Section 2:
Network monitoring based on flow measurement techniques

This research is performed within the scope of the SURFnet Research on Networking (RON) project (Activity 1.2 - Measurement Scenarios).

Authors
Michiel Uithol  University of Twente
Vincent van Kooten  University of Twente

Supervisors
Aiko Pras   University of Twente
Pieter-Tjerk de Boer  University of Twente

Abstract

In current networks an improved monitoring service might be required, which provides usage and performance figures to its users. In the field of usage figures there are possibilities for improvements to be made on top of the currently available statistics at SURFnet. This document is written with the aim to provide insight into the possibilities for such a monitoring service. A clear overview of currently available open source tools for flow measurement is provided. Some scenarios descriptions provide insight in solutions for deploying some of these tools.

Keywords: SURFnet, Flow measurement, Tools
Chapter 1: Introduction

This report describes how traffic measurement and reporting can be presented to SURFnet users. These users are specified to get insight in their demands.

In order to fulfill the users’ needs several measurement techniques are evaluated. In the SURFnet network a new type of usage has arisen: extremely large data streams between two fixed points in the network. Flows are a good solution to locate and monitor these streams.

The use of flows provides a compact method to export usage information. In relative large networks like SURFnet this will still generate gigabytes of data per day. For this amount of data it is important to have an efficient storage system.

Different standards are available for exporting flow information. To provide an accurate advice about which standard should be used, an overview of these standards and their future usage is given.

The selected standard is taken into account for the tool requirements. With these requirements a selection is made among various open source tools. The most promising tools in this preliminary selection are selected for testing.

By performing these tests an overview of the usability of the tools is created. Tools performing well in the test phase are presented in scenarios. These scenarios give an example of how these tools can be deployed in the network.
Chapter 2: Target users

The authors have chosen to describe four different kinds of users covering the most practical situations. These users are derived from the situation at the University of Twente and SURFnet. All profiles contain different responsibilities and or interests in the network. The four profiles are described as:

- Network managers (SURFnet)
- Sub-domain administrators (such as the ITBE)
- Interested end-users (Researchers)
- Regular end-users (Students)

SURFnet is the organization that manages the research fibre network covering the Netherlands. The ITBE is connected to this fibre network and manages the sub-domain at the University of Twente. The interested end-users also include interested students.

Please note that sub-domains are not widely used within SURFnet’s topology. Some end-users are directly connected to the network’s back-bone; others are connected through a large sub-domain. End-users directly connected to the backbone are interpreted as sub-domain administrators.

2.1 Regular end-user

The regular end-user has no interest in the network’s topology or whatsoever. This category encapsulates students, employees and researchers who just want a reliable and fast internet connection. For such an individual the network is like a cellular phone. It does not matter how it works, as long as it functions he or she is satisfied. Some more advanced regular end-user might be monitoring his or her own PC.

An investigation by INTERVIEW/NSS for SURFnet points out that 85% of its users has never heard about SURFnet’s existence or is unaware of making use of it [INT04]. This group of individuals is also considered as regular end-users.

2.2 Interested end-user

Besides interest in their private connection, the interested end-user likes to have some insight in their connection with SURFnet’s core-network. Information from this router should thus be available for such a user. “What is my router’s throughput and quality of services?” are issues this particular user could wonder about. Also connectivity between the user’s router and other routers on the network should be visible.

A scenario could be that the sub-domain presents to its interested end-users what sort of traffic is traversing through the connection with the back-bone. Network utilization of services like bit-torrent, ftp, edonkey, skype and more can be graphed and presented in tables.

2.3 Sub-domain administrator

Since this type of user needs insight in the core-network for his or her profession, more detail on the network’s behaviour is required.

Next to the backbone statistics they require in-depth-information of their sub-domain, users and facilities. As far as the core network is concerned, they mainly need to know the throughput which their access point towards SURFnet accounts for.
Extras the user requires are statistics that offer detailed throughput information from and to other specific points within their sub-domain. When the sub-domain’s bandwidth usage towards the backbone is increasing, measurements should point out where the increased usage is originating. It might also be that the usage of a certain service e.g. ftp has overall increased dramatically. The sub-domain administrators can use these detailed statistics for performing more precise actions.

2.4 Network manager

Currently throughput is only measured at specific points, the data’s source and destination is unknown. This system should be improved by presenting the throughput between specific points. It can also be seen what parties connected to SURFnet are making extensive use of the relatively more expensive transatlantic links.

For network managers it is important to react fast on network-failures. Therefore not only monitoring is suited, but also the ability to set threshold-alerts might be welcome. When a threshold is reached, the network managers should be notified.

This group has no specific interest in the network’s sub-domains and focuses only the core-network. To get more insight in what types of traffic is entering the core-network; the measurements on sub-domain level should be accessed.
Chapter 3: Measurement techniques

To provide solutions for users’ needs, this chapter first discusses methods for active and passive monitoring. A more in depth view into flow measurement is provided because this is the most promising method to present throughput between specific points. For SURFnet this method will result in gigabytes of data generated every day. A possible storage method for this problem is introduced.

3.1 Active

With active monitoring artificial traffic is injected into the network for measurement purposes. Statistics are generated based on the network’s reaction to this known traffic. Common methods are the use of Round Trip Time (RTT) and one-way delay measurements. RTT measures the total propagation delay to and from another observation point.

One-way delay measurements give an estimation of the time it takes to propagate from one point to another within the network. This means that at both ends of the measurement clock synchronization has to be performed. Synchronization can be achieved by using the Global Positioning System (GPS).

Stress-tests demand more resources from the network. For getting insight in the maximum throughput of a connection, one can overload one side of the link and listen at the other side. Throughput, delay, errors and other Quality of Service (QoS) aspects can be measured this way. Such a method should be applied for just a short period, but in any case it will bother the network’s current users. Since it is not desirable to let the network’s quality suffer during measurement, this method is not widely used. Such stress-tests could however be performed over private connections.

Active monitoring allows performing controllable experiments that are not possible with passive monitoring. A drawback is that the injected traffic is not fully representative for a network’s behaviour, e.g. the RTT measurements performed by ping rely on the Internet Control Message Protocol (ICMP). This protocol is treated different from normal traffic [PAM01].

3.2 Passive

Passive monitoring makes use of already existing network traffic traversing through observation points. In other words, it does not produce any extra traffic specifically needed for measurement purposes. For collecting and combining all measurements in one centralized database, some additional bandwidth is still needed for passive monitoring.

Data collected via passive monitoring can be presented by observation points in two different ways: as Management Information Base (MIB) variables fetched by the Simple Network Management Protocol (SNMP) or by exporting flow information.

A SNMP monitor can periodically request some MIB parameters from an SNMP agent. Examples of these parameters are the current throughput, the number of occurred errors, loss percentage and CPU utilization. The polling mechanism will return only information as it is on a specific moment of time. Running a SNMP agent does not require many resources on the monitored system. Besides that, most network components are probably already equipped with a SNMP agent.
Exporting flow information is usually done by aggregation into flow datagrams. A flow datagram describes the source, destination and size of a flow. This information provides a method for getting a detailed view of network traffic. The use of flows allows differentiation among used protocols and traffic directions.

Aggregation into flows requires resources from the observation point. In a worst case scenario this can result in a decreased routing capacity of the network. This can be prevented by using hardware support or by locating flow technology at a separate device directly connected to the observation point.

Depending on the network size, gigabytes of flow data will be generated every hour\(^1\). All this data can not be stored and filtering is required. Filtering will cause loss in detail.

Concluding, support for SNMP is most likely already available in most network components and makes this technology attractive. Flow measurement needs hardware support or the deployment of extra devices. Advantages are the availability of a much more detailed and accurate image of the network’s usage compared to SNMP. The price is a more demanding technology.

### 3.3 Flow measurement architecture

To perform flow measurements several components are required: probes, collectors and analyzers. Figure 2.1 gives an abstract overview of a simple architecture derived from the methods that most flow measurement tools are using. In some implementations the roles of probes, collectors and analyzers are combined.

![Diagram](image)

**Figure 2.1: The probe-collector-analyzer architecture.**

During the lifecycle of the system, probes will collect data from the network at specific locations. Probes have just little intelligence; their main purposes are the efficient collection of traffic-data, aggregate this into flow entries and continuously forward these entries to the collector. Forwarding is, in most cases, done by sending packets over UDP.

All the gathered flow-data will be stored in one or more collectors. When multiple probes are deployed in a network, they can produce an enormous amount of data.

\(^1\) Example: Uninett (Norway) produces 7GB data a day, this is after filtering and compression [MIU04]. Uninett is a smaller network compared to SURFnet.
Therefore storage has to be done in an efficient way. For better performances on larger networks functionality must be distributed over several systems.

The analyzer is the presentation tool towards the user. When the user makes a selection among the available data, the analyzer will send a request to the collector that returns the requested data. The analyzer will then present the results to the user.

To be able to handle the amount of data generated by a flow measurement system, it is advisable to use a database that aggregates over time. Data inserted to the database has the highest level of detail. When this data gets older it will be aggregated into entries averaged over a period of time. The data is moved forward to another section within the database. The older the data gets, the less detail remains.

![Figure 2.2: Aggregation of data over time.](image)

*Figure 2.2: Aggregation of data over time.*

Figure 2.2 is an example of how an aggregating database can be configured. Every hour all data more than one day old will be summarized into 5 minute averages. These new entries will move into the ‘1 Week’ section of the database and the original data is removed. After every week the data in the ‘1 Week’ section will be summarized and is moved to the ‘1 Month’ section. This process repeats itself for the number of configured sections. The example shows four different sections with different granularities.

The Round Robin Database (RRD), a part of the RRD-tool toolset by Tobi Oetiker, performs the aggregating property by itself. Other solutions are the use of more common databases and perform aggregation by scripting.
Chapter 4: Flows and Flow formats

The following text contains a review of the main formats used for exporting router-based information about network traffic flows. Flow data is sent to collection devices and network management systems in order to be stored and processed.

After a short introduction on flows, the pros and cons of each type are named. This summary of the different available types is followed by a conclusion. A determination is made which of these flow aggregation types is preferably used.

The flow standards introduced in this chapter are roughly dividable into two categories. The first category consists of the standards where aggregation to flows is performed at the collector. Realtime Traffic Flow Measurement (RTFM) and sFlow are available in this category. The second category consists of the standards that aggregate to flows directly at the probe. NetFlow and Internet Protocol Flow Information eXport (IPFIX) represent this second category.

4.1 Flows

In the old days, analyzing network traffic was possible by storing all packets or headers that passed through a router. Soon it appeared this was not feasible for higher speeds and bigger networks. The solution was to determine flows in the network data. Using this aggregation technique meant a 20:1 decrease in amount of data that has to be stored [BBR05].

In order to export data, routers and switches represent network traffic flows based on seven key fields:
- source IP address
- destination IP address
- source port
- destination port
- layer 3 (L3) protocol type
- type-of-service (ToS-byte)
- input logical interface

![Fields used for flow recognition](image-url)
In addition to the first four fields in which generally flows are described, NetFlow and IPFIX also use the last three fields.

If these seven fields in two packets match, both will be considered as belonging to the same flow. New arriving packets are matched with the same criteria and might be added to an existing flow. If a match does not occur, a new flow will be created. There are additional non-key fields that can be used for network accounting purposes, such as the source IP mask, destination IP mask, source autonomous system, destination autonomous system, TCP flags, destination interface and IP next-hop. These can be embedded in some of the flow formats.

4.1.1 Flow sampling

In addition to flow aggregation, sampling is used by some routers and probes to be able to keep up with the vast amount of data. This method was first introduced to flows by CISCO because of the high CPU load resulting from the use of NetFlow. Before being used in flows, the concept was already around since 1991, it was yet only used for packet capturing. It is based on the statistical fact that multiplying gives roughly the amount of data that actually flows through the routers and switches [DLT02].

There was also an initiative to create and promote smart sampling, based on the principle of ignoring mostly the smallest flows, but this project has yet to prove its usefulness. This method of sampling would have eliminated primarily UDP and ICMP. The sampling methods differ in their sampling strategy, systematic or random. Of these two methods random is preferred because otherwise it is possible to be synching on a pattern in the packets. [DLT04]

4.1.2 Flow templates

Templates enable the possibility to use one method for outputting flows from both routers and switches. These have different requirements for output fields and were previously separated into different versions of NetFlow.

When templates are used there are fields that can be defined by an administrator. These fields can be configured to contain extra flow characteristics. These templates inform the collector of the type of data it can expect.

Flow templates encourage developments in flows because of the ease in altering the fields in the export format. Templates are a feature that should allow future enhancements to flows without requiring concurrent changes to the basic flow-record format. Cisco was the first with implementing templates. The template function has also been implemented into IPFIX.

Flow datagrams sent to collector contain the template that is used. This shows how the flow data is formatted. This ensures that the collector will react in the correct way to the received data.

The usage of templates in flow exports results in more complex collectors. Collectors have to anticipate to the different templates that are received.
4.1.3 Flow usage

General data that can be derived from analyzing flows:
- Detecting and diagnosing network problems.
- Real-time congestion management.
- Understanding the application usage (e.g., P2P, Web, DNS etc) over the network and detect changes in this usage.
- Usage accounting for billing and charge-back.
- Audit trail analysis to identify unauthorized network activity and trace the sources of denial-of-service attacks.
- Route profiling and peering optimization.
- Trending and capacity planning.
- Data warehousing and mining.

These are the most common uses of Flows. Flows can also be used to measure the exact delays end-users experience. This can be calculated by recording the time between the pair of TCP SYN and TCP ACK packets. Measuring these delays with flows is not often performed because of the major adjustments that must be made to the probe.

4.2 RTFM

When RTFM’s RFC2721 was published in October 1999, its development stopped. Records indicate that work was in progress on RTFM already as early as 1994. One of the major users of this format was NetraMet, which also seems to have ceased to produce any results in the recent years. Since December 2000 no updates have been reported [RTM00].
Users may specify their traffic flow measurement requirements by writing rule sets, allowing them to collect the flow data they need while ignoring other traffic. RTFM has a slightly different architecture from the usual probe-collector combination. It consists of three parts, the meter (probe), the meter reader (collector) and a manager. The manager can interact with both the probe and collector for configuration purposes [RTC99]. RTFM is also different because it is basically built on a SNMP MIB used by the meter. With this SNMP interaction flows are recorded.

