



## Stager

### A Web Based Application for Presenting Network Statistics

Arne Øslebø <arneos@uninett.no>

**Keywords:** Network monitoring, web application, NetFlow, network statistics

#### Abstract

Stager is a web based application for presenting most types of network statistics. It was originally designed as an application for presenting NetFlow reports, but as the development progressed, it evolved into a more generic application that can be used for presenting most types of network statistics. The current version has support for NetFlow, SNMP and roundtrip delay measurements.

The backend system collect the network statistics, processes it and inserts reports into a PostgreSQL database. Some examples of NetFlow reports that are inserted into the database are IP protocol usage, the 200 source IP addresses that sends the most traffic, the 200 destination IP addresses that receives the most traffic etc.

It is easy to create new report types as the frontend is completely generic and to add a new report all that is need is to create a new report template.

Stager was designed from the start as a high performance application that should be able to handle large amount of data. To solve the performance problems Stager supports distributed processing of raw data which is then inserted into a centralized database. All SQL calls to the database are highly tuned for best possible performance.

The current setup in UNINETT processes NetFlow data from 27 routers and more than 200 interfaces. During peak hours more than 500 new entries are inserted into the database every second and the largest tables in the database now have more than 450 million entries. Even so the performance of the web based frontend is very good.

Stager is available for download and released under GPL. More than 120 people have subscribed to the public mailinglist and several successful installations have been reported.

# Overview

- Introduction
  - Main features
  - Screen shots
- Design
  - Backend
  - Frontend
- Performance
  - Netflow collectors
  - Database
- Future plans
- Installation requirements

2



This paper will first give a general overview of what Stager is. Several screen shots will be provided to demonstrate how the application can be used.

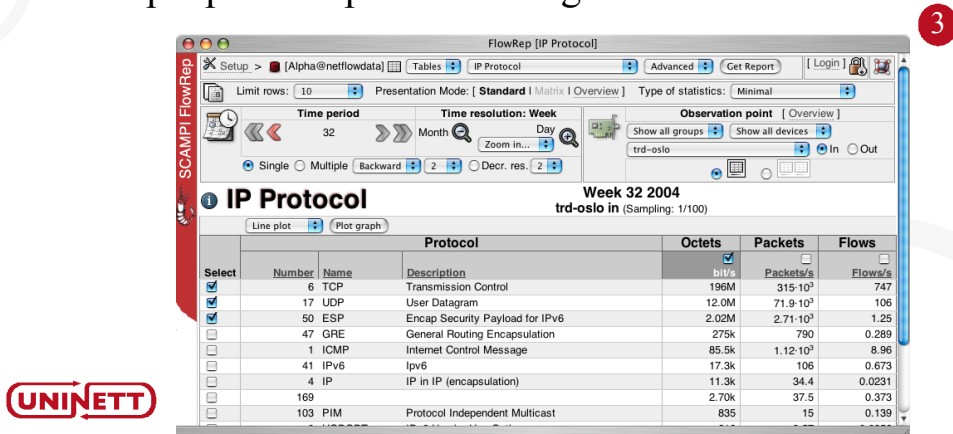
It will then describe in some detail the design of Stager, both the backend and the frontend.

Performance is an important issue in an application like Stager. A description of potential bottlenecks and performance limitations will be given as well as some real world performance numbers.

The paper concludes with an overview of future plans for Stager and some details about what is required to install it.

## What is Stager?

- A web-based tool for aggregating and presenting most types of network statistics
- GPL
- 120+ people on a public mailinglist



Stager was originally designed as a web frontend for NetFlow reports stored in a database. Since then it has evolved into a generic application capable of presenting most types of network statistics. It has several backend modules that collect various types of network statistics, processes it and inserts reports into a PostgreSQL database.. There are also scripts for aggregating statistics from for example hourly statistics to daily, daily to weekly etc.

Stager is released under the GNU General Public License and is available for download at <http://software.uninett.no>

There is a relatively active mailing list available with more than 120 people that can be used for getting support.

