possible resource savings by restricting classification rules to the relevant flows. Furthermore we want to compare the two basic metering approaches. We plan to investigate a "fine-granular" approach where all existing flows are classified in accordance to a fixed set of attributes in comparison with a configurable approach where all relevant flows are explicitly stated and arbitrary attributes are used for flow distinction. One goal is to find out which approach has lower resource consumption.

B. *Tested Meters*

For our first tests we used the meter NeTraMet version 4.4.b5. NeTraMet is very flexible and can be applied within a variety of test scenarios. It can be used as a configurable meter by explicitly defining the relevant flow classes via a set of classifier rules. It also can act as a simple, inflexible meter that collects data from all existing flows with a fixed granularity by using the automatic flow distinction feature provided by NeTraMet.
By enabling the BSD packet filter (BPF) that resides in the BSD kernel we created a modified NeTraMet setup for which it is possible to pre-select flows in kernel space before packets are passed to NeTraMet. (The BPF provides filtering functions but no classification and therefore can not be seen as an alternative to NeTraMet.) The pre-selection can be used in accounting scenarios to filter out in advance all irrelevant flows, or to explicitly allow only relevant packets to pass. For instance if provisioning of IP services for certain networks (e.g. networks of partner providers or networks within the same country) is for free, packets with this destination can be neglected. Such packets can be filtered out in kernel space and do not need to be processed in user space by the NeTraMet classification process. In order to keep both processes comparable, we always used disjunctions of explicit classifier rules to form the filter rule for the BPF.
We analyzed the behavior of NeTraMet under various conditions. First we examined the classification process in case explicit rules are given. That means that all relevant flows are explicitly listed in the NeTraMet rulefile. Secondly we looked at the alternative approach where all flows are captured with fine granularity by using the automatic flow distinction.

C. *Test Setup*

The meter testbed for our measurements is shown in Figure 3. The flow generator workstation (a SUN SparcStation 10 under Solaris equipped with one TANYA ATM Test interface card) is connected via an ATM switch to the test network. Meter systems based on an ATM interface can be connected directly to the switch. Using ATM point-to-point connections it is an easy task to deliver all generated packets to the system under test (SUT). An Ethernet based meter can be connected between two routers or (in our network) in parallel to a real addressable destination for the generated traffic.

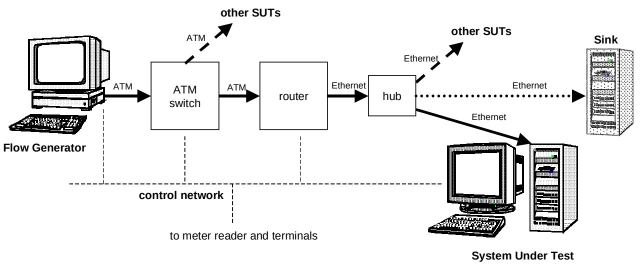

Figure 3: Test Network

In our scenario we use separate connections for the test traffic flows. The meter, the generator and other equipment is controlled via the normal infrastructure network of FOKUS, a mixed infrastructure of ATM and Ethernet/Fast Ethernet equipment.
The CPU load of the meter processing workstation is measured with vmstat, an UNIX command reporting some kernel statistics. Started with an interval option the vmstat command outputs one line of statistical data for each interval, summing up several kernel events over this period. In our measurements we were mostly interested in the CPU load, which is shown separately for user and system processes. Additional values of interest are the number of interrupts (e.g. the number of arriving packets plus some clock interrupts) and the number of system calls (related to and significantly contributing to the system load).
Although the vmstat statistics shows only the percentage of load during an interval (e.g. over 10 s) in whole numbers, there is still some fluctuation in the values. However, these measurements give a good estimation of the load of the meter processing workstation and allow to show a trend in the measurements. Nevertheless, before applying this method to a new operating system or to new configurations like for multi-processor machines it is necessary to check the accuracy of the vmstat output. (E.g., we experienced problems with vmstat accuracy when using a certain BSD variant.)