**RTFM discussion**

Granularity of the data is exactly configurable by an administrator with use of rule sets. RTFM was a widely used format because of popularity of NetraMet, which means that much knowledge about RTFM is available.

Support of RTFM seems to have stopped, its technology is getting behind compared to the newer sFlow and NetFlow.

**Conclusion**

It is not advisable to create any new application based on RTFM because it is not widely used and supported. However, RTFM provides a method for sampling every n<sup>th</sup> packet, so it could be able to keep up with gigabit speed networks.

The RTFM’s developers will make no further progress in the direction of differentiating new technologies. Although adjustments can be made by administrators this will make RTFM more and more a legacy product and unsuitable for the new development in technology.

**4.3 sFlow**

SFlow is managed by sflow.org [SFL01]. The sFlow system is based on the old principle of packet sampling as explained in the flow sampling section. SFlow is mainly aimed at being available as an embedded agent. Furthermore it is based on the same structure of agent (probe) and collector as the other Flow formats.

First definitions of sFlow were made in 2001. From the start the project is aiming to deliver sampled flows for gigabit speed networking. Attempts are made to let around 1000 agents use the same collector.

SFlow also introduces a MIB for remotely controlling and configuring an agent by a network administrator. This is a useful feature that is also implemented in RTFM.

The difference between sFlow and NetFlow mainly consists of the fact that sFlow does not aggregate into flows on the router itself. The probe only sends a sampled packet to the collector. Before sending the packet, relevant information on forwarding and an interface counter is added. By using this method of sampling, the router does not have to cache all flows and is more robust and better scalable. It also indicates that sFlow is just a packet oriented system and not really flow oriented.

**sFlow discussion**

SFlow offers the user sampling methods for lower load on observation points. This is an advantage when deployed on a gigabit speed network. Capabilities include support for Internetwork Packet eXchange (IPX) and various other protocols because of the packet level exports.
The format includes support for Virtual LAN (VLAN) tags, which provide more insight into layer 2 switching.

A disadvantage of sFlow is the minor support from router manufacturers. In the current form it is not a real flow format, because in reality only sampled packets are sent from the probe. This also means that despite a small amount of flows, a lot of UDP packets will be sent over the network. Because sFlow lets the collector aggregate into flows, this can result in a bit higher load on the collector.

**Conclusion**

SFlow might be promising when it manages to get more support from large companies. The functions in the latest version of sFlow are good and allow easy adaptation to other protocol types (e.g. IPX) without updates to routers. Currently sFlow’s largest supporters are Hitachi, Foundry Networks and Hewlett Packard. Despite that CISCO started with the research of sampling earlier, sFlow made faster progression on implementing new sampling techniques. It seems sFlow can react faster to market demands, while CISCO is trying to make features but lagging behind with the actual implementation into NetFlow.

Another benefit is its simple method for sample pre-processing. This is particularly good for performance.

### 4.4 NetFlow

The generic term NetFlow refers to multiple different notions: the metering process, the exporting process and the export protocol, as defined in the IPFIX terminology.

The principle behind NetFlow exists since 1996 when Darren Kerr and Barry Bruins developed it at CISCO.

**NetFlow version overview**

The NetFlow version 1 format is the original format supported in the initial Cisco Internetwork Operating System (IOS) releases containing NetFlow functionality. The NetFlow version 5 format is an enhancement that adds Border Gateway Protocol (BGP) autonomous system information and flow sequence numbers. This version is still widespread and used the most of all NetFlow versions.

The NetFlow version 7 format is an enhancement that adds NetFlow support for Cisco Catalyst 5000 series switches equipped with a NetFlow Feature Card (NFFC). Besides the Catalyst 5000 NFFC no other equipment makes use of version 7.

Version 8 is the NetFlow Export format used when the Router-Based NetFlow Aggregation feature is enabled on Cisco IOS router platforms [NFL02].

The latest NetFlow version 9 format is the first template based design for flow records. These templates provide an extensible design to the record format, a feature that should allow future enhancements to NetFlow services without requiring concurrent changes to the basic flow-record format. Version 9 also introduces IPv6 hosts [NFL03].

NetFlow also has a distinct feature of limiting the length of flows in standard options to 30 minutes. This is an extra feature to prevent connections that are very long, to be displayed incorrectly. Because throughput is fluctuating throughout the existence of the...
flow detail over time is lost. If these flows were allowed to continue indefinitely they would create an unrealistic and incorrect throughput.

**Netflow discussion**

NetFlow has already evolved into an industry standard. The templates introduced by CISCO since version 9 give this protocol extra room for input from administrators. Sampling which has become available is also positive. This will ensure a lower load on routers when they are monitoring gigabit speed links.

NetFlow versions prior to version 9 are not flexible and are not able to handle IPv6. There is no support for IPX or VLAN tags in NetFlow. It is limited to TCP and UDP recognition. Older routers that support NetFlow version 5 are also known to cause high CPU loads.

**Conclusion**

For any new NetFlow implementation it is advisable to use NetFlow version 9. The older version 5 is still usable but is not be able to distinguish IPv6 and thus has less value for network analysis if this feature is wanted. On top of that, version 9 has an improved support for sampling. The sampling will drastically lower the CPU load on the equipment exporting NetFlow.

NetFlow should be ready for the challenge it faces with sFlow gaining in popularity. sFlow contains more features then is possible with NetFlow. Also sFlow has another implementation of the sampling mechanism, which is more tolerant to (UDP) packet loss. Improvements to NetFlow can be made on security and reliability. With the support of CISCO there should be sufficient embedded hardware compatible with NetFlow. This, together with that fact that soon NetFlow version 9 will be compliant to the IPFIX standard, makes NetFlow a good and well supported format.

**4.5 IPFIX**

IPFIX is an initiative from the Internet Engineering Task Force (IETF) to standardize the format used for the export of flows. Flows are used to export router-based information about network traffic towards data collection devices and network management systems [IPF03][IPF04].

IPFIX is based on the NetFlow version 9 implementation of CISCO. Thus it is a flexible and template based definition of the format by which IP flow information can be transferred from different kinds of network equipment to a collector. With the use of templates network managers will be free to change the fields of their flows. As such it is very useful for experimenting with different kinds of views on the network. This could give new insight in the network performance and possibilities for improvement.

**IPFIX discussion**

Standards are useful for the future improvements to the layout of the flows and will ensure compatibility between various products.

IPFIX is based on the NetFlow version 9 template structure, which makes it a flexible standard. Sampling is available for lower load on routers when monitoring very high speed links.
Currently there are no available probes that produce IPFIX compliant flows, because of the short time this standard has been around since its real publication in 2004. An announcement has been made that nProbe will support IPFIX in the future. Issuing a standard could lead to a slowdown in future improvements; this is partially negated by the templates. This will encourage developers to define new fields.

**Conclusion**

At the moment only a few implementations with the IPFIX standard are being tested, and none are currently available to the public. This new standard does have future opportunities when these implementations become available. Recently announcements were made that IPFIX will be supported in the future by nTop and Stager. The little use of this new format is probably explained by the fact that there are not many probes capable of producing this format.

There was also a discussion going on about IPFIX and the use of Stream Control Transmission Protocol (SCTP) for the transport of the flow data. Although it is an innovative idea, its benefits do not compare to the drawback that SCTP is not widely supported on platforms.

Overall IPFIX seems to get accepted. It should not take too long before NetFlow version 9 becomes compliant with IPFIX, since IPFIX is being developed by CISCO employees.

**4.6 Summary**

In conclusion, for an implementation currently being chosen it would be logical to check whether and which exports the current routers or switches already support. NetFlow, IPFIX and sFlow are all good implementations with their own pros and cons. The author’s preference would go to IPFIX though for the reasons stated below.

**4.6.1 Transit and loss**

In the case a NetFlow or IPFIX UDP packet is lost in transit, you may have lost as little as one DNS lookup flow (one packet) or you may have lost several gigabyte size file transfers. There is no way to know. Running over TCP or SCTP to correct this problem is no solution because it raises further scalability issues. The SCTP issue was discussed in the development of IPFIX and was rejected in favour of UDP. Issues with scalability consist of the extra buffering of packets in case a resend is needed and the complexity of the setup of the connection. Both require additional resources while the load on the router should be minimized.

In contrast, sFlow is more tolerant to packet loss. One percent loss is equal to adjusting the sampling rate from 1-in-100 to 1-in-101. This is more reliable then the unknown deviation with NetFlow.

**4.6.2 Router/probe load**

The NetFlow router needs an indeterminate amount of RAM to hold the flow-cache. CPU is required to walk the flow-cache and flush expired flows. In a layer-2 switch, extra processing is required to decode fields from each packet. A sFlow agent has no cache and requires no decode, so none of this work is needed. This makes the load on the router
minimal compared to unsampled NetFlow. This should, in theory, even have a lower load than sampled NetFlow.

4.6.3 Collector load

For both sFlow and NetFlow there are issues here. The sFlow collector needs to save the received packet into a flow. NetFlow and IPFIX collectors need to look at the timestamps that are present and spread the throughput over the amount of time that elapsed.

4.6.4 Probe on mirror port

For probes running on a system connected to a mirror port. This means that all traffic traversing the switch or router is mirrored on one port. sFlow, IPFIX and NetFlow are suited.

4.6.5 Overall

A final advantage of NetFlow and IPFIX is the usage of templates. Within limitations, an administrator can exactly define the data returned with flow packets. In a direct confrontation of sFlow and NetFlow, both embedded in a router, there is no real preference. SFlow might give a small performance benefit but currently there are much more systems with embedded support for NetFlow compared to sFlow.

At the time there are not many open source sFlow projects going on. Third part probes are therefore very rare, resulting that NetFlow and IPFIX are preferred. These exports have got broader support.

The RTFM implementation should not be counted on to implement further improvements. IPFIX will probably evolve to a standard that is actually used for exporting the data from routers and switches. CISCO employees largely do the design of IPFIX and NetFlow will very likely be IPFIX compliant in the future. For sFlow being a success in the near future, it should soon gain broader support.

Because of all the reasons stated above the author’s preference out to IPFIX and encourage any implementation of this standard.
Chapter 5: Tool requirements

The requirements provided here are used to create reviews of the open source tools. Readers can also use these requirements as a basis for making a tool selection for themselves. Tools surviving this selection are described in chapter 6, the remaining tools can be found in appendix A.

5.1 Probe
- Because of current transfer speeds reached in routers, the probe should be capable of handling gigabit speeds.
- Should have no noticeable impact on routing capacity. This can be achieved by using a separate probing system.
- Should run on a widely available operating system.
- There should be a recent stable release of the tool.
- Capable of monitoring multiple interfaces on the same observation point.
- Good documentation and support.
- Continuity of the tool: It must be supported for some time in the future.
- Multiple instances of the probe should be able to monitor multiple interfaces.
- Future support for IPFIX and current support for NetFlow.

5.2 Collector
- Be capable of handling sampled flows.
- Pre-selection of data.
- Performs aggregation over time for more storage capacity.
- End-user anonymity must be ensured.
- Should run on a wide-spread operating system.
- Scalability by distribution of functionality is very welcome.
- Future support for IPFIX and current support for NetFlow.

5.3 Analyzer
- The preference goes to a web-based presentation tool. This makes it possible for any user at any place to take an in-depth look into the network’s behaviour.
- Make use of clear and easy navigation, every user should be able to get the wanted information within a few mouse-clicks.
- Support for user accounts. This makes it possible to distinguish the level of information provided to the different types of users.
- Compatibility, it should be able to cooperate with a selection of the collectors.
- Able to process data from several distributed collectors.
- Must support the plotting of graphs. If appropriate, they should preferably be used instead of tables.
- Tables should only be used as extension to graphs or in the occasion when graphing is not appropriate.
- Graphs and tables should at least:
  - make use of different colours
  - have a clear legend
- Additional requirements for graphs:
  - an optional logarithmic scale for the measured values
  - a zoom function would be very welcome

### 5.4 Throughput

Throughput information should be available in the forms of average, maximum (peaks) and minimum (lows) for both incoming and outgoing data. These should be available for a single observation point\(^3\) and for a link between two observation points. Throughput has to be presented over the following time scales:

- predefined time scales (last hour, day, week, month, year)
- user defined periods of time
- since the activation of the observation point

For comparison it would be useful to combine the throughput information from several observation points in one graph or table.

The total system should be capable to detect and distinct the following aspects of flows:

- IPv4 & IPv6
- TCP & UDP

### 5.5 End-user anonymity

When implementing a measurement system in a network, always keep in mind that data can be sensitive. Measures have to be taken to handle this correctly. Measurement systems are capable of recording private information, it is not recommended to do so. When such information is stored, it might for some reason reach non authorized people. For privacy reasons the gathered data should be filtered and aggregated properly. Detailed information about end-users will then be removed.

Flow PDUs that traverse over the network are not encoded. These PDUs could thus be sniffed and viewed by a third party.

### 5.6 Backward compatibility

Backward compatibility should be taken into account when selecting measurement tools for a network on which measurement is already performed. When preferred and available, historic statistics should be parsed into the new information system. This makes comparison possible between new and legacy statistics. Please take note that this is only feasible for the same types of data.

---

\(^3\) An observation point will most likely be a router in the core-network or a device connected to it.
Chapter 6: Tool summary

A short description is created for all the tools that survived the primary selection conform to the requirements as stated in chapter 5. The authors have tested all these tools. More detailed descriptions are found in the appendices B to G. Tools not compliant to the requirements are described in appendix A.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Probe</th>
<th>Collector</th>
<th>Analyzer</th>
</tr>
</thead>
<tbody>
<tr>
<td>nProbe</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Softflowd</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pmacct</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Flow-Tools</td>
<td></td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>nTop</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Cflowd</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Stager</td>
<td>yes</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Cacti</td>
<td></td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Flowscan+</td>
<td></td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Pmacct-fe</td>
<td></td>
<td></td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 6.1: overview of the selected tools.