The screen shot shows an example of a typical Stager report, in this case a NetFlow report giving details about IP protocol usage for one single observation point.

## Main features

- Easy to use web frontend
  - Text based reports and graphs
- Support for different types of network statistics
  - NetFlow, SNMP, Mping +++
- Easy to add new reports
  - Templates and plugins
- Statistical functions
- Access control
  - Observation points and reports
- Scalable
  - Handle large volumes of data and store years of statistics
- Comparison of Stager with other NetFlow tools:
  - [http://monstera.man.poznan.pl/jra1-wiki/images/d/dd/SYS-2005-11Netflow\\_tools\\_usage.doc](http://monstera.man.poznan.pl/jra1-wiki/images/d/dd/SYS-2005-11Netflow_tools_usage.doc)



A lot of work has gone into the design of the Stager web user interface. We wanted an interface that was both easy to use as well as advanced enough to create complex reports containing information from multiple time periods and observation points. To combine these potential conflicting design goals, the user can select if he wants a simple or an advanced user interface. In the simple mode only a subset of features are available, but it takes less screen real estate and new users are not overwhelmed by all the options available in the advanced mode.

The current version has three different backends collecting data from NetFlow, SNMP and round trip delay measurements.

The backend uses plugins for adding new sources of data and the frontend uses report templates to display reports. With the templates it is possible to add new types of reports as well as new time resolutions without doing any additional coding.

There is support for detailed statistical functions so that reports can contain averages, maximum, minimum, standard deviation, percentiles etc.

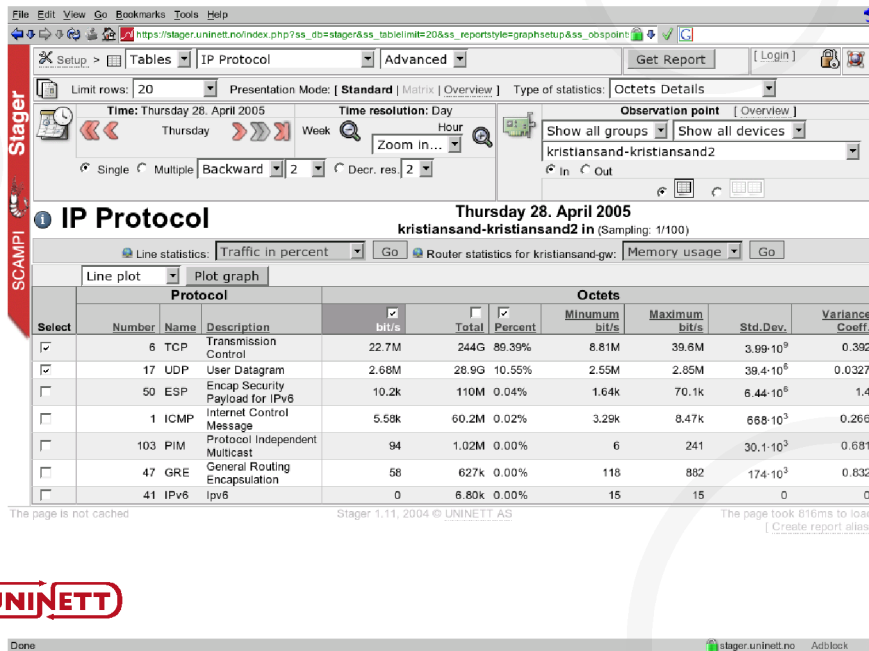
Stager also supports access control where it is possible to restrict access on a per user or group basis based on observation points and report type.

One key design criteria for Stager was scalability. We wanted an application that was able to handle large volumes of data as well as being able to store at least high level statistics for a long time.

Even in the beginning when Stager was supposed to only be able to handle NetFlow data we did not find any other open source solutions that fulfilled all the requirements we had. Now that Stager has been extended to also support most other types of network statistics, there is to our knowledge no other open source software available that comes even close at offering the same feature set.

The GEANT2 JRA1 project has written a good document comparing Stager and other NetFlow related tools.