D. *Test Conditions*

We performed several tests in order to analyze meter performance under various conditions. Most measurements where done on a freeBSD PC with a 450 MHz Pentium III and 128 MB RAM (tests 1-4). In addition to this we performed tests on a SUN Ultra 1(testset 5).
In the first four tests we kept the total bandwidth (ATM bandwidth) constant to 60 Mbit/s and always sent 100 flows that differed only in one flow attribute (the portnumber). We sent UDP packets with a packet size of 200 Bytes payload. We used portnumbers in the range 2000 -2099.
In order to examine the influence of the number of rules onto meter resource consumption, we increased the number of classification rules from 0 to 100 (in steps 0, 10, 20, 50, 70, 100). In all tests we introduced a rule to first classify all flows based on a source/destination combination. Since source and destination did not vary, all packets belonged to this class.

V. Measurement Results

In the following section we analyze the results from our tests.

A. Test 1: NeTraMet, Influence of the Number of Rules

In this test we used NeTraMet with no modifications on a freeBSD machine. We used explicit rules that classify flows based on the portnumber. In order to increase the percentage of matching flows, we increased the number of rules in the NeTraMet classifier. Packets that do not match an explicit rule are collected in the class "others".

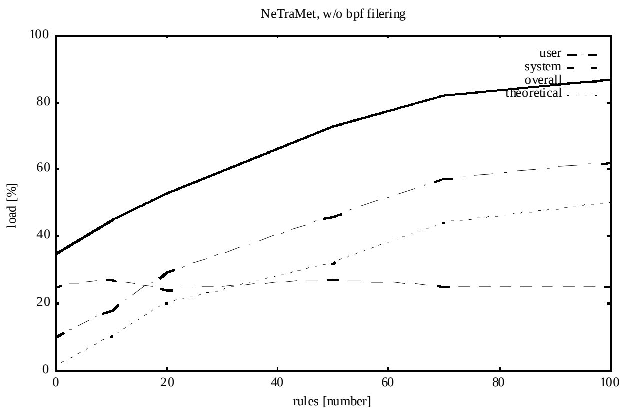

Figure 4: NeTraMet, CPU Load vs. Number of Rules

Figure 4 shows the CPU load caused by the metering process. It can be seen that the CPU utilization for kernel space processing (system load) remains nearly constant. As expected the load for user space processing increases with the number of classes distinguished by the meter. The more classes are used the more rules have to be checked for each packet. The overall load curve results from the user space and kernel space load curves.
The curve marked "theoretical" shows the relative trend of load caused by a classifier. It takes into account that all packets that do not match have to be checked against all rules. If there are only 10 rules, most packets have to be compared against all 10 rules. If 90 rules are applied, most packets match on average after 45 comparisons.

B. Test 2: NeTraMet, Worst Case Scenario

In this test we generated a worst case scenario for the meter. We purposely generated packets that never match the classes specified in the CSN.
We again used NeTraMet with no modifications and explicit rules for classes based on the portnumber. By increasing the number of classes we increased the number of rules for the NeTraMet classifier.
As none of the generated packets belongs to a given class, each packet has to be checked against all rules before it can be classified as belonging to the "others" class. As expected the CPU consumption for the user space process increases with the number of rules (Figure 5). The machine limit is reached with 70 active rules.

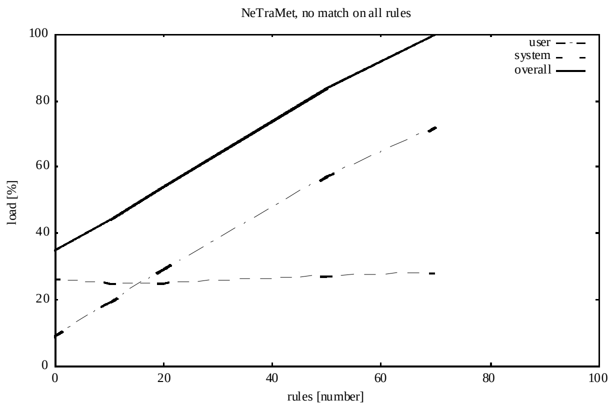

Figure 5: NeTraMet, Worst Case Scenario

C. Test 3: NeTraMet with BPF, Constant Number of Flows Filtered in Kernel Space

In this test we used our modified NeTraMet setup to enable the BPF in the BSD kernel. The BPF was used to constantly filter out 25 flows in kernel space. (From the list of flows we filtered out the last 25 with the portnumbers 2075-2099.) We used a disjunction of explicit terms (one for each portnumber) for the filter instruction.