6.1 nProbe

role probe
tested versions demo v3.1, July 16, 2004
developer(s) Luca Deri
URL http://www.ntop.org/nProbe.html
license type GPL
support & documentation good
dependencies Libpcap
OS POSIX, win32
flow format nFlow, NetFlow versions 5 and 9
real-time/legacy n/a
graphs/tables/logarithmic n/a
see for test information appendix B

Nprobe comes from the same developer as nTop. Deri also created the PF_Ring, which makes packet capture more efficient. This PF_Ring can also be used in conjunction with nProbe for an improved performance.

Nprobe captures network traffic and efficiently aggregates them into flows. The program is made to have a limited memory footprint (less than 2 MB of memory regardless of the network size) and has a low CPU load. So it is able to run on many platforms, even older ones.

NProbe has got a very clear manual and was easy to test because of the binary distribution. No problems occurred while testing and it is a very useful and easy to use tool. According to other users experiences nProbe is very CPU savvy.
### 6.2 Softflowd

<table>
<thead>
<tr>
<th>role</th>
<th>probe</th>
</tr>
</thead>
<tbody>
<tr>
<td>tested versions</td>
<td>v0.9.7, January 14, 2005</td>
</tr>
<tr>
<td>developer(s)</td>
<td>Damien Miller</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://www.mindrot.org/softflowd.html">http://www.mindrot.org/softflowd.html</a></td>
</tr>
<tr>
<td>license type</td>
<td>BSD License (revised)</td>
</tr>
<tr>
<td>support &amp; documentation</td>
<td>average</td>
</tr>
<tr>
<td>dependencies</td>
<td>Libpcap</td>
</tr>
<tr>
<td>OS</td>
<td>POSIX</td>
</tr>
<tr>
<td>flow format</td>
<td>NetFlow versions 1, 5 and 9</td>
</tr>
<tr>
<td>real-time/legacy</td>
<td>n/a</td>
</tr>
<tr>
<td>graphs/tables/logarithmic</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Softflowd creates flows by listening on a network interface or by reading a packet capture file. These flows are reported via NetFlow to a collecting host or can be summarized within Softflowd itself.

Softflowd is fully IPv6 capable; it can track IPv6 flows and export to IPv6 hosts. During testing compatibility issues were found. Softflowd appears to create non-existing traffic when used in combination with nTop, see appendix D for more information about this.

### 6.3 Pmacct

<table>
<thead>
<tr>
<th>role</th>
<th>probe and collector</th>
</tr>
</thead>
<tbody>
<tr>
<td>tested versions</td>
<td>0.8.2, 0.8.6 and 0.8.7, June 14, 2005</td>
</tr>
<tr>
<td>developer(s)</td>
<td>Paolo Lucente</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://www.ba.cnr.it/~paolo/pmacct/">http://www.ba.cnr.it/~paolo/pmacct/</a></td>
</tr>
<tr>
<td>license type</td>
<td>GNU General Public License, Version 2</td>
</tr>
<tr>
<td>support &amp; documentation</td>
<td>good</td>
</tr>
<tr>
<td>dependencies</td>
<td>Libpcap</td>
</tr>
<tr>
<td>OS</td>
<td>POSIX</td>
</tr>
<tr>
<td>flow format</td>
<td>none</td>
</tr>
<tr>
<td>real-time/legacy</td>
<td>n/a</td>
</tr>
<tr>
<td>graphs/tables/logarithmic</td>
<td>n/a</td>
</tr>
<tr>
<td>see for test information</td>
<td>appendix C</td>
</tr>
</tbody>
</table>

Pmacct is a network tool that gathers and aggregates IP traffic. Traffic is aggregated into a Pmacct defined type of flow. The gathered data is stored in an in-memory table whose content can be retrieved by a client program via a local stream-oriented connection. Just recently NetFlow support was added to the tool, this function has however not been tested by the authors.

The installation of Pmacct is quite easy; it is a decent tool and can easily be deployed in a network.

### 6.4 flow-tools

<table>
<thead>
<tr>
<th>Role</th>
<th>collector and analyzer</th>
</tr>
</thead>
<tbody>
<tr>
<td>tested version</td>
<td>0.68, May 11, 2005</td>
</tr>
</tbody>
</table>
Flow-tools are a collection of command line programs used to collect, process, and generate reports from NetFlow data. These tools can be used together on a single system or be distributed to multiple systems for large deployments. The latter makes Flow-tools very scalable.

Flow-tools is totally text-based and comes with a Perl and Python interface. Its library provides an API for development of custom applications for NetFlow export versions 1, 5 and 6. Support for NetFlow version 9 is still in development and should be finished soon. The set of tools can also be used together with third party scripts for presenting simple text-based statistics on a website. When troubleshooting it best to consult the good support on Flow-tools mailing-list.

### 6.5 nTop

Role: collector and analyzer
Tested version: 3.1, December 21, 2004
Developer(s): Luca Deri
URL: [http://www.ntop.org/](http://www.ntop.org/)
License type: GNU GPL
Support & documentation: good
Dependencies: PHP, Apache, Libpcap
OS: POSIX, win32
Flow format: NetFlow versions 1, 5 and 9, nFlow, sFlow
Real-time/legacy: yes/yes
Graphs/tables/logarithmic: yes/no/no

nTop is a network traffic analyzer that offers the possibility to monitor network usage, similar to what the popular Unix command “top” does. nTop is based on Libpcap and it has been written in a portable way in order to virtually run on every Unix platform. It even runs on Win32. nTop combines a collector and analyzer in one. It even contains a built in probe, which can be activated.

Overall nTop tries to display way too much information on some pages (e.g. the RRD-plugin page and the primary page with traffic stats). The pages do contain a lot of useful information, but all the statistics just overflow the user.

CPU usage was low during tests. This was mainly caused by the small amount of data that passed the test machine. There is a lot of documentation available for nTop.
Most simple problems are solved in the large community FAQ. Also a mailing list is available for more difficult problems.

### 6.6 Cflowd

<table>
<thead>
<tr>
<th>role</th>
<th>collector and analyzer</th>
</tr>
</thead>
<tbody>
<tr>
<td>developer(s)</td>
<td>CAIDA</td>
</tr>
<tr>
<td>recent version</td>
<td>v2.1, October 24, 2000</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://www.caida.org/tools/measurement/cflowd/">http://www.caida.org/tools/measurement/cflowd/</a></td>
</tr>
<tr>
<td>license type</td>
<td>GNU General Public License, Version 2</td>
</tr>
<tr>
<td>support &amp; documentation</td>
<td>no support and average documentation</td>
</tr>
<tr>
<td>OS</td>
<td>Linux</td>
</tr>
<tr>
<td>flow format</td>
<td>NetFlow version 1 and 5</td>
</tr>
<tr>
<td>aggregation</td>
<td>no</td>
</tr>
<tr>
<td>real-time/legacy</td>
<td>yes/yes</td>
</tr>
<tr>
<td>graphs/tables/logarithmic</td>
<td>yes/yes/no</td>
</tr>
</tbody>
</table>

Cflowd is CAIDA’s effort for developing a NetFlow measurement tool. The tool is not very scalable; measuring more then one router at the same time could bring the system down. In October 2000 CAIDA has stopped supporting Cflowd. CAIDA’s website states to use other tools.

The authors were unable to install this program because the dependency arts++ could not be installed. The problem arose because arts++ has to be compiled with one specific version of gcc, see also appendix G.

### 6.7 Stager

<table>
<thead>
<tr>
<th>role</th>
<th>collector and analyzer</th>
</tr>
</thead>
<tbody>
<tr>
<td>tested version</td>
<td>1.2.4, June 14, 2005</td>
</tr>
<tr>
<td>developer(s)</td>
<td>Arne Øslebø, Andreas Åkre Solberg and Espen Breivik from Uninett</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://software.uninett.no/stager/">http://software.uninett.no/stager/</a></td>
</tr>
<tr>
<td>license type</td>
<td>GPL</td>
</tr>
<tr>
<td>support &amp; documentation</td>
<td>good</td>
</tr>
<tr>
<td>dependencies</td>
<td>Flow-Tools, PostgreSQL, Perl(10 modules), PHP, Apache, PEAR::DB, JpGraph, Smarty</td>
</tr>
<tr>
<td>OS</td>
<td>Linux, FreeBSD, MacOS X</td>
</tr>
<tr>
<td>flow format</td>
<td>NetFlow version 5 (version 9 in development)</td>
</tr>
<tr>
<td>real-time/legacy</td>
<td>yes/yes</td>
</tr>
<tr>
<td>graphs/tables/logarithmic</td>
<td>yes/yes</td>
</tr>
<tr>
<td>see for test information</td>
<td>appendix E</td>
</tr>
</tbody>
</table>

Stager is an analyzer, which displays network data to users. Stager consists of two main components, the so-called frontend and backend. The latter, driven on Perl, is responsible for storing and aggregating flow-data. Currently three modules exist for the backend:

---

4 Cooperative Association for Internet Data Analysis. Caida has also developed NeTraMet, the RTFM measurement tool. See Appendix A.
NetFlow, MPing and SNMP. These backend modules rely on third party tools like flow-tools. The frontend is a web application that is responsible for presenting the gathered statistics to the user.

The NetFlow module is responsible for handling all flow information. MPing performs latency measurements on the network and the SNMP module speaks for itself. Stager can automatically detect new measurement points and reconfigure itself. Besides this, Stager is highly scalable and suitable for advanced high speed networks. Although still in development and IPFIX or NetFlow version 9 are not yet supported, Stager is very promising. It is one of the better solutions available and support is very good.

### 6.8 Cacti

<table>
<thead>
<tr>
<th>role</th>
<th>analyzer</th>
</tr>
</thead>
<tbody>
<tr>
<td>tested version</td>
<td>0.8.6d, April 26, 2005</td>
</tr>
<tr>
<td>developer(s)</td>
<td>Ian Berry</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://www.cacti.net">http://www.cacti.net</a></td>
</tr>
<tr>
<td>license type</td>
<td>GPL</td>
</tr>
<tr>
<td>support &amp; documentation</td>
<td>good</td>
</tr>
<tr>
<td>dependencies</td>
<td>PHP, MySQL, Apache, Net-SNMP, RRDtool, Perl</td>
</tr>
<tr>
<td>OS</td>
<td>Linux, Windows</td>
</tr>
<tr>
<td>flow format</td>
<td>not applicable</td>
</tr>
<tr>
<td>real-time/legacy</td>
<td>yes/yes</td>
</tr>
<tr>
<td>graphs/tables/logarithmic</td>
<td>yes/yes/yes</td>
</tr>
<tr>
<td>see for test information</td>
<td>appendix F</td>
</tr>
</tbody>
</table>

Cacti is a complete front-end to RRDTool. The front-end is completely PHP driven. Along with being able to maintain graphs, different data sources, and round robin databases, Cacti handles the data gathering itself. Cacti can easily use third-party scripts for polling. On the Cacti website many of these scripts can be found.

The tool is however not primarily designed for working with flow-related data. When doing so, flow-measurements should be hooked up by third-part scripts. If a measurement tool stores it data in a RRD, Cacti can use this RRD to present the statistic to the user.

Most surprising is Cacti’s feature to zoom into graphs. The user can select an area within a graph and zooms into the selected time domain. This feature works very smooth.

When monitoring a large network navigating the frontend can become rather complex. The graphs created by RRDTool are very clear.

Concluding, Cacti is a very flexible tool. Although Cacti is not specifically designed for working with flow-data, all kinds of measurements can be hooked up to it. A flow-extension would make Cacti even more interesting then it already is.

### 6.9 Flowscan+

Role analyzer

tested version this tool was not tested

developer(s) Korea Institute of Science & Technology Information (KISTI)

URL http://cosmos.kaist.ac.kr/~navihp/

license type GNU General Public License
support & documentation  bad, some documentation is in Korean language.
dependencies        Cflowd
OS                    Linux
flow format
real-time/legacy      yes/yes
graphs/tables/logarithmic yes/no/no
see for test information appendix G

In 2001 KISTI funded a project to survey several flow measurement tools. After this survey they had put effort in improving the Flowscan tool which was earlier developed by Dave Plonka. The project, which ended in December 2001, resulted in Flowscan+.

Flowscan+ relies on Tobi Oetiker’s RRDtool for graphing and CAIDA’s cflowd for gathering flow data.

Although Flowscan+ is used on several locations it is not desirable to use this tool, or its predecessor. It relies on cflowd, which is not supported by CAIDA anymore. Their website advises to use Flow-tools instead. Most Flowscan+’s components, except RRDtool, lack support and future development. Future support for NetFlow version 9 or IPFIX can not be expected. The authors were unable to get Flowscan+ working [FLW00].

6.10 Pmacct-fe

role         analyzer
tested version 0.1, April 20, 2005
developer(s) Paolo Lucente
URL          http://www.ba.cnr.it/~paolo/pmacct-fe/
license type GNU General Public License, Version 2
support & documentation good
dependencies PHP, Apache, Horde, JpGraph, Pear, SQL
OS           POSIX
flow format  n/a
real-time/legacy yes/yes
graphs/tables/logarithmic yes/yes/no
see for test information appendix H

Pmacct-fe is a network traffic analyzer implemented in PHP. It uses Horde for user authentication, JpGraph to plot graphs and relies on a PostgreSQL database to read collected network data. The only probe that fills the database conform to Pmacct-fe’s requirements is the Pmacct probe. It is a little effort to configure programs to fill the database with a compatible format.

All data is presented through Horde, which also handles the user authentication. In Horde statistics can be selected for different periods of time. After making this selection, the result is presented to the user. Navigation in Pmacct-fe is very naturally.

In the main table of Pmacct-fe there is however some excess information being displayed.
Chapter 7: Scenarios

The scenarios below give an indication of how flow measurement tools can be deployed in the network. All scenarios are described with SURFnet in mind. Please note that this is just a small selection of all possible scenarios.

Scenario 1a: Measuring throughput between the different points of presence (network manager)

Currently throughput measurement is done separately for all points of presence. This gives a good indication of the backbone utilization of e.g. the University of Twente. On top of this, flow measurement makes it possible to see the correlation among the points of presence. Per connection it can be pointed out where data comes from and goes to. It is interesting to see that for example 40% of the University of Twente’s outgoing traffic goes to the University of Groningen. This could be due to known reasons, or may be not.