# Detailed statistics



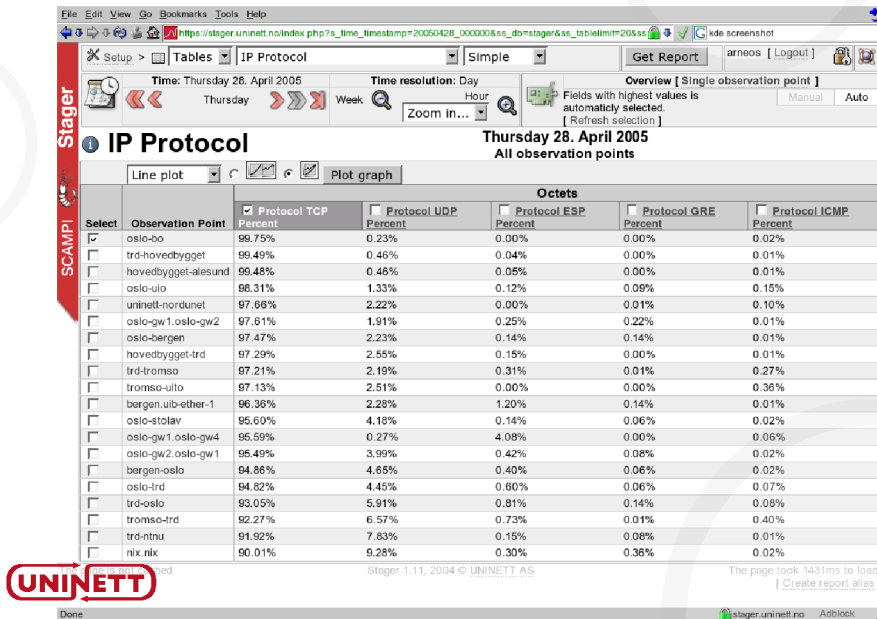
The previous screen shot on slide 3 showed a simple IP protocol report that only displayed bit/s, packets per second and flows per second. It is also possible to show more detailed statistics for a given report. This picture shows detailed information about octets for the IP protocol report. In addition to just bit/s this report shows the total amount of transferred data, the percentage of each IP protocol, minimum bit/s, maximum bit/s, standard deviation and variance coefficient.

This picture shows the advanced user interface with the most options available. The top toolbar in the picture is used for selecting the type of report that should be displayed as well as changing the user interface mode.

The second tool bar makes it possible to adjust the maximum number of rows that should be displayed in a report, if the report should be displayed as a standard table, a matrix or an overview report. It is also possible to choose the types of statistics that should be shown.

The third and biggest toolbar is used for time navigation and selecting observation points.

# Overview report



6

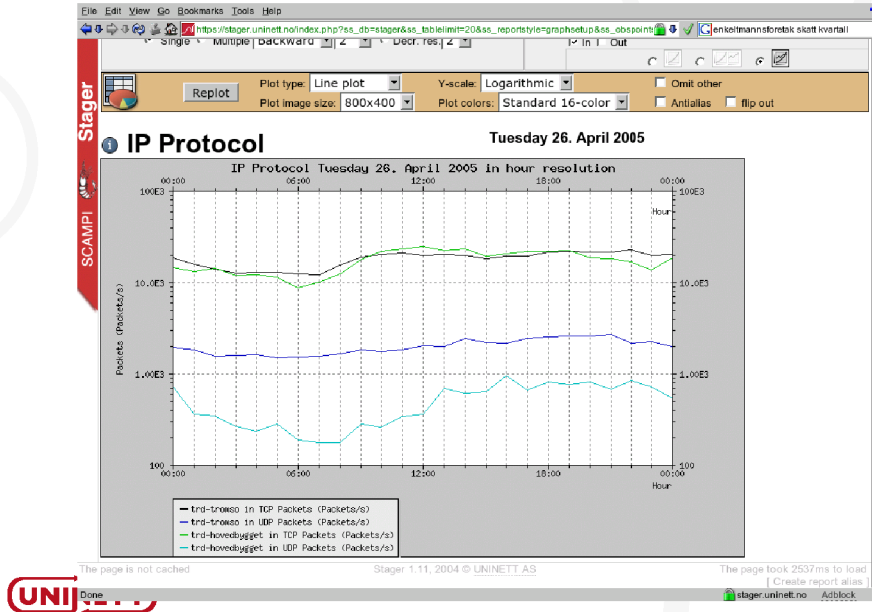
For large networks with hundreds or even thousands of observation points, it is not possible to just look at one single observation point at a time to identify trends or problems.