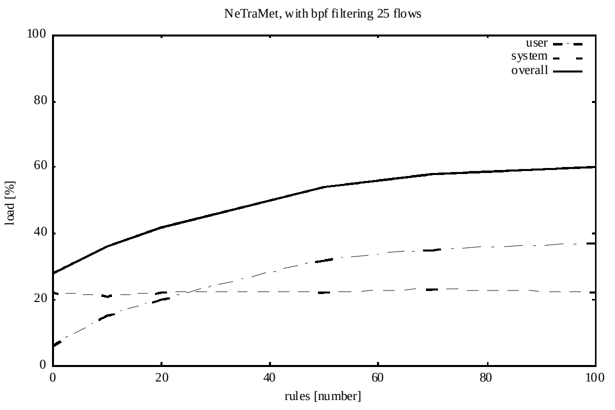

Figure 6: NeTraMet with BPF, 25 Flows Filtered in Kernel Space

As expected the total load of this test is less than the total load for test 1. The system load is nearly constant, because the BPF works for all cases with the same filter rule. Since we filtered out the last 25 flows from the list of flows, only a slight load increase can be observed when we increase the number of rules from 70 to 100.

D. Test 4: NeTraMet with BPF, Variable Number of Flows Filtered in Kernel Space

In the next test we used the modified NeTraMet setup to enable BPF in the BSD kernel. The filter rule for BPF is varied together with the NeTraMet rules. All packets that belong to irrelevant flows, that means for which no class exists in the CSN file, are filtered out by the BPF. This is achieved by combining a set of classifier rules by disjunctions into one filter rule and then passing this filter rule via NeTraMet to the BPF.

Figure 7 shows that enabling of BPF filtering significantly reduces the overall load. Like in Figure 4, the first x-axis shows the number of classes in the CSN file which directly corresponds to the number of rules in the NeTraMet rulefile. The second x-axis shows the number of terms used in the BPF filter rule. In order to examine the difference to the cases where no BPF is enabled we always applied a filter rule to the BPF consisting at least of one term.

We varied the number of terms in filter rule for the BPF in a way that the sum of BPF terms and NeTraMet rules remains constant. In case of no specific rule in the NeTraMet ruleset, the BPF contains a filter rule that combines explicit terms for all 100 portnumbers of the traffic mix. This means that no packet is passed to the NeTraMet classifier. Therefore, this case leads to zero CPU consumption for the classification process in user space. When 100 classes are distinguished in the NeTraMet classifier, the BPF passes all packets to the user space. In order to avoid disabling of BPF, we apply for this case a BPF filter rule to pass all UDP packets.

Although the number of terms in the BPF filter rule was decreased, the system load did not decrease. This means that additional terms in the filter rule do not incur a higher resource consumption for the filtering process. Instead of a decreased load we observed a slight increase. In order to understand this effect it was necessary to look at the number of system calls for this test. If the BPF contains less terms in the filter rule, more packets are passed to user space. Therefore the number of system calls and consequently the system load increases.

One surprising effect can be observed if the results for NeTraMet without BPF (Figure 4) and NeTraMet with BPF (Figure 7) are directly compared. At the point where NeTraMet has 100 active rules, in both cases all packets are passed to the user space. Although BPF is enabled in the second case it does not filter out any packet. BPF uses just one filter term that allows all UDP packets (i.e., all traffic) to pass. We expected to obtain for this case the same or a slightly higher total load than for the case without BPF. Surprisingly the total load is slightly smaller for this case. The results can be explained as follows: NeTraMet uses the libpcap function pcap_open_live to open the kernel interface. In this function a snapsize can be specified which determines for each packet the amount of data passed to user space. We expect a performance difference as observed for the case where this snapsize value is only applied if the BPF is enabled, while a disabled BPF always passes the complete packet payload to user space.