Concluding, all traffic flows among the different points of present can be monitored. Also a detailed overview of the usage of the transatlantic links can be presented. For presenting this information the following tools can be used.

probe NProbe exporting NetFlow version 5 with a sampling rate of 100 running on Linux compiled with the PF_Ring5. collector Flow-tools set to receive NetFlow version 5. analyzer Cacti which periodically runs flow-tools to analyze the NetFlow datagrams and populated an RRD.

The probes must be directly connected to the monitoring port of the point of presence. This requires one probe per point of present. All probes export their NetFlow packets to one of the three collectors in the networks. To make sure all NetFlow packets can be

5 IPFIX or NetFlow version 9 is preferred but not yet available for this configuration.
handled correctly functionality is distributed over three systems. The collectors can drop their data periodically since the analyzer will do the storage in this scenario.

Cacti, which functions as the web-interface, polls the collectors every minute. It only requests the information it needs and populates its round robin databases that are fixed sized. One database is needed for every possible connection among the points of present. Ten points of presences will result in ninety databases.

For a clear overview Cacti allows the user to select one of the locations within the network. All throughput statistics towards the other observation points in the network will be returned.

The advantage of this configuration is the requirement of only little storage. All detailed flow information is thrown away at the collector side. This can also be seen as a disadvantage because the possibilities provided by flow measurements are not fully utilized.

See appendix F for how this configuration with Cacti can easily be expanded with SNMP and QoS measurements.

**Scenario 1b: An alternative to scenario 1a (network manager)**

The scenario as given in 1a can easily be extended to provide more detail on flows. On top of NProbe and Flow-tools Cacti can be deployed. Stager backends have to be installed on the systems where first only Flow-tools was operating. The best option is to let all three backends connect to a single database, which will be storing all the gathered information. Take note that the database system will require lots of resources for storage.

Instead of using Cacti, the Stager frontend is used for presentation. This configuration will result in much more detailed results. Instead of just taking a looking where flows are traversing through the network, this can also differentiate in the different protocols that are used. See appendix E for how such a configuration will look like.

**Scenario 2: Sub-domain service usage (interested end-user)**

To present the type of traffic is traversing through the router that connects the end-user to the SURFnet backbone, just a single probe has to be deployed. The utilization of services like bit-torrent, ftp, edonkey, skype and many more can be monitored this way.

<table>
<thead>
<tr>
<th>probe</th>
<th>NProbe exporting NetFlow version 5 with a sampling rate of 100 running on Linux compiled with the PF_Ring(^6).</th>
</tr>
</thead>
<tbody>
<tr>
<td>analyzer</td>
<td>Third party PHP-script.</td>
</tr>
</tbody>
</table>

The probe must be directly connected to the monitoring port of the router. The collector and analyzer can run on the same system, which is preferably directly connected to the probe for reducing the network load.

The analyzer is a third-party PHP script that performs lookups in the NetFlow data by running flow-tools. The website written in PHP just presents usage figures in simple tables. When the analyzer just has to present the usage figures of the last hour or day, old data can periodically be removed.

\(^6\) IPFIX or NetFlow version 9 is preferred but not yet available for this configuration.
**Scenario 3: Performing simultaneous QoS and flow measurement (network manager & sub-domain managers)**

This scenario describes how QoS and flow measurements can be used in conjunction with each other. Passive measurements like NetFlow methods cannot provide the kind of controllable experiments that can be achieved with active measurements. Data that is required to generate simple statistics like latency, jitter and loss is removed when aggregation to flow datagrams is performed.

Deploying QoS measurement points (QoSMP) in the network provides a method for performing all kinds of activities. *Figure 7.1* shows how this could be implemented.

The QoSMPs and QoS collector are responsible for the recording and storage of the QoS statistics. This data can be used by the web server to create graphs and present this result to the user. It might also be interesting to deploy QoSMPs closer to the end-users for getting a better view of what they experience.

Although QoS was not the main priority in the performed research, the authors tested some tools that can be seen as good additions to flow measurement. The following tools could be deployed on the QoSMPs/QoS collector.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SmokePing</td>
<td>SmokePing can measure, store and display latency, latency distribution and packet loss. It stores data in RRD files. Third party scripts can be hooked up. Echoping and FPing are most commonly used in combination with SmokePing.</td>
</tr>
<tr>
<td>Multicast beacon</td>
<td>Multicast Beacon is a diagnostic tool written in Perl which uses the RTP protocol to provide useful statistics and diagnostic information about a given group's connectivity characteristics.</td>
</tr>
<tr>
<td></td>
<td><a href="http://dast.nlanr.net/Projects/Beacon/">http://dast.nlanr.net/Projects/Beacon/</a></td>
</tr>
<tr>
<td>Mping</td>
<td>Multi-ping (Mping) is a system for collecting packet delay and loss statistics in a TCP/IP network using ICMP echo. It can be used in conjunction with Stager, see annex E.</td>
</tr>
<tr>
<td></td>
<td><a href="http://mping.uninett.no/">http://mping.uninett.no/</a></td>
</tr>
</tbody>
</table>
Figure 7.2: Example architecture showing how active QoS measurements can be used in conjunction with flow measurement.
Appendix A: Extra tool overview

This appendix shortly describes the tools that did not survive the primary selection. The selection was performed conform to the requirements as stated in chapter 5. Main reasons for rejection are poor scalability and the lack of support for IPFIX or NetFlow version 9. The use of this appendix is to give an overview of most open source flow measurement projects currently available. Table 1 can be used for a fast overview.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Probe</th>
<th>Collector</th>
<th>Analyzer</th>
<th>Promising</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flowprobe</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pf-flowd</td>
<td>yes</td>
<td></td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>NG NetFlow</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IpCad</td>
<td>yes</td>
<td></td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>TCPflow</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pcNetFlow</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sflow probe</td>
<td>yes</td>
<td></td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>Sflow toolkit</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>Netramet</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>NNFC</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSSP flow2rrd</td>
<td>yes</td>
<td></td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>Flowd</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neye</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NFDC</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NetFlow2XML</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NetFlow2MySQL</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NFDUMP</td>
<td>yes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F.L.A.V.I.O.</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fluxoscope</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Netflow Monitor</td>
<td>yes</td>
<td>yes</td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>EHNT</td>
<td>yes</td>
<td></td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>Argus</td>
<td></td>
<td></td>
<td></td>
<td>yes</td>
</tr>
</tbody>
</table>

Table 1: overview of the tools that were not tested.

Flowprobe

role: probe
developer(s): Slava Astashonok
recent version: 1.1, January 29, 2005
URL: http://sourceforge.net/projects/fprobe
OS: POSIX
flow format: NetFlow version 1, 5 and 7

7 Linux/BSD/UNIX-like OSes
Fprobe and Fprobe-ulog are NetFlow probes. Fprobe is based on Libpcap, the tool collects network traffic data and emits it as NetFlow PDUs towards the specified collector. Fprobe-ulog is based on Libipulog, and is used as a fork of fprobe.

There is only little documentation available for flowprobe.

**PFFlowd**

role probe  
developer(s) Damien Miller  
recent version 0.6, September 6, 2004  
URL [http://www.mindrot.org/pfflowd.html](http://www.mindrot.org/pfflowd.html)  
OS OpenBSD  
flow format NetFlow version 5

PFFlowd acts as a NetFlow probe. PFFlowd converts OpenBSD PF status messages, sent through the pfsync interface, to NetFlow PDUs. These datagrams are eventually sent to the collector.

Utilizing the OpenBSD standard packet filter infrastructure means that flow tracking is very fast and accurate. Using BSD, which is made with issues like packet capture in mind, makes PFFlowd a very efficient tool. Comparable speeds can however be achieved on POSIX by using Luca Deri’s PF_Ring; also see appendix I.

**Ng_netflow**

role probe  
developer(s) Gleb Smirnoff  
recent version 0.2.5, August 27, 2004  
OS FreeBSD  
flow format NetFlow version 1 and 5

Ng_netflow, a probe, is actually a netgraph kernel module that monitors all traffic and aggregates to NetFlow PDUs. The authors have not tested ng_netflow because of the lack of available information.

**IPCAD**

role probe  
developer(s) Lev Walkin  
recent version 3.6.6, June 09, 2005  
URL [http://freshmeat.net/projects/ipcad/](http://freshmeat.net/projects/ipcad/)  
OS POSIX, Solaris, MacOS X  
flow format NetFlow version 5, rsh

The IP Cisco Accounting Daemon (IPCAD) listens to traffic on specified interfaces and exports the collected information using rsh or NetFlow. It uses raw Berkeley Packet Filter (BPF) devices, PCAP library, divert, tee, or Linux iptables' ULOG & libipq packet sources. By using this diversity, multiple platforms and situations are supported. IPCAD is documented very well.
Tcpflow
Role: probe
developer(s): Jeremy Elson
recent version: 0.21, August 7, 2003
URL: http://www.circlemud.org/~jelson/software/tcpflow/
OS: POSIX, Mac OS X, Solaris
flow format: text files

Tcpflow captures TCP traffic into text files. These are stored in a way that is convenient for protocol analysis or debugging. Tcpflow reconstructs the actual data streams and stores each flow in a separate file for later analysis. Obviously this does not match the requirements.

PcNetFlow
role: probe
developer(s): Fujii Satoshi
recent version: 0.1e-92
URL: http://cluster19.aist-nara.ac.jp/public/#pcNetFlow
OS: Linux, FreeBSD
flow format: NetFlow version 5

PcNetFlow is only capable of exporting NetFlow version 5. It does not strictly follow the flow definition as stated by Cisco. This makes PcNetFlow incompatible with most collectors. Usage of PcNetFlow is not recommended by its developer.

Sflow probe
role: probe
developer(s): Inmon corporation
recent version: unknown
URL: http://www.inmon.com/products/probes.php
OS: Linux
flow format: sFlow

The Inmon sFlow probe is capable of continuously monitoring traffic flows with speeds up to 1 Gigabit. Therefore it can provide accurate and complete measurements when reliable measurements are most needed. When deployed on a small network, the probe can monitor several routers or switches at the same time. This probe can easily be used in combination with the Sflow toolkit stated below.

Sflow toolkit
role: collector
developer(s): Neil McKee from Inmon corporation
recent version: 3.8
URL: http://www.inmon.com/technology/sflowTools.php
OS: Linux, Solaris, Windows
flow format: sFlow, NetFlow, text format
The sFlow toolkit provides command line utilities and scripts for analyzing sFlow data. SFlow’s core component is its command line utility. It provides text based output that can be used in scripts to provide customized analysis and reporting. SFlow can easily be combined with tools such as MRTG, RRD-Tool or cacti.

Besides sFlow it also has legacy support for NetFlow on which the toolset originally was based. Inmon decided to switch over to sFlow due to reasons that are also stated in chapter 4.4.

The authors’ opinion is that SFlow is a very good alternative for NetFlow. SFlow’s toolkit is one of the few open source sFlow collectors that are around. If SFlow reporting is required, SFlow should be considered.

**Netramet**

<table>
<thead>
<tr>
<th>role</th>
<th>probe, collector analyzer</th>
</tr>
</thead>
<tbody>
<tr>
<td>developer(s)</td>
<td>Nevil Brownlee</td>
</tr>
<tr>
<td>recent version</td>
<td>5.0 and 4.4</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://www.caida.org/tools/measurement/netramet/dist.xml">http://www.caida.org/tools/measurement/netramet/dist.xml</a></td>
</tr>
<tr>
<td>OS</td>
<td>DOS, UNIX</td>
</tr>
<tr>
<td>flow format</td>
<td>RTFM</td>
</tr>
<tr>
<td>aggregation</td>
<td>none</td>
</tr>
<tr>
<td>real-time/legacy</td>
<td>yes/yes</td>
</tr>
<tr>
<td>graphs/tables/logarithmic</td>
<td>no/yes/no</td>
</tr>
</tbody>
</table>

Netramet is an all-round tool for packet capturing which works with the RTFM architecture. All services from capture to presentation are provided. It is developed and still supported by the University of Auckland. Netramet is basically a good and robust tool that was widely used in the past.

Overall support for RTFM is just little and it is more logic to use a standard with wider support. RTFM can however still be used and will probably perform without any major problems. A clear installation manual is available on the website, which also provides good information on RTFM.

**NNFC**

<table>
<thead>
<tr>
<th>role</th>
<th>collector</th>
</tr>
</thead>
<tbody>
<tr>
<td>developer(s)</td>
<td>Andrew Klyachkin</td>
</tr>
<tr>
<td>recent version</td>
<td>0.8.1 (beta), October 28, 2004</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://sourceforge.net/projects/nnfc/">http://sourceforge.net/projects/nnfc/</a></td>
</tr>
<tr>
<td>OS</td>
<td>POSIX, MacOS X</td>
</tr>
<tr>
<td>flow format</td>
<td>NetFlow version 1, 5 and 7</td>
</tr>
<tr>
<td>aggregation</td>
<td>no</td>
</tr>
</tbody>
</table>

The New NetFlow Collector (NNFC) aims to be a total POSIX-compliant collector. NNFC receives, processes and stores NetFlow version 1, 5 and 7 PDUs. The received data can be stored in text-files, MySQL or PostgreSQL by using its module interface.
There is not much information supplied on the website. Updates are not recent and it seems not much support is offered.

**OSSP flow2rrd**

- role: collector
- developer(s): Ralf S. Engelschall
- recent version: 0.9.1, December 26, 2004
- OS: POSIX, MacOS X
- flow format: NetFlow (which version is unknown)
- aggregation: yes

Flow2RRD collects and processes NetFlow PDUs; afterwards the information is stored in a Round Robin Database (RRD). RRD-Tools can also be used for graphing purposes; could be good in combination with Stager. The first stable release is expected soon.