To detect things like which observation point has the highest percentage of TCP traffic or which router has the highest CPU load, Stager support overview reports.

An overview report show one observation point with some statistics for each row in the table. By sorting the table on a specific column it becomes easy to see which observation point has the highest or lowest value.

In this picture we can see the IP protocol overview report sorted on the TCP percentage column. This view gives us an overview of which observation points have the highest percentage of TCP traffic.

# Graph



Stager also support graphs and all numerical values shown in a report can be plotted. This picture shows a line graph where the number of packets per second for TCP and UDP traffic is plotted for two observation points. It would also have been possible to plot the two observation points in two separate graphs.

In addition to line graphs, Stager also supports area graphs and pie charts.

This graph show one day of data plotted using hourly intervals. To zoom in to a specific hour it is possible to just click on the graph. Clicking on the left or right border will display a new graph for the previous or next available time period. To zoom out it is possible to click on the top border of the graph. This makes it easy to navigate between different time periods.

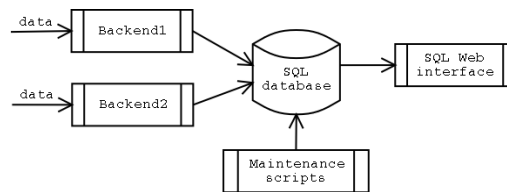
Above the graph there is a toolbar which control the appearance of the graph. This controls the size and color of the graph, the graph type, if a logarithmic scale should be used for the y-axis and if "other" should be plotted.

For the graph shown, if "other" was selected we would get two new lines in the graph showing the sum of all traffic not being TCP or UDP for each of the observation points.

# Design

- One or more backends for collecting data
- Database for storing and aggregating reports
- Web frontend

8



The figure shows the basic design of the Stager application. One or more backends collect some type of data, processes it and stores aggregated reports in the SQL database. It is up to each backend to decide what should be stored in the SQL database. For the existing NetFlow backend it is important to realize that it do not try to insert every flow record into the database as that would be impossible to scale for larger networks. Instead, what it does it to generate several high level reports based on the raw NetFlow data and insert these results into the database. Examples of reports generated are IP protocol usage, the 200 source IP addresses that generates the most traffic or the highest numbers of flows, the most commonly used multicast IP addresses etc.

There are scripts that automatically aggregates the data from one time resolution to another, for example from hourly statistics to daily. Currently all backend scripts are implemented in Perl.

The web frontend is implemented in PHP and is used for browsing the reports stored in the database.

## Backend robustness

- Handle database problems
  - Store raw data to disk
  - Generate reports later when problems are resolved
- Avoid multiple instances of the same backend
  - Wait for previous instances that are still processing the raw data
  - Detect dead locks and memory starvations

9



It is important to be able to trust the data that is stored in the database. With this in mind a lot of effort has gone into making the Stager backend as robust as possible. It is for example possible to do database maintenance and even shut down the database for short periods of time without any danger of losing data.