E. Testset 5: Automatic Flow Distinction with Fixed Set of Attributes

In the following combined tests we looked at the second meter approach. The question is whether the meter performance also depends on the number of flows if the automatic flow distinction feature of NeTraMet is used. For this setup all existing flows are classified based on a given set of attributes. The NeTraMet ruleset does not contain an explicit rules for each class, but rather one rule that specifies the set of attributes used to differentiate flows.

In the following tests the ruleset was not changed. Instead the number of flows where increased up to 2000 flows by modifying source addresses and portnumbers

We used two different sets of attributes to classify the packets. The all-ip ruleset puts all IP packets into one class. The src-dst-port ruleset classifies packets based on the combination of source address, destination address and port numbers.

Both tests where made on a SUN Ultra 1 workstation under two different traffic load conditions with 1Mbit/s and 2Mbit/s ATM bandwidth respectively. The packets had payload of only 1 Byte so that each packet fits into a single ATM cell. This setup leads to a very high overall load even for a low bandwidth.

For NeTraMet with automatic flow distinction the load does not depend on the number of flows. Only one rule is active and the additional flow table entries generated for each new combination of attributes does not seem to influence performance. On the other hand it can be observed that the number of attributes used for the differentiation of flows influences performance.

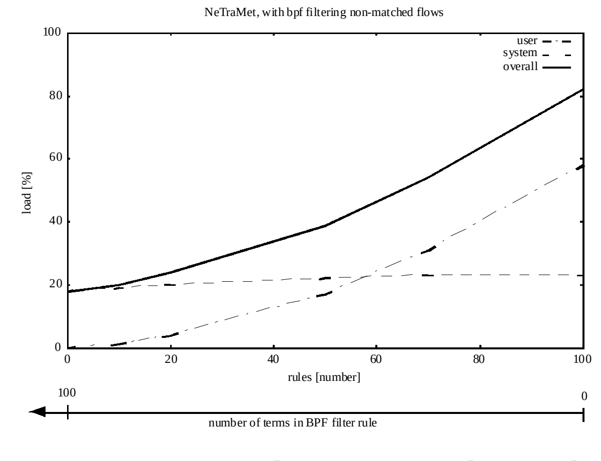

Figure 7: NeTraMet with BPF, CPU Load vs. Number of Rules

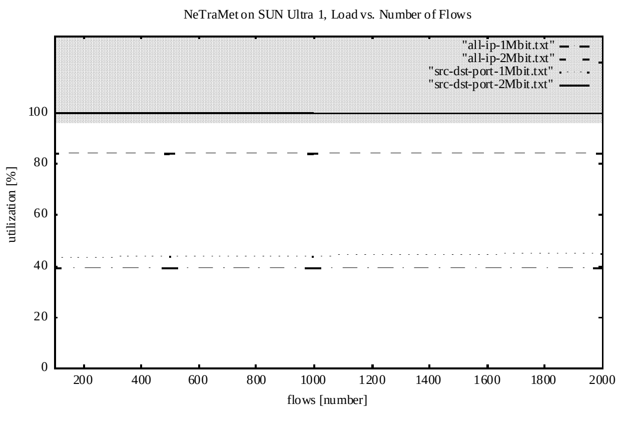

Figure 8: NeTraMet with Automatic Flow Distinction

With the fine granular differentiation (src-dst-port) we observed for the 2 Mbit/s traffic mix a CPU utilization above 100%, which means that the workstation is in overload condition where packet loss can occur. This heavy load probably is caused by the very small packet size we used which results in a about 4700 packets/s.

VI. Conclusion and Future Work

We presented a flexible testbed for the assessment of meters, where classification rules and traffic mix for various conditions can be derived from the same Classifier Specification Notation (CSN). CSN is used to describe traffic classes in a simple, human readable way. Due to its simplicity CSN allows automated editing of rules. The traffic generation in our testbed is based on the TANYA ATM card, which is supported by a sophisticated toolset.

We performed first measurements where we analyzed how the performance depends on the number of distinct classes and investigated resource consumption for different meter approaches.

As expected the resource consumption increases with the number of rules. We found out that for an automatic distinction of flows the resource consumption does not depend on the number of different flows in the traffic mix. A surprising result was that the meter performs better if a fixed attribute set is used and all flows are captured. This effect might be a result of the internal structure of the classification process and will be subject of further investigation.