**FLOWD**

- role: collector
- developer(s): Damien Miller
- recent version: 0.8.5, May 14, 2005
- URL: [http://www.mindrot.org/flowd.html](http://www.mindrot.org/flowd.html)
- OS: POSIX
- flow format: NetFlow versions 1, 5, 7 and 9
- aggregation: No

FlowD is an effort by Damien Miller for making a small, fast and secure collector. The only functionality it provides is receiving NetFlow PDUs and writing them to disk. It does not provide any storage methods. A scenario could be using Flowd c in combination with scripts to perform simple measurements.

**NEye**

- role: collector
- developer(s): Gilberto Persico
- recent version: 1.0.1, February 6, 2005
- URL: [http://neye.unsupported.info/](http://neye.unsupported.info/)
  [http://osx.freshmeat.net/projects/neye/](http://osx.freshmeat.net/projects/neye/)
- OS: POSIX, Mac OS X, Solaris
- flow format: NetFlow versions 1 and 5
- aggregation: no

NEye’s basic functionality is the collecting of NetFlow version 5 PDUs. It is capable of logging flow information into ASCII, MySQL and SQLite databases. NEye is written in the C language and due to scalability it makes use of POSIX threads if available. Its storage methods are not suitable for monitoring large networks.
NFDC
role: collector
developer(s): Unknown
recent version: 1.4, February 12, 2005
URL: http://nfdc.sourceforge.net/
OS: POSIX
flow format: NetFlow (which version is unknown)
aggregation: no

NFDC is short for NetFlow Datagram Collector and still in development. It stores the gathered flow data in a PostgreSQL database. For a better performance it buffers incoming data and periodically writes to the database in bursts. NFDC has successfully been tested on a network generating over $3 \times 10^6$ flows every 24 hours.

NetFlow2MySQL
role: collector
developer(s): FUJII Satoshi
recent version: 0.24, April 9, 2005
URL: http://cluster19.aist-nara.ac.jp/public/#NetFlow2MySQL
OS: Linux, FreeBSD
flow format: NetFlow versions 5 and 9
aggregation: no

NetFlow2MySQL collects and stores the flow data in a MySQL database. MySQL has, for Linux, a limited table size of 4GB [SQL05]. This could be insufficient. NetFlow2MySQL does not perform aggregation on its database.

NetFlow2XML
role: collector
developer(s): FUJII Satoshi
recent version: 0.2, April 9, 2005
URL: http://cluster19.aist-nara.ac.jp/public/#netflow2MySQL
OS: POSIX
flow format: NetFlow versions 5 and 9
aggregation: no

NetFlow2XML comes from the same developer as NetFlow2MySQL. This variant stores the flow data into XML files, allowing analysis of traffic with XML parser or XMLDB.

NFDUMP
role: collector
developer(s): Peter Haag
recent version: 1.31, May 25, 2005
URL: http://nfdump.sourceforge.net/
OS: POSIX, Solaris, MacOS X
NFDump, short for NetFlow dump, is a collection of tools that work from the command line. NCapd is used for sniffing the interface. NetFlow version 7 PDUs are transparently transformed to NetFlow version 5. The set of tools does not perform any aggregation, but provides a method for filtering the acquired data.

**F.L.A.V.I.O**

<table>
<thead>
<tr>
<th>Role</th>
<th>collector and analyzer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer(s)</td>
<td>Flavio Villanustre</td>
</tr>
<tr>
<td>Recent version</td>
<td>2.0.0, July 27, 2002</td>
</tr>
<tr>
<td>OS</td>
<td>runs on perl</td>
</tr>
<tr>
<td>Flow format</td>
<td>NetFlow (which version is unknown)</td>
</tr>
<tr>
<td>Aggregation</td>
<td>no</td>
</tr>
<tr>
<td>Real-time/Legacy</td>
<td>unknown/yes</td>
</tr>
<tr>
<td>Graphs/Tables/Logarithmic</td>
<td>yes/yes/unknown</td>
</tr>
</tbody>
</table>

F.L.A.V.I.O runs entirely on Perl and thus can be deployed on many different kinds of systems. It has been designed in such a way that new functionality can easily be hooked up. The last update on F.L.A.V.I.O already dates from 2002. Furthermore it uses a MySQL backend which is only suitable for relatively small network.

**NetFlow Monitor**

<table>
<thead>
<tr>
<th>Role</th>
<th>collector and analyzer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer(s)</td>
<td>Jan Nejman</td>
</tr>
<tr>
<td>Recent version</td>
<td>2.1, 2003</td>
</tr>
<tr>
<td>OS</td>
<td>Linux</td>
</tr>
<tr>
<td>Flow format</td>
<td>NetFlow versions 1, 5, 6 an 7</td>
</tr>
<tr>
<td>Aggregation</td>
<td>yes</td>
</tr>
<tr>
<td>Real-time/Legacy</td>
<td>yes/yes</td>
</tr>
<tr>
<td>Graphs/Tables/Logarithmic</td>
<td>yes/yes/no</td>
</tr>
</tbody>
</table>

The NetFlow Monitor is a collector-analyzer combination which has a distributed architecture with a maximum processing capacity of 40 gigabit/s. CESNET\(^8\) is currently using the NetFlow Monitor for measuring their network. Navigation in NetFlow Monitor is very unnatural and requires some improvement, but it able to create clear graphs. Upcoming support for NetFlow version 9 or IPFIX has not been announced.

**Fluxoscope**

<table>
<thead>
<tr>
<th>Role</th>
<th>collector and analyzer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer(s)</td>
<td>SWITCH</td>
</tr>
</tbody>
</table>

---

\(^8\) CESNET is the Czech research network – [http://www.ces.net](http://www.ces.net)
Fluxoscope, developed by SWITCH\textsuperscript{9}, was developed back in 1997 for accounting one of their transatlantic links. Also some minor traffic analysis was performed with this tool. Nowadays one of Fluxoscope’s purposes is monitoring network activity at CERN\textsuperscript{10}.

The Common Lisp source code is available upon request. Not that the system requires extensive configuration and lacks any documentation. It relies on RRD-Tool for storage and generating graphs.

**Flowc**

<table>
<thead>
<tr>
<th>role</th>
<th>collector and analyzer</th>
</tr>
</thead>
<tbody>
<tr>
<td>developer(s)</td>
<td>Taras Shevchenko</td>
</tr>
<tr>
<td>recent version</td>
<td>1.5rc3, March 9, 2004</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://netacad.kiev.ua/flowc/index.php?id=1">http://netacad.kiev.ua/flowc/index.php?id=1</a></td>
</tr>
<tr>
<td>OS</td>
<td>Linux, *BSD</td>
</tr>
<tr>
<td>flow format</td>
<td>NetFlow version 5</td>
</tr>
<tr>
<td>aggregation</td>
<td>yes</td>
</tr>
<tr>
<td>real-time/legacy</td>
<td>yes/yes</td>
</tr>
<tr>
<td>graphs/tables/logarithmic</td>
<td>yes/yes/yes</td>
</tr>
</tbody>
</table>

Some ISPs and companies are using Flowc for analysis, billing and network troubleshooting. Flowc is well documented and provides good support. It provides many features for presenting flow data. A drawback is that only NetFlow version 5 is supported and that it relies on MySQL. Currently the developer is working on PostgreSQL support, which is a promising progress. Flowc should be considered when using NetFlow version 5.

**EHNT**

<table>
<thead>
<tr>
<th>role</th>
<th>collector &amp; analyzer</th>
</tr>
</thead>
<tbody>
<tr>
<td>developer(s)</td>
<td>Craig Small and Nik Weidenbacher</td>
</tr>
<tr>
<td>recent version</td>
<td>0.4, July 28, 2003</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://ehnt.sourceforge.net/">http://ehnt.sourceforge.net/</a></td>
</tr>
<tr>
<td>OS</td>
<td>POSIX</td>
</tr>
<tr>
<td>flow format</td>
<td>NetFlow version 5</td>
</tr>
<tr>
<td>aggregation</td>
<td>yes</td>
</tr>
<tr>
<td>real-time/legacy</td>
<td>yes/yes</td>
</tr>
<tr>
<td>graphs/tables/logarithmic</td>
<td>no/yes/no</td>
</tr>
</tbody>
</table>

\textsuperscript{9} The Swiss Educational and Research Network.

\textsuperscript{10} CERN is the world’s largest particle physics laboratory.
Extreme Happy NetFlow Tool (EHNT) can only generate reports based on NetFlow version 5. Everything works from the console and output is totally text based. Little information is available about this tool. It seems to work with raw NetFlow files.

**Argus**

- **role**: collector and analyzer
- **developer(s)**: Argus role collector and analyzer developer(s)
- **recent version**: 2.0.6, May 10, 2004
- **OS**: Linux, Solaris, *BSD and MacOS
- **flow format**: NetFlow (which version is unknown)
- **aggregation**: no
- **real-time/legacy**: yes/yes
- **graphs/tables/logarithmic**: no/yes/no

The Argus project started at the SEI\(^\text{11}\) and became open source in 1996. It is unknown which NetFlow versions are supported, most likely only version 5. Collected NetFlow PDUs are stored in a special RA-format. These files can be analyzed by tools that come with the Argus package.

Currently no effort has been put in developing a GUI for Argus; everything is still text based. There are future plans for integrating a database in Argus. Although the last version dates from 2004, the mailing-list is very active and provides good support.

---

\(^{11}\) Carnegie Mellon's Software Engineering Institute
Appendix B: Nprobe test report

Nprobe, an easy to use network probe, comes from the same developer as nTop, Luca Deri. He also created the PF_Ring, which performs an efficient method for packet capturing. This PF_Ring can be used in conjunction with nprobe. A more detailed explanation of the PF_Ring is available in appendix I.

Nprobe was created as a light network probe, which can send network information towards a central traffic analysis console such as nTop. The program was made to have a limited memory footprint (less that 2 MB of memory regardless of the network size) and be CPU savvy. So it would be able to run on many platforms, even older ones.

The probe has multiple possibilities for output information and is also easily configurable by the user. Eventually it will provide flow statistics to the user.

Conformance to the previous selection

Nprobe has got all the features the authors found important in the theoretical examination of the tool. It performed as a stable tool which has excellent capture possibilities.

Installation

A demo-copy of nProbe was used for testing the program. It is provided on the nprobe website, and an installation is not necessary. The precompiled file runs without any extra work.

Running nprobe

Nprobe was run with the following command

```
./nprobe-linux-static -i eth0
```

The interface eth0 is the interface the packets are received. Another possibility of nprobe is to create a file where it will store the recognized flows this feat can be accomplish by running nprobe as follows:

```
./nprobe-linux-static -D b -P /home/probe/output/nprobe/ -i eth0
```

Problems encountered

No problems were encountered while testing nprobe.

Possibilities for improvements

Nprobe is pretty complete with all functions a probe needs and there are no real suggestions for improvements.

Remarkable features

Most remarkable about nprobe is the wide variety of supported formats and the great efficiency with which the packet capture occurs in combination with the PF_Ring.
**Scalability**

Nprobe should be able to work on low-end machines and has been tested by Luca Deri and according to his results a 1.7 GHz p4 should be able to monitor a 1 gigabit connection.

**Ease of use**

The probe is easy to use and is a lightweight program that is easily installable and usable.

**Probe specific**

Nprobe supports the NetFlow v5, v9, nFlow and sFlow formats. Another output format of nprobe is to create dump files of the flows. Future support for IPFIX has been announced.

The capture percentage was not tested because of the slow connection that was available. Because of the low amount of traffic that passed by the test machine\(^\text{12}\) there was no chance to test the capture efficiency of the tool. It should be able to monitor a 1Gbit connection when running with the PF_Ring.

The CPU load was not tested because of the slow connection that was available. On the slightly loaded interface it never got above 1% of CPU usage on a p2 400MHz. Note that this is not representative for the capabilities of nprobe.

---

\(^{12}\) A Pentium 2 400 with 256 MB ram.
Appendix C: Pmacct test report

Pmacct is a network tool that gathers and aggregates IP traffic. Aggregation works by describing the packets in flows. In other words, Pmacct acts as a flow-probe. The gathered data is stored in an in-memory table whose content can be retrieved by a client program via a local stream-oriented connection. Paolo Lucente, the developer, comments that Pmacct is still under strong development. User experiences show that Pmacct runs stable though.

Pmacct is a project of Paolo Lucente from the University of Bari in Italy. He created a probe called Pmacct, which could aggregate network traffic into flows.

Conformance to the previous selection

Based on the previous primary selection, Pmacct almost entirely performed like expected. One major drawback is that Pmacct currently does not support the exportation of NetFlow compatible UDP packets. This is necessary for, easily, sending the aggregated flow information to a central collector.

Installation

The installation of Pmacct is quite easy. It is just a matter of downloading the source code, compiling it and the user is ready to run. In the tar-ball the file ‘config-keys’ was missing, therefore the authors had to retrieve it from another version.

For excessive configuration some more knowledge is required. Pmacct works with configuration files that can be loaded when starting the monitoring. The syntax of these configuration files is however quite naturally and easy to understand.

Although while installing an improved version the user should be careful to remove any existing Pmacct files, otherwise some plugins might not work correctly. Pmacct was later also tested in the versions 0.8.6 and 0.8.7.

Running Pmacct

Despite the NetFlow exportability drawback, Pmacct functioned as expected. Several tests up to durations of 25 hours were performed. No problems were encountered during these tests. The output is very basic and gives just the most relevant information that is available when processing flows.

Problems encountered

While enabling the PostgreSQL support some remarkable things arose. While installing the PostgreSQL enabled version the authors had to remove every existing Pmacct file, otherwise the plugin would not work correctly.

Possibilities for improvements

As mentioned before, the lack of the NetFlow export ability makes the probe less all-round and unable to work with any of the other collectors that were tested. Later Pmacct was tested with an analyzer, namely Pmacct-fe. The author of Pmacct is currently working on the probe to create a NetFlow exporting ability.
Scalability

Scalability might become a problem because the probe had primarily got output methods for processing on the same machine.

Ease of use

Pmacct proved easy to install, but is a little harder to configure. Especially when pmacct is used to output flows into a database.

Probe specific

The output formats of Pmacct include:
- Logging to a file on the disk
- Writing into a MySQL or PostgreSQL

Capture percentage for Pmacct was 100 percent on our test machine. This can not be seen as a representative figure, because there was not much traffic passing the test machine.