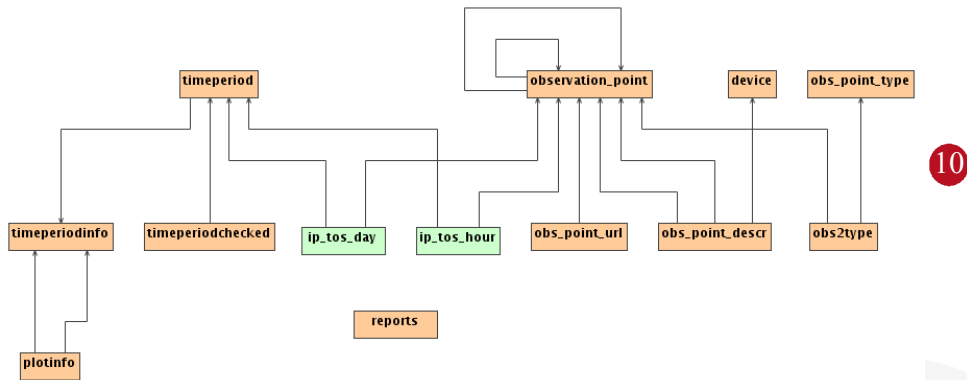
All the backend modules store the raw data they collect to disk so if there are problems inserting reports into the database, the missing reports will be generated as soon as the database is available again.

There are also several mechanisms implemented for avoiding running multiple instances of the same backend at the same time. The backends are typically started as a cron job at regular intervals. In some cases, for example during a DoS attack, the backend might not finish in time before the next instance is started. When this happens, the second instance will wait until the first one finishes before starting to process any data. If the first instance is in a deadlock and never finishes, it will eventually be killed.

If a backend is killed it means that some reports are not inserted properly into the database. In case the problem was only temporary the backend will continue to try to insert these missing reports up to a configurable number of times before giving up.

There is also a mechanism for detecting if a backend starts using too much memory. If this happens the backend is killed.

# Database design



This figure shows a simplified view of the Stager database. All reports inserted into the database are related to one specific observation point and time period. For NetFlow an observation point is the same as an interface on a router, while for other types of statistics it might a PC doing some measurements or an entire router providing SNMP data. Each backend module can define the exact meaning of an observation point.

It is also possible to group observation points into a device. For the Netflow backend all interfaces of a router are grouped together.

Stager is designed for presenting network statistics that are collected at regular intervals, for example every hour, every 5 minutes etc. Each report type have a set of time resolutions available. For the Netflow backend this is hour, day, week and month. The SNMP backend also have 5 minute intervals. Information about available intervals are stored in the timeperiodinfo table and contains information that the frontend uses for navigating between the available time periods.

To improve the performance and scalability, each report type for each time resolution has its own table. In the figure above we can see the ip\_tos\_hour and ip\_to\_day tables. These two tables are used to store hourly and daily statistics of the IP ToS usage report.

Information about how to present a specific type of report is stored in the reports table.

## Report templates

- Specifies how to present a report in the frontend.
  - bps, pps, % etc.
- Transformations
  - Overview, table and matrix.
- Multiple views for the same report type. 11
  - Specifies which data types should be visible
- Stored in a table in the database
- Example of how to define available data types for a typical NetFlow report:
  - `portnr:number:none:plotkey|  
name:string:LOOKUP(portnr,port_names,portnr,  
name):plotname`



A report template tells the web frontend how to display the data for a specific report. It tells the frontend if a number should be presented as bit/s, packet per second, %, temperature etc. One of the things a report template does is to define all the available data types that can be displayed by a report.

The report template also provides information about the transformations that are available. A single report can be displayed using different methods. The standard method is to display the report in a table where information from one single observation point is shown. Another transformation is to show an overview report where the report data is shown in a table where each row shows data from one observation point.

The last transformation supported is to show the report as a matrix. This is typically only used by source destination reports like source destination IP address.

Each transformation can also have different views. A view is a subset of the available data types that can be shown by a report. It is for example possible to have a simple view just showing the average bit/s and a more detailed view showing maximum, minimum, average and standard deviation.