For a more detailed comparison of the different metering approaches we will perform further tests with a larger number of flows. We would like to investigate whether the load for the automatic flow distinction also remains constant for a larger number of flows in the range of 10 000.

We examined a metering setup where certain classifier functions are performed in kernel space. This has been achieved by enabling BPF filter functions in the BSD kernel. As expected the overall performance can be improved by enabling kernel-level filtering of irrelevant packets with BPF. A further effect we observed was that we got a slightly better performance by simply enabling the BPF functions, even if all packets are passed to the classification process in user space. We associate this observation with a more efficient way copying of packets to user space by lipbcap. We will further investigate this effect.

Our measurement results show that not only the number but also the order of rules may influence meter performance. We plan to perform further tests to investigate these dependencies, where we plan to put different weights on specific rules by applying a different percentage of the overall traffic to these classes. We also plan to consider additional accounting scenarios: distance based accounting with a classification based on source and destination addresses, accounting for IntServ where classification is based on microflows specified by the RSVP quintuple, and DiffServ where DiffServ Codepoint information is used as further attribute.

VII. Acknowledgements

We would like to thank Lars Karow and Carsten Schmoll for their help with the measurements and the implementation of the CSN2rulefile compiler.

VIII. References

[BaGP94] Mary L. Bailey, Burra Gopal, Michael A. Pagels, Larry L. Peterson, Prasenjit Sarkar: PATHFINDER. A Pattern-Based Packet Classifier, In Proceedings of the 1st Symposium on Operating System Design and Implementation, Monterey, California, November 1994.

[BeMG99] Andrew Begel, Steven McCanne, and Susan L. Graham: BPF+: Exploiting Global Data-Flow Optimization in a Generalized Packet Filter Architecture, Proceedings of ACM SIGCOMM 1999, Cambridge, Massachusetts, USA, September 1999

[BoSS99] Niklas Borg, Emil Svanberg, Olov Schelén: Efficient Multi-field Packet Classification for QoS Purposes. International Workshop on Quality of Service (IWQoS'99), London, UK, June 1999.

[Brow98] N. Brownlee: srl Compiler and Language User's Guide Version 4.2, Information Technology Systems & Services, The University of Auckland, New Zealand, August 1998 (http://www.auckland.ac.nz/net/Accounting/ntm.Release.note.html)

[Cisc99] Cisco Systems: Cisco IOS Software solutions, NetFlow Services and Applications, http://www.cisco.com/warp/public/732/netflow/

[CoKW97] C. Courcoubetis, F. Kelly, and R. Weber. Measurement-based Charging in Communication Networks. Statistical Laboratory Research Report 1997-19, University of Cambridge, 1997

[DeWD97] D. Decasper, M. Waldvogel, Z. Dittia, H. Adiseshu, G. Parulkar, and B. Plattner. Crossbow - A Toolkit for Integrated Services over Cell-Switched IPv6. In Proc. of the IEEE ATM'97 Workshop, Lisboa, Portugal, June 1997.

[EnKa99] Dawson Engler, Frans Kaashoek: DPF: Fast, Felxible Message Demultiplexing using Dynamic Code Generation. Proceedings of ACM SIGCOMM 1999, Cambridge, Massachusetts, USA, September 1999

[GuMc99] Pankaj Gupta and Nick McKeown: Packet Classification on Multiple Fields, Proceedings ACM SIGCOMM 1999, Cambridge, Massachusetts, USA, September 1999

[McJa93] Steve McCanne, Van Jacobson: The BSD Packet Filter: A New Architecture for User-Level Packet Capture, USENIX Winter Technical Conference, Monterey, California, November 1993, pp. 259-269.

[RFC2123] N. Brownlee. Traffic Flow Measurement: Experiences with NeTraMet. IETF RFC2123, March 1997.

[RFC2723] N. Brownlee: SRL: A Language for Describing Traffic Flows and Specifying Actions for Flow Groups, RFC 2723, October 1999

[ShCE96] S. Shenker, D. Clark, D. Estrin and S. Herzog. Pricing in Computer Networks: Reshaping the Research Agenda. Communications Policy. Vol. 20(1), 1996.