CPU load of Pmacct was also low, but in regard to nProbe it used more CPU time while working on the same connection.

Options

This listing shows the results of calling the help function of PMACCT. This gives a good overview of its functions.

```bash
Akira # /usr/local/sbin/pmacctd --h
Promiscuous Mode Accounting Daemon, pmacctd 0.8.2
Usage: /usr/local/sbin/pmacctd [-D|-d] [-i interface] [-c primitive[,...]]
[-P plugin[,...]] [filter]
    /usr/local/sbin/pmacctd [-f config_file]
    /usr/local/sbin/pmacctd [-h]

General options:
- h    show this page
- f    configuration file (see also CONFIGKEYS)
- c    [src_mac|dst_mac|vlan|src_host|dst_host|src_net|dst_net|src_port|dst_port|
    prot|tos|src_as|dst_as,sum_host,sum_net,sum_as,sum_port,none]
    aggregation primitives (DEFAULT: src_host)
- D    daemonize
- N    disable promiscuous mode
- n    path to a file containing network definitions
- o    path to a file containing port definitions
- P    [memory|print|mysql|pgsql]
    activate plugin
- d    enable debug
- I    listening interface
- r    read packets from the specified savefile
- S    [auth|mail|daemon|kern|user|local[0-7]]
    syslog facility
- F    write Core Process PID into the specified file
- w    wait for the listening interface to become available

Memory Plugin (-P memory) options:
```
-p  socket for client-server communication (DEFAULT: /tmp/collect.pipe)
-b  number of buckets
-m  number of memory pools
-s  memory pool size

PostgreSQL (-P pgsql)/MySQL (-P mysql) plugin options:
-r  refresh time (in seconds)
-v  [1|2]
    table version

Listing 1: overview of PMACCT options.

Test output
The dump to disc contains no begin or finish time for different flows.
[src_mac|dst_mac|vlan|src_host|dst_host|src_net|dst_net|src_port|dst_port|proto|tos|src_as|dst_as,sum_host,sum_net,sum_as,sum_port,none] are the possible option for the Flow format which pmacct relays from the probe. If they are not selected, they are just zeroed out.

```
# akira# /usr/local/sbin/pmacctd -c src_port,dst_port,src_host,dst_host, proto -P print -i eth0 -r 30000 > test
OK: link type is: 1
SRC IP      DST IP      SRC PRT DST PRT  PROTOCOL   PACKETS  BYTES
130.89.145.41  130.89.13.111  22    3522      tcp        4       160
130.89.157.4   130.89.145.41  6667   1170      tcp       1304    147778
130.89.145.1   224.0.0.13     0      0         pim       1014    38532
130.89.145.41  130.89.13.41   22    3713      tcp        3       120
130.89.145.111 224.0.0.9      0      0         igmp      234     7488
130.89.13.41   130.89.145.41  3738   21        tcp       103     4426
130.89.145.41  130.89.13.111  22    3534      tcp       2851    336984
130.89.145.25  130.89.145.255 137    137       udp       17      1326
130.89.145.129 130.89.145.255 137    137       udp       39      3186
130.89.145.139 130.89.145.112  0     0         icmp      19      1596
130.89.145.41  130.89.175.16  57123  52845     tcp       1       40
```

Listing 2: test output.

ID | SRC MAC | DST MAC | VLAN and TOS were removed from these flows to improve the readability.
Appendix D: NTop test report

NTop is a network traffic analyzer that offers the possibility to monitor the network usage, similar to what the popular Unix command ‘top’ does. NTop is based on Libpcap. It has been written in a portable way in order to run on virtually every Unix platform and on Win32 as well.

Users have access to the statistics with a web browser to navigate through NTop traffic information and get a dump of the network status.

Conformance to the previous selection

NTop consists of a collector and an analyzer in one. In the selection of the collector function of NTop all the claims that were made on the website were fulfilled. The analyzer section was even larger and contained more functionality then the authors had expected. This tool allows an extensive inspection of all the traffic that is present on the network.

Installation

An installation and test environment for nTop was created on a Gentoo system. Before installation it was necessary to emerge several package dependencies for nTop:
- glibc, glibc-devel, gcc, cpp
- gawk
- libtool (1.4+)
- gdbm, gdbm-devel
- libpcap (http://www.tcpdump.org)
- gd, gd-devel
- libpng, libpng-devel

The installation went smoothly and without any problem. Note that if nTop ran as a different user then root, after the installation the access rights to some directories should also change. This is necessary to allow nTop to operate properly.

Running nTop

```
/usr/local/bin/nTop @/home/probe/nTop.conf
/home/probe/nTop.conf:
-P /usr/local/share/nTop
-w 130.89.145.41:3000
-W 130.89.145.41:3001
-u root
-m 130.89.0.0/16
-M
```

Listing 1: The parameters given to nTop when starting.

- **-P** specifies the location of the nTop databases
- **-w** is the place of the standard webserver nTop runs

--

13 nTop also acts as a web server.
14 An installation method for the Gentoo Linux distribution.
-W is the place of the https webserver ran by nTop
-u specifies the user nTop runs under
-m gives the limit of local IP’s.
-M tag is used in order to be able to use multiple probes

NTop automatically starts its own probe on the local machine, this behaviour can be prevented with the ‘--interface none’ switch. All the settings can also be set from the web-based presentation nTop provides.

As specified before, IPv6 support is available in nTop. A ‘P2P Recently Exchanged Files’ option is also available; although this is information that can be derived from the statistics it should not be included. This is partially because of privacy reasons.

In a month long test almost no packets were dropped. There was no real stress test for nTop because the test system has a 100 Megabit network card and nTop should be able to handle this low speed with ease.

Problems encountered

The package worked without any major problems, except for the Admin login. Despite the input of the correct password, it was not recognized or usable in the web based presentation that nTop generates.

After enabling URL_DEBUG in nTop it became visible that something keeps going wrong with the password authentication. This problem was encountered in both Internet Explorer and Firefox 1.0.1. The authors posted this problem on the mailing list and after a few tries the problem was resolved. The solution was to remove both nTop_pw.db files and input a new password.

There is an incomplete legend with some of the pie charts, only half of the colours in the chart are named in the legend.

Another strange error occurred when sorting the Network Traffic of All Protocols by IPSEC, OSPF or IGMP. This page does not show up and nTop comments that the browser does not support frames. It appears to be the case that any protocol beyond the 16th is being invoked wrongly, which causes nTop to display an error because the page does not exist.

Error messages in nTop do not contain the concerning interface. Very likely they are coming from the softflowd probe, which is another probe that was tested in combination with the nTop system. Because after it was shut down the errors did not occur anymore.

Another slip-up that was found in the main traffic summary:

---

Listing 2: Errors in nTop

Another slip-up that was found in the main traffic summary:

---

3.0 GB [15,911,918 Pkts]
2.8 GB [2.8 GB Pkts]

Figure 1: nTop slip-up

---

15 The file existed in two different directories and had to be removed twice to enable the acceptance of an new password.
16 Softflowd is an implementation of a probe to monitor passing traffic.
Between the brackets in the IP Traffic category is the amount of data displayed again instead of the total number of packets.

**Possibilities for improvements**

More explanation is needed on the rrdgraph page. Sometimes the user get overflowed with information, this can be improved.

The problems discussed in the ‘Problems encountered’ section are eligible here; these should pose some easy improvements.

**Scalability**

On the test machine\(^7\) nTop keeps using around 6 to 10 percent of memory usage. After running for a month nTop has generated 62 threads and takes possession of 140 MB of Virtual Memory.

NTops CPU usage is also very low on the test system, it uses around 1% of the cpu power on average.

**Ease of use**

There is a lot of documentation available for nTop and most simple problems occur in the large community FAQ that is available. Also an active mailing list is available for more difficult problems. Most of the problems are resolved quickly.

**Collector input methods**

nTop can receive the following flow formats:

- NetFlow v1/v5/v9,
- sFlow
- nFlow
- IPFIX\(^8\)

Besides receiving these formats, nTop also provides the possibility to import TCPdump files.

**Compatibility**

Data received by the collector can only be processed by the built in analyzer, which is able to analyze the stored data. The only available option for analyzing the collected flows with another program is to create dump files. This is because most of the data that is collected is not written away to a file, but kept in memory for immediate analysis.

During the tests some problems occurred when using Softflowd in combination with nTop. Softflowd creates 19.4 gigabyte in ghost traffic and then seems to die. Softflowd reports 42,57 billion packets within a few seconds. It claimed to be pulling 16 Gigabit/s over a 100 Megabit/s card. It does not even fit within the graph scaling of nTop. *(Graph in Figure 2)*

The authors did not find out what caused this erratic behaviour.

---

\(^7\) A Pentium 2 400 with 256 MB ram.

\(^8\) Not confirmed
Storage
NTop makes use of RRD-Tool to display a portion of the graphs. The RRD part is the only persistent storage nTop has. The data presented on the nTop web pages, other than the RRD based graphs, are generated from memory based tables and counters. When nTop restarts, this data is lost. There is a way to backup this data by dumping it into a file, but this backup can only be performed by using an unsupported plugin.

Graphs
NTop offers a wide variety of graphs to the user, these are all decent graphs. All the pie charts and bar graphs are understandable, only the RRD graphs lack explanation on the special RRD page.
The same series of graphs are made for each individual interface that can be added to nTop quite easily via the NetFlow plugin.

Tables
Most of the tables are clear and understandable for the user. A lot of useful information is presented with tables by nTop. Mostly the throughput is displayed in these tables. They display throughput information in a clear fashion between different points in the network. Also all the totals on the current interface are displayed in the same fashion. These totals are made even clearer by the supporting graphs to clarify this data. This makes sure the main task of monitoring the total traffic over a gateway towards SURFnet will be clear.
An administrator can make the necessary adjustments to the tables in ‘Network Traffic’ section and select which of those will be displayed. This selection of protocols displayed in the table is a basic startup option.
Further only the ‘Last minute view’ can be a little confusing to the user, despite the explanation that is present.

Presentation
NTop has a fair amount of pages which can be visited by the user, sometimes it is not clear which information is retrieved when selecting a certain menu item. Instinctive usage could be a problem in the beginning.
There are no standard possibilities to export graphs or tables.

Excess information
NTop tries to display way too much information on some pages. (E.g. RRD-plugin page and the first page with traffic stats) It does contain a lot of useful information but the user just gets overflowed by information he or she is not looking for. Consequently the user himself has to filter out this excess data. It would be nice if the first page was split up in 2 real sections, it just contains too much information.

---

19 A table where all traffic in the last minute with their ports are named
Figure 2: The large amount of ghost traffic reported by softflowd

<table>
<thead>
<tr>
<th></th>
<th>Dropped (hlopcap)</th>
<th>Dropped (ntop)</th>
<th>Total Received (ntop)</th>
<th>Total Packets Processed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unicast</td>
<td>0.5%</td>
<td>0.0%</td>
<td>18,835,642</td>
<td>18,835,642</td>
</tr>
<tr>
<td>Broadcast</td>
<td>15.6%</td>
<td></td>
<td>2,614,650</td>
<td></td>
</tr>
<tr>
<td>Multicast</td>
<td>0.8%</td>
<td></td>
<td>132,459</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3: Part of the nTop main view
<table>
<thead>
<tr>
<th>Host</th>
<th>Details</th>
<th>Data</th>
<th>Packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.10.10</td>
<td>0.0 bps, 0 bps</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>192.168.10.20</td>
<td>0.0 bps, 0 bps</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>192.168.10.30</td>
<td>0.0 bps, 0 bps</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>192.168.10.40</td>
<td>0.0 bps, 0 bps</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>192.168.10.50</td>
<td>0.0 bps, 0 bps</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>192.168.10.60</td>
<td>0.0 bps, 0 bps</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>192.168.10.70</td>
<td>0.0 bps, 0 bps</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>192.168.10.80</td>
<td>0.0 bps, 0 bps</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Figure 4: nTop throughput overview
Appendix E: Stager test report

Stager is a generic tool for storage, aggregation and presentation of network statistics. Stager consists of two main components, the so-called frontend and backend. The backend, written in Perl, is responsible for storing and aggregating flow-data. Currently three modules exist for the backend: NetFlow, MPing and SNMP. These backend modules rely on third party tools like e.g. flow-tools.

The frontend is a web application, which is responsible for presenting the gathered statistics to the user.

Stager evolved from the European SCAMPI project. SCAMPI was a project to develop a scaleable monitoring platform for the Internet. It also aimed to promote the use of monitoring tools for improving services and technology. The project ended in September 2004, but Stager is still supported and in further development. Arne Øslebø, Andreas Åkre Solberg and Espen Breivik from Uninett\(^{20}\) are currently working on Stager.

Conformance to the previous selection

Stager consists of a collector and an analyzer, which are called the back- and frontend. In the selection of the collector function of Stager all the claims that were made on the website were fulfilled. Stager is able to analyzed more then just throughput data. This tool allows the user to gain a good insight into all the traffic that is present on the network.

Installation

Stager relies on several dependencies. The backend has the following main prerequisites: Perl, Flow-tools and PostgreSQL. Furthermore, Perl needs 10 specific modules to serve Stager correctly.

The frontend relies on PHP, Apache, PEAR, JpGraph and Smarty. Stager comes with a detailed installation manual made with system administrators in mind. Experience in using *nix systems is required when installing Stager.

The installation manual is a good document but lacks some explanation. It describes all the subcomponents very well, but some structure is missing. Sometimes it can be unclear in what order actions have to be done. This is quite evident because some actions do have to be performed in a certain order.

As a helping device Stager comes with a diagnostics page showing the status of the installation. When the installation requirements are not fulfilled, the diagnostics page will inform the system administrator what still has to be done. The authors’ experience showed out that the diagnostics page is a great help but surely not nirvana.

Stager has not been tested in all possible configurations. While installing, the authors of this document encountered some problems. Stager is still in development and therefore bugs can show up. Compatibility could also have been better. However, Stager is evolving quite fast and thus has a great potential.

Around Stager a close community has gathered. When encountering problems, consulting this community should probably help out.