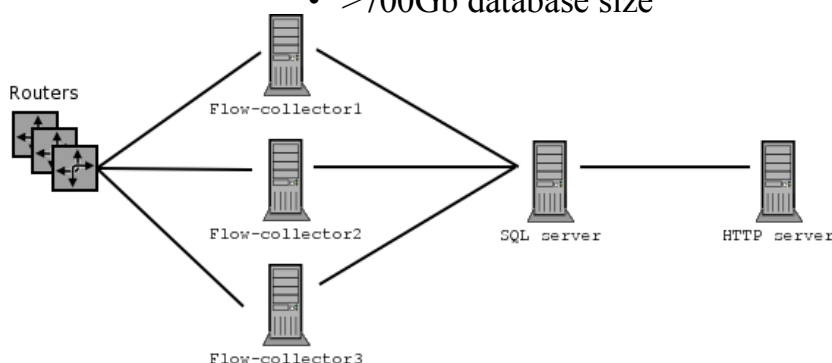
The example shows how two of the available data types of a NetFlow report are defined. Each definition is separated by a |. The first definition defines a data type called “portnr” which should be presented as a normal number. No function should be applied to this data type and it should be used as a plot key. A plot key is something the web frontend uses to decide which values to plot.

The second definition defines the data type called “name” and is displayed as a string. For this data type a function called LOOKUP is applied. What the LOOKUP function does is to use the value of “portnr” to look up the name of the port number in a table called “port\_names”.

## Our Netflow setup

- 27 routers
- 207 interfaces
- >30Gb of raw Netflow data every day
- 400.000 new entries in the db every hour
- >450 millions entries in a single table
- >700Gb database size

12



The figure above shows the setup of NetFlow collection in UNINETT's network. In addition to this we also have one PC doing SNMP collection and roundtrip measurements.

We are currently collecting around 30Gb of NetFlow data every day from 27 routers and a total of 207 interfaces. In average more than 110 new rows every second are inserted in various tables in the database. So far the size of the database is more than 700Gb and some of the larger tables have more than 450 million entries in them.

We have three PC's that collect raw NetFlow data, processes it and insert reports into a centralized database. Routers are configured to send NetFlow to the closest flow collector.

The database server is a dual Xeon 3Gz with 6Gb of memory and an Ataboy hardware raid. With this hardware the performance is quite good. In the beginning we ran the database server on a software raid and while the performance was still acceptable, there were some performance issues while doing database maintenance.

When we first started this project we assumed we would keep about 6 months of NetFlow statistics at 1 hour resolution before starting to delete it and just keep 1 day statistics. Since the performance is better than we first expected we now have more than one year of hourly statistics.

The only problem we have encountered is that PostgreSQL is not very good at reusing disk space from deleted rows in a table. So to keep the database size down we run the "cluster" command at regular intervals. Since this command physically reorder the rows in a table on the disk according to some index, performance is also increased. The drawback of running the cluster command is that it takes a long time to run and while it is running the table it works on is locked for both reading and writing.

The new PostgreSQL version 8 is supposed to have fixed this problem.

# Performance

Data from January 17 between 08:00-09:00

	PC1	PC2	PC3	Total
<b>Netflow size</b>	537MB	161MB	399MB	1097MB
<b>Sequentially</b>	9min 17s	3min 8s	5min 51s	18min 16s
<b>No insert in DB</b>	7min 11s	1min 48s	4min 52s	13min 51s
<b>Simultaneously</b>	9min 21s	3min 7s	5min 56s	18min 24s
<b># of new DB entries</b>	164702	69549	184706	418957
<b># of entries/second</b>	295.69	369.94	526.23	382.26

13

Report	Time	# of entries in table
IP Protocol overview report	614ms	17,228,300
IP Protocol overview report (previous timeperiod)	524ms	
IP Protocol detailed report	509ms	
Top src port overview	1221ms	454,034,919
Top src port overview (previous timeperiod)	717ms	
Top src port detailed	498ms	
Top src port plot (one day, two ports and obs. points)	1590ms	



The first table shows some performance numbers for processing one hour of raw NetFlow data and insert reports into the database. The NetFlow data was collected on January 17 between 08:00 and 09:00 and the first row in the table shows how much data each of the three PC's collected.

The second row shows how long each PC took in processing the raw NetFlow data and inserting the reports in the database when the processing was done sequentially. That means that PC1 first processed its data and when finished PC2 started and so on.