\(^{20}\) Uninett is the Norwegian research network.
Installation notes:

PHP must be compiled with --enable-memory-limit and --enable-calendar. This is not stated in the manual. Make sure that host or hostx is installed, this is needed for getting dns-entries. When using hostx the file getRouterInfo.pl should be edited.

Problems encountered

Some problems were encountered with PHP version 5. When the authors installed Stager it was not yet stated that Stager was not compatible with this version of PHP. Currently the website informs about this issue. The development team will solve this compatibility with the next big release.

Possibilities for improvements

Like stated above, PHP version 5 is not yet supported. Within a short period from now a new release will overcome this problem.

Stager can only work with NetFlow version 5. Looking at the adoption of POSIX and NetFlow version 9, this can be considered a drawback. Stager’s developers are eager to adopt POSIX into the architecture. The bottleneck is that Flow-Tools does not yet support these newer versions. It is unclear when this support will be provided. Stager is currently dependent on Flow-Tools, but the relation in between is very loose. Therefore Stager might be combined with some other tools in the future.

Remarkable features

When adding a new router to the network, Stager is capable to automatically detect this. Stager periodically checks incoming NetFlow packets for new measurement points. When a new router has been found, it is added to the database and Stager reconfigures itself. This obviously enlightens the network administrator. Please note that routers are only detected when exporting NetFlow to Stager's collector.

While updating or reconfiguring, Stager can go into a special maintenance mode. In such an occasion the frontend notifies the users that maintenance is in progress and a lower performance can be expected.

Another great advantage of Stager is its scalability.

Scalability

Stager is built with scalability in mind. Several backends can be distributed over the network. These backends can be placed on strategically locations near routers with high capacity. Every backend collects the NetFlow data from several observation points. This data is processed and stored in a database. Several backends can share such a database as can be seen in figure 2. Stager is also capable of working with distributed databases. An example:

Uninett is collecting NetFlow data from 23 routers with 277 interfaces in total. Speeds range from 155Mbs to 10Gbs. Sampling rates up to 100 are used for the routers with high capacity. 3 Systems are distributed of the network for collecting NetFlow data. During peak hours more then 30GB of data gets collected. These 3 systems process the gathered date and store it in the database, a fourth machine. Each hour 375000 entries are added to the database. The frontend, a fifth machine, presents the measurements to the user.

Stager can be fully designed to fulfil personal needs.
Figure 1: Backend functionality. Data collection is performed by Flow-Tools.

Figure 2: Example of Stager running with three backends, two databases and one frontend.

Collector input methods
The collecting activities are located at the backend. Stager depends on Flow-Tools for gathering the NetFlow data. Flow-Tools listens on a designated port for these NetFlow PDUs and stores its information in a dedicated and structured folder.

Stager periodically checks this folder, processes and stores the data in the PSQL\textsuperscript{21} database. Currently Stager can only work with NetFlow version 5. Besides gathering

\textsuperscript{21} PostgreSQL.
NetFlow, Stager can also be installed with the SNMP en MPing modules. The MPing module does latency, loss and jitter measurements on multiple hosts on the network. For gathering and presenting CPU load, temperature and more, the SNMP module can be used. Overall this makes Stager quite a complete package.

Compatibility

Stager’s architecture is quite specific. The backend’s and frontend’s interfaces are specially defined for Stager. When it is desirable to let another tool gather information from the database, it should exactly know the database’s structure. Thus, such a tool should be custom patched.

Although Stager provides the module architecture, it is not possible to hook up self written modules. Stager must be compiled with support for the used modules. Currently it can only be compiled with the domestic modules. In the future they are planning to make the architecture more open.

Like stated earlier, NetFlow version 9 or IPFIX is currently not supported. The developer’s goal is to provide this compatibility as soon as possible.

Storage

All measurement data is stored in a PostgreSQL database. This is a good choice compared to MySQL. Depending on the monitored network’s size, Stager will generate quite some data. Therefore, PostgreSQL is a better choice above MySQL.

To reduce the database’s size, data in the PostgreSQL database is aggregated over time. This aggregation is performed by a separate script that is run by the crontab. Default aggregation is done after every day, week and month. However, this can be fully configured by the system administrator.

Experiences show that performance of both frontend and backend heavily rely on the performance of the database. Therefore the developers have put much effort in improving the SQL queries. It is recommended to run the database on a system with much RAM and a good IO-performance. When monitoring a large network, distribution of the database can be desirable for performance reasons.

Stager comes with documentation for tuning PostgreSQL’s performance. This can be rather evident since the database is most likely the bottleneck in Stager’s architecture. The big data volumes would make a sequential search impossible within a reasonable amount of time. Therefore all queries to the database rely on use of indexes. The database will operate much faster when these indexes can be totally stored in memory. That is why RAM is so important.

Graphs

After making a selection of the required statistics, the user can plot a graph to its own purposes. Stager provides advanced plotting options. It can plot lines, area graphs (both cumulative and direct) and pie charts. A selection can be made among a linear and logarithmic scale. Even the output resolution can be defined.

All these option make Stager highly configurable and allows the user to graph the exact information he or she needs. A clear disadvantage is that all these options just suit a certain user.
Tables

Tables are used very frequent in Stager. This is due to the fact that Stager can provide lots of information which is best presented by tables. Within the tables a selection can be made for what to plot in a graph.

Presentation

On the first glimpse the user can get confused and overwhelmed by all the information Stager provides. After getting used to the system navigation gets easier. The tool provides a lot of information and it is clear that the developers made efforts to present in a clear way.

It can be concluded that Stager is most interesting for network managers. For the interested end user Stager is simply too complex and provides too many options.

A user who is unfamiliar with Stager can not get the wanted information within a few mouse-clicks.

Figure 3: Example of stager table view
Figure 4: Example of stager graph view
Appendix F: Cacti test report

Cacti is a complete front-end to RRD-Tool. All configuration settings are stored in a MySQL database. An option for detailed user-administration with different user-rights is provided. The front-end is completely PHP driven. Along with being able to maintain graphs, data sources, and round robin archives in a database, Cacti handles the data gathering. There is also SNMP support for those used to creating traffic graphs with MRTG. [CAC05]

Conformance to the previous selection

Based on our previous theoretical selection, Cacti performed like expected. The tool allows its users to create and administrate measurements in an easy to use way. It outputs RRD graphs, which are very clear. When monitoring a large network the overview of all graphs will be lost. Just take a look in Figure 1 and imagine the monitoring of 100 nodes.

Furthermore, Cacti’s presentation method does not allow a clear presentation of flow-statistics. For SNMP or polling variables it is perfect though. Cacti could be used for creating and administrating graphs, presentation possibilities are limited.

Monitoring flows is possible with Cacti. This is however limited to the use of graphs. For every parameter required to monitor a new database has to be created. Tables are not used and discussed more extensively further in this report.

When making use of the right scripts, Cacti is capable of presenting delay, loss and jitter information between different hosts. These measurements could be performed remotely over a secure SSH connection. It has the big advantage that Cacti allows the user to measure with self chosen tools.

Installation

Installing this tool on a Gentoo system was quit easy. By just emerging the installation package almost all dependencies were properly installed as well. Some points of attention are the USE-flags that have to be set correct and a right configuration of both the crontab and MySQL.

Running experience

Experiences concerned running Cacti are positive. It seems to be stable and the user interface is surprisingly friendly. A minor problem that was encountered is that the logarithmic scale didn’t work as expected.

Cacti allows the user to create data sources, graphs and templates. To handle data gathering the user can create SNMP queries and/or supply Cacti the paths to any external script/command. Cacti will then gather this data in a cron-job and populate a MySQL database/the round robin archives.

Once one or more data sources are defined, an RRD-Tool graph can be created using the data. Cacti allows to create almost any imaginable RRD-Tool graph using all of the standard RRD-Tool graph types and consolidation functions. A color selection area and

---

automatic text padding function also aid in the creation of graphs to make the process easier.

Cacti is able to scale to a large number of data sources and graphs through the use of templates. This allows the creation of a single graph or data source template, which defines any graph or data source associated with it. Templates for hosts enable to define host properties, so Cacti knows exactly what kind of information can be gathered from that kind of systems. [CAC05]

**Possibilities for improvement**

To the authors’ opinion Cacti is not designed for working with flow-related data. Supporting this would be a great enhancement to the current version. Flow measurement allows a user to gather very specific data that currently cannot easily be presented by Cacti. With such a flow extension the tool would become even more interesting then it already is. Flow measurement occurs more often and thus Cacti should react to that.

**Remarkable features**

The option to periodically export generated graphs is very welcome. This could also be done with some script making use of ftp, but a built-in function is more user-friendly. Exporting the graphs makes it possible to easily use them on another location, e.g. another website.

The most surprising feature of Cacti is its zoom-function. The user can select an area within a graph and zoom into this time domain. This works really simple and allows the user to examine certain measurement behaviour more closely. This is a nice extension to the good calendar function. This function allows the user to select a start and end point. Cacti will then present the requested data over this user specific time domain.

One other feature that should be mentioned is that the user can hook-up his or her own scripts to Cacti. This topic is further discussed elsewhere in this report.

**Scalability**

To run Cacti several other software components are necessary: PHP, MySQL, Net-SNMP, Apache2, RRD-Tool and Perl.

Apache2, PHP & MySQL are used for displaying the Cacti web-interface. Furthermore MySQL is used to keep track of the monitoring activities and databases. These databases are generated with RRD-tool, which is also the core that actually generates the graphs. Net-SNMP and Perl are used for gathering data. A system running Cacti should at least support these dependencies.

What the actual system requirements are is hard to say. This also depends on the numbers of objects Cacti has to monitor. For every monitored object a new RRD is needed. One of the test systems has been monitoring 57 different objects. This took up to 80% of the available resources when Cacti was in its polling procedure which occurs every minute. However, everything is running very smoothly on this Celeron 800 / 128 RAM Gentoo system.

23 Date and time

A graph is counted as an object. A monitored system can contain different objects.
**Easy of use**

Cacti is very well documented. The installation was made very easy by just following the provided tutorial. For extra support cact.net has a lively community on which people help each other out with Cacti troubleshooting.

Because Cacti is entirely configured through a web-interface, people without special knowledge should be able to use it.

**Collector input methods**

Like mentioned before, Cacti is capable of polling by using third party scripts. These scripts can be installed for all supported languages on the system. Cacti was successfully tested with Perl and Java scripts. It just runs a command line entry and stores the returned data into a RRD.

Cacti can make use of the Net-SNMP package. This allows it to perform SNMP queries on other machines. It is even capable of detecting active OIDs.

**Compatibility**

Every tool that is able to work with RRD is compatible with Cacti. However, the formatting should naturally be the same.

Because Cacti can run scripts of every installed language, it is also compatible with all of these. This makes the system very flexible and allows the user to create input methods.

**Storage**

It makes use of RRD for storing the statistics and uses MySQL for keeping track of its own activities (settings, users, measurements, etc).

**Graphs**

First of all, Cacti can be seen as a direct user interface for RRD-Tool. When familiar with RRD-Tool, one knows what Cacti’s graphing capabilities are. RRD is a system to store and display time-series data. It stores the data in a very compact way that will not expand over time, and it presents useful graphs by processing the data to enforce a certain data density. [RRD05]

**Tables**

Cacti does not make use of tables. All statistics are presented by making use of graphs. Since it gathers data that actually is best presented in graphs, this more or less suits Cacti’s current services.

A table for comparing current delay and loss statistics would however have been nice. On the other hand, this could also be combined in a single graph. Tables can in certain contexts provide an extra overview and therefore hopefully will be supported in the future.

**Presentation**

There are three different methods for presenting the available statistics: tree view (Figure 1), list view (Figure 2) and the preview view (Figure 3)
The tree view, as discussed before, can become unclear when presenting a big number of hosts. The same counts for the list view. This option presents the user a list with all available graphs and prompts the user to make a selection for viewing. For a big number of hosts the preview view might even be the best available choice. Users can select a host, by making use of a pop-down menu, and Cacti returns previews of all available graphs for this host. This gives a nice overview of the available data for the selected host. The pop-down menu on its turn will be a pain in the neck when working with 100 hosts.

For a better overview of all available data, another way of presentation should be developed. In some contexts, tables can provide an extra overview.

**Excess information**

Since the user can define what information should be monitored, the amount of analyzed data that is irrelevant to the user is minimal. When Cacti is displaying a host, it presents all available graphs. Making a selection is not possible.

![Figure 1: Tree view can become large and unclear for big networks.](image)
Figure 2: List view provides a bad overview.

Figure 3: Preview view.
Appendix G: Flowscan+

In 2001 KISTI funded a project to survey several flow measurement tools. After this survey they had put effort in improving the Flowscan tool, which was developed by Dave Plonka. The project, which ended in December 2001, resulted in Flowscan+.

Flowscan+ relies on Tobi Oetiker’s RRD-tool for graphing and CAIDA’s cflowd for gathering flow data. Flowscan+.

Despite that Flowscan+ is used on several locations it is not desirable to use this tool, or its predecessor. It relies on cflowd, which is not supported by CAIDA anymore. Their website advises to use Flow-tools instead. Most of Flowscan+’s components, except RRD-tool, lack support and future development. Future support for NetFlow version 9 or IPFIX can not be expected. The authors were unable to get Flowscan+ working.

Problem

The problem encountered came down to that arts++ needs gcc 2.96-112-7.2 for compilation. Arts++ is a required component for running the Flowc. On the authors’ test machines only gcc 3.3.5 was available. Both this compatibility problem and the earlier description was reason for not continuing the Flowscan+ test.

Scalability

According to the developers a DDoS can seriously hamper the measurements which are performed: “During a recent inbound Denial-of-Service attack consisting of 40-byte TCP SYN packets with random source addresses and port numbers, I have seen a single ‘5-minute’ flow file greater than 500MB! Even on our fast machine, that single file took hours to process.”

This shows traffic measurements can also be used to recognize certain attacks. But at the same time the measurement system can be seriously bothered and may even break down.
Appendix H: Pmacct-fe test report

Pmacct-fe is a network traffic analyzer that is implemented in PHP. It uses Horde for user authentication, JpGraph to plot graphs and relies on a PostgreSQL or MySQL database to read the data that is collected by a probe. The only probe that fills the database in the correct way is Pmacct. Although with a little effort other programs can be configured to fill the database in a compatible format.

Pmacct-fe is a recent project of Paolo Lucente from the University of Bari in Italy. Lucente previously created a probe called Pmacct which could aggregate network traffic into flows. The Pmacct-fe project is an effort to display the gathered flow data to network administrators.

Installation

Pmacct-fe was tested on a Gentoo system. Before installation it was necessary to emerge several package dependencies for Pmacct-fe.

During the installation of Pmacct-fe the authors only encountered a few problems. These were partly caused by compatibility with the Horde framework. Pmacct-fe uses some packages from the PEAR plugin for PHP. PHP also has to be compiled with a variety of options enabled.

Some other problems were encountered while installing Horde, but these were relatively easy to correct. By using a PostgreSQL database user management becomes relatively easy in Horde.

Other problems were found when trying to get the Pmacct probe to place its output into the PostgreSQL database. Performing a complete reinstall of the probe solved this problem.

The following options must be enabled in PHP for Horde to work correct:

--with-gettext, --with-dom, --with-mcrypt, --with-xml, --with-icnv, --enable-mbstring
--with-mbstring=all, --with-gd, --with-mime-magic, --enable-bcmath.

Running Pmacct-fe

Before being able to access the statistics, permissions for horde users had to be set in the database. After this the installation of Pmacct-fe was complete. Pmacct-fe can now be accessed by using any web-browser. The PHP files were working correctly and the data that was collected by Pmacct can be extracted from the PostgreSQL database.

Statistics are presented through Horde on which a selection can be made among different time periods. After such a selection the traffic is shown to the user. This gives immediate insight into the amount of data traffic per IP. This IP information might be privacy sensitive so it’s better to be careful with who has access to these statistics.

Problems encountered

A problem with JpGraph remained and kept displaying the same error. Also the Pmacct-fe icon in horde did not work anymore at some point. Another thing is that the outgoing traffic seems to be stuck at zero, whether this is caused by the machine it runs on is still unknown; also see Figure 1.

---

An installation method for the Gentoo Linux distribution.
Possibilities for improvements

In the main table of Pmacct-fe (see Figure 1) there is some excess information being displayed. A few entries are not very useful to be displayed on the main page. The bit/s and pkt/s rows could be removed and put under the section of the amount of data. This action will make room for future enhancements to the main page with the possibility to show some other useful data.

Another point of interest is the method Pmacct uses to store information in the database. A sample configuration file for Pmacct would be very welcome to be distributed together with Pmacct-fe.

Remarkable features

Pmacct-fe contains most of the standard functions that are required to display flow information to the user. It has an extensive user authentication backend with which all user access possibilities can be configured. This allows restrictions in displaying information to the different user profiles.

Scalability

In the current configuration Pmacct-fe can be used to display data from multiple hosts. These hosts have to run Pmacct and use the PostgreSQL database that is located on the machine that runs Pmacct-fe.

Pmacct-fe is limited to just one PostgreSQL database. This can become a limiting factor in bigger networks where multiple probes are deployed.

Ease of use

Once installed the program is very easy to use and navigate. This is partially because of the compact way of displaying the information. Once a timeframe has been selected everything can be accessed from one single page.

Analyzer input methods

Pmacct-de uses a PostgreSQL database as input method. Pmacct must have supplied the data in the database. This makes Pmacct-fe and Pmacct a unique couple.

Graphs

The few graphs Pmacct-fe can generate could not be reviewed because JpGraph was not running correctly with Pmacct-fe. The authors did not take the time to solve this problem. Judged on the graph that is viewable on the Pmacct-fe website it seems to be a standard throughput per host graph.

Tables

The program provides the user with clear tables as can be seen in Figure 1. The IP’s and data traffic are clickable for a more in depth view. The data traffic can be viewed split up over the different ports that were used. This includes a per port breakdown in amount of traffic.
Presentation
  The presentation of the data is done in a clear and orderly fashion so the user can get to the relevant data fast.

Excess information
  Pmacct-fe has no real superfluous information. Information it contains is derived from the basic fields that form a flow.

Network traffic summary

<table>
<thead>
<tr>
<th>IP Address</th>
<th>In Traffic</th>
<th>Out Traffic</th>
<th>Total Traffic</th>
<th>Last Packet</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>130.89.145.41</td>
<td>3.66 Mb</td>
<td>5.35 kbytes</td>
<td>7.99 packets</td>
<td>1 b 0 b/ls 0.00 b/ls</td>
<td>3.66 Mb 2005-06-06 16:20:02</td>
</tr>
<tr>
<td>130.89.145.41</td>
<td>1.50 Mb</td>
<td>2.3 kbytes</td>
<td>1.15 packets</td>
<td>1 b 0 b/ls 0.00 b/ls</td>
<td>1.50 Mb 2005-06-06 16:20:02</td>
</tr>
<tr>
<td>130.89.145.41</td>
<td>1.49 Mb</td>
<td>2.3 kbytes</td>
<td>0.52 packets</td>
<td>1 b 0 b/ls 0.00 b/ls</td>
<td>1.49 Mb 2005-06-06 16:20:02</td>
</tr>
<tr>
<td>130.89.145.41</td>
<td>1.17 Mb</td>
<td>1.7 kbytes</td>
<td>0.68 packets</td>
<td>1 b 0 b/ls 0.00 b/ls</td>
<td>1.17 Mb 2005-06-06 16:20:02</td>
</tr>
</tbody>
</table>

Figure 1: The main table that is presented by Pmacct-fe.

Port traffic summary

<table>
<thead>
<tr>
<th>DET port</th>
<th>Protocol</th>
<th>Traffic</th>
<th>Last Packet</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>B</td>
<td>1.23 kbytes</td>
<td>2005-06-06 16:24:02</td>
</tr>
<tr>
<td>3136</td>
<td>17</td>
<td>194.7 kbytes</td>
<td>2005-06-06 15:28:01</td>
</tr>
<tr>
<td>3000</td>
<td>B</td>
<td>71.2 kbytes</td>
<td>2005-06-06 15:42:01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SRC port</th>
<th>Protocol</th>
<th>Traffic</th>
<th>Last Packet</th>
</tr>
</thead>
<tbody>
<tr>
<td>47621</td>
<td>B</td>
<td>503.7 kbytes</td>
<td>2005-06-06 18:18:01</td>
</tr>
<tr>
<td>2591</td>
<td>B</td>
<td>259.1 kbytes</td>
<td>2005-06-06 16:02:02</td>
</tr>
<tr>
<td>1074</td>
<td>B</td>
<td>213.8 kbytes</td>
<td>2005-06-06 16:04:02</td>
</tr>
<tr>
<td>53</td>
<td>17</td>
<td>194.7 kbytes</td>
<td>2005-06-06 15:26:01</td>
</tr>
<tr>
<td>3018</td>
<td>B</td>
<td>119.6 kbytes</td>
<td>2005-06-06 16:24:01</td>
</tr>
<tr>
<td>2351</td>
<td>B</td>
<td>60.1 kbytes</td>
<td>2005-06-06 11:40:01</td>
</tr>
</tbody>
</table>

Figure 2: The traffic breakdown per IP in Pmacct-fe
Appendix I: Different ways of traffic snooping

This appendix provides a short overview on the topic of traffic snooping. Traffic snooping can be performed in several different ways.

Such measurements could be performed by a router while forwarding the traffic, by a middlebox [RFC3234], or by a traffic measurement probe attached to a line or a monitored port.

Only measurements performed by a traffic measurement probe attached to a line or a monitored port are discussed. This choice was made because if the routers are already capable of efficient capturing. It would be preferable to use the embedded capturing device to export flows. Middleboxes are not feasible on high speed (over 10 gigabit) fibre networks, for lower speed non-fibre interfaces this could be useable.

Introduction to packet capture

Saving to disk each captured packet or just the relevant bytes is not feasible. Saving packets requires a significant amount of resources. Therefore, such an approach cannot be considered as a general basis for scalable traffic monitoring procedures. Single packets are not necessarily relevant for many types of traffic analysis where focus is more on packet flows.

Despite the ability to capture every packet, it will not be feasible to save them all to disk. Research has pointed out that flows give a 20:1 reduction in disk space used compared to packet saving. Disk space must be saved by using data mining techniques. In this way only useful data is stored [BBR05]. Benefits of specific flows are further explained in Chapter 4.

Libpcap

Libpcap stands for LIBrary Packect CAPture. It is a widely used Linux library used for packet capturing. Despite the fact that libpcap offers the very same programming interface across different Operating Systems. The library performance differs significantly depending on the platform being used. [LPC04]

TCPdump is a program that is also available on the same website. This is a tool that is widely used for viewing traffic.

Libpcap MMAP

Modified version of Libpcap with a small gain in performance by replacing some system calls with mmap calls. Mmap stands for Memory MAPping and is used for faster access to data by placing it in the systems memory.

Winpcap [WPC04]

Winpcap has basically the same programming as libpcap. The only difference is the program is ported to windows.
Berkeley Packet Filter (BPF)

Some library components (e.g. BPF packet filtering) have been implemented into the kernel for better performance. Drawbacks are that only some lower level protocols are supported.

Filtering is done by only selecting packets that share a common property, such as an identical header field. Unlike sampling, filtering has to carefully analyze every packet to determine whether it belongs to the set of filtered packets. This can be very computational intensive.

This is a more efficient implementation than libpcap because of the filtering. The filtering reduces the amount of packets that have to be processed.

PF_Ring [PFR04]

PF_RING is a type of network socket that dramatically improves the packet capture speed. The new ring socket is right on top of the netif_rx / netif_prepare_rx routines: no driver changes are needed.

Enhanced libpcap is used to seamlessly support the ring. Few modifications have to be done to existing systems to support this new type of socket.

It features a greatly improved performance over standard libpcap and BPF.

nCap

nCap is a recent development from Luca Deri. With improvements this program is built after the development of PF_Ring. It is an improvement that enables even a p3 500 to capture efficiently on a 1 gigabit line. With the PF_Ring the same performance is gotten from a P4 1700. [NCA05]

Conclusion

From the possibilities to perform packet capturing, Libpcap is still the most widely used alternative, either with or without the BPF extension. Libpcap is also the packet capturing method advisable for non-BSD machines. If using Libpcap it is highly recommendable to compile a PF_Ring into the kernel, because of the great speed increase. This should enable unsampled packet capture use up to at least one gigabit. With sampling enabled this could scale up to higher speeds.

For BSD machines it is advisable to be using the methods already provided in the kernel because they will have a better performance. There are special tools available for these Operating Systems.
### Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface, establishes how programs use various network features</td>
</tr>
<tr>
<td>BGP</td>
<td>The primary function of a Border Gateway Protocol speaking system is to exchange network reachability information with other BGP systems. This network reachability information includes information on the list of AS's that reachability information traverses.</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>FAQ</td>
<td>Frequently Asked Questions</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol, a standard system for sending and receiving files over IP networks.</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>ICMP</td>
<td>Internet Control Message Protocol, best known because of the popular PING program</td>
</tr>
<tr>
<td>IETF</td>
<td>The Internet Engineering Task Force is charged with developing and promoting Internet standards. It is an open, all-volunteer organization, with no formal membership nor membership requirements</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>IPv4</td>
<td>Internet Protocol version 4</td>
</tr>
<tr>
<td>IPv6</td>
<td>Internet Protocol version 6</td>
</tr>
<tr>
<td>IPFIX</td>
<td>Internet Protocol Flow Information eXport</td>
</tr>
<tr>
<td>IPX</td>
<td>Internetwork Packet eXchange, a networking protocol</td>
</tr>
<tr>
<td>MIB</td>
<td>Management Information Base, a collection of managed objects residing in a virtual information store. Collections of related managed objects are defined in specific MIB modules.</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PDU</td>
<td>Protocol Data Unit</td>
</tr>
<tr>
<td>POSIX</td>
<td>Portable Operating System Interface for uniX, includes Linux/BSD/UNIX-like operating systems.</td>
</tr>
<tr>
<td>RRD</td>
<td>Round Robin Database, a system to store and display time-series data. It stores the data in a very compact way that will not expand over time</td>
</tr>
<tr>
<td>RTT</td>
<td>Round Trip Time</td>
</tr>
<tr>
<td>SCTP</td>
<td>Stream Control Transmission Protocol, a message oriented, reliable transport protocol with direct support for multihoming that runs on top of IP</td>
</tr>
<tr>
<td>SNMP</td>
<td>Simple Network Management Protocol, a standard TCP/IP protocol for network management.</td>
</tr>
<tr>
<td>SURFnet</td>
<td>SURFnet is a high-grade computer network specially reserved for higher education and research in the Netherlands</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>VLAN</td>
<td>Virtual LAN, a network of computers that behave as if they are connected to</td>
</tr>
</tbody>
</table>
the same wire even though they may actually be physically located on different segments of a LAN

| QoS | Quality of Service |
## References

<table>
<thead>
<tr>
<th>Reference Code</th>
<th>Author(s)</th>
<th>Title and Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>[RRD05]</td>
<td>RRDtool</td>
<td>April 2005, [link]</td>
</tr>
<tr>
<td>[RTM00]</td>
<td>N. Brownlee</td>
<td>RTFM, [link]</td>
</tr>
<tr>
<td>[SFL01]</td>
<td>sflow.org</td>
<td>[link]</td>
</tr>
<tr>
<td>[WPC04]</td>
<td>University of California, Lawrence Berkeley Laboratory and its contributors</td>
<td>[link]</td>
</tr>
<tr>
<td></td>
<td>Cisco</td>
<td>“NetFlow”, 2005, [link]</td>
</tr>
<tr>
<td></td>
<td>Paessler company</td>
<td>“Bandwidth and Network Usage Monitoring using SNMP”, 2005, [link]</td>
</tr>
<tr>
<td></td>
<td>S. Ubik, V. Smotlacha</td>
<td>“Performance monitoring of high-speed networks from the NREN perspective”, TERENA Networking Conference (Greece), 2004, [link]</td>
</tr>
</tbody>
</table>