The third row shows the same processing but without actually inserting the reports into the database. As we can see this processing only takes a total of 4.5 minutes shorter to run compared to also inserting the reports in the database.

The table also shows the results when all three PC's processes reports and inserts them into the database simultaneously. This is virtually identical to running things sequentially and clearly demonstrates that with this setup, the centralized database is not a bottleneck.

During the processing of reports, as much as 500 new entries per second can be inserted into the various tables in the database. Even so the performance of the frontend is very good. It is important to realize that stager uses a PostgreSQL function called `copy_from` when inserting the reports. This means that we do not get 500 transactions per second since one whole report from one observation point is a single transaction.

The second table shows some performance number of the frontend. It is difficult to measure this accurately as the measured times vary depending on what the backed and maintenance scripts are doing.

What we can see in this table is that even for the largest tables with more than 450 million entries, it only takes a little bit more than one second to get an overview report and just 700ms to get a detailed report of one single observation point.

Even plotting a graph that contains four lines (two port numbers and two observation points) take only 1.6 seconds.

## Future additions to Stager

- Hierarchical observation points
- Present data from other sources
  - Raw NetFlow data, perfSONAR
- Automatic alarms and anomaly detection
  - Correlate data from multiple reports and data sources
- Support anycast for receiving NetFlow data
  - Data for one single time period can be spread over multiple collectors.
- New backends for passive monitoring
  - Accurate traffic characterization
  - Extended NetFlow
  - <http://www.ist-lobster.org>



14

We are currently working on version 2.0 of Stager. The two main new features in this release is the support for hierarchical observation points and the capability of presenting data from other sources than the database.

With hierarchical observation points it becomes possible to get reports for not just single observation points but also groups of observation points like a router or the entire network. The way it is being implemented is that it is possible to specify that an observation point is actually a collection of other observation points. When a report for such an observation point is generated, the data for all the underlying observation points are automatically aggregated.

Presenting data from other sources is primarily targeted towards NetFlow reports but can easily be used for other things as well. The idea is that a user can brows the data stored in the database and if he wants more details or other reports not stored in the database, the web frontend will try to locate the raw NetFlow data for the specific time period and if it is available it will generate the report in realtime.

After version 2.0 is released, implementing support for automatic alarms have a high priority. We want to implement a system that can take advantage of all the data from various sources in the Stager database. For example, if the SNMP data shows that a router has a high CPU load, it is possible to analyze NetFlow data to see if the reason is a DoS attack.

We also want to implement support for anycast in the NetFlow collectors. Supporting this means that if one of the flow collectors goes down, the routers will automatically send NetFlow data to one of the other collectors. The challenge here is that data for one single time period can be spread across several collectors so the backend will have to be modified to handle this.

As part of the LOBSTER project, we will also add several new backends that use passive monitoring cards for gathering network statistics. One backend will look into the payload of packets to more accurately classify packets even for applications that uses dynamic ports. Another backend will use PC's with passive monitoring cards to generate IPFIX records with additional attributes that provide information about jitter, burstiness, packet reorder and other QoS attributes for each flow.

## What do you need to run Stager?

- Linux or BSD
- Perl
  - DBI – database interface module
- PostgreSQL
- HTTP server
  - PHP
    - Smarty – template engine
    - JpGraph – graph creating library
- Stager
  - <http://software.uninett.no/>
- Mailing list for support
  - <http://tyholt.uninett.no/mailman/listinfo/stager>

15



To install Stager you basically need a Linux or BSD PC with a PostgreSQL and HTTP server.

The development and main testing is done under Debian Linux, but there are several reports of successful installation on other Linux distributions as well as BSD.

The backend is implemented in Perl and requires the PostgreSQL DBI module.

You also need a PostgreSQL and HTTP server. The frontend is implemented in PHP and requires the Smarty template engine as well as the JpGraph library for creating graphs.

The software is available for download from the homepage <http://software.uninett.no/> and there is an active mailinglist for support in case of problems.