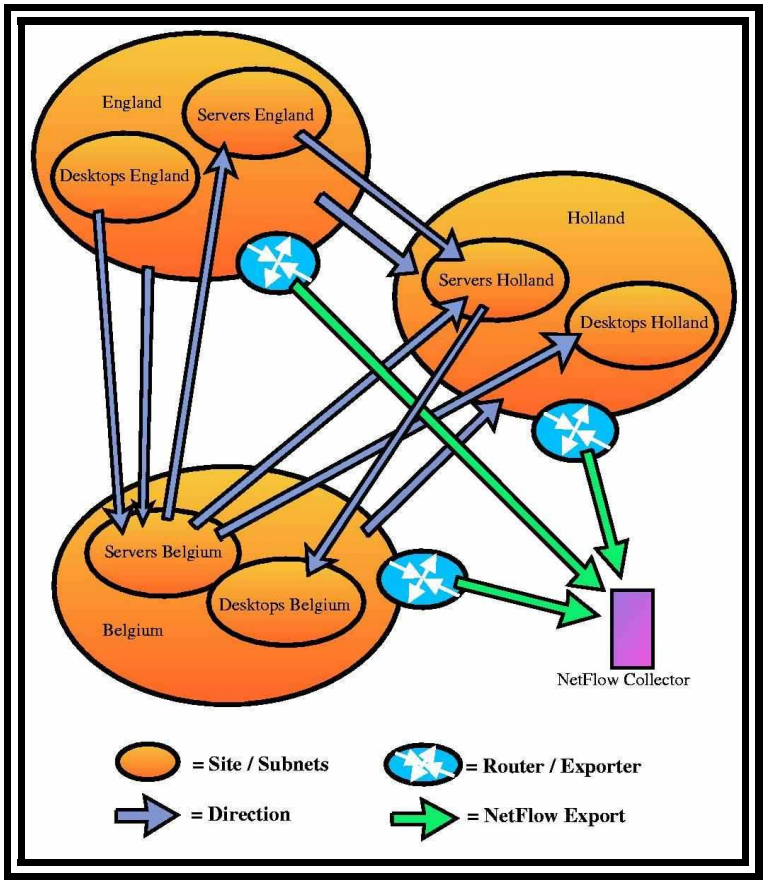
JKFlow Manual

Flexible XML configurable report module for FlowScan
Updated to version 3.5



Edited by Jurgen Kobierczynski
Email: jurgen.kobierczynski@telenet.be
URL: http://jkflow.sourceforge.net

Preface

This document describes the setup and configuration of a Netflow accounting system based on FlowScan and JKFlow. I have developed JKFlow in my final year in 2003 at GroepT in Leuven for a central Pharmaceutical site. During the project I've noticed several shortcomings in the existing report modules, so I decided to recode a new report module. My promoter allowed me to further develop and redistribute the existing code. The code base with all revisions, and all released versions are downloadable from Sourceforge.

JKFlow is a report module for FlowScan. FlowScan reads the flows from the collected flowfiles and pushes all data through the report module. The report module contains all logic for data consolidation, and is in this way actually more time/feature critical than FlowScan itself. In short: FlowScan delivers the NetFlow processing framework, but JKFlow creates the reports.
JKFlow is different with existing report modules because it is very XML flexible configurable but still very optimized to deliver the best performance in contrast to the configured features.

JKFlow is intended for network monitoring on Enterprise network where multilayer switching (MLS) consolidating multiple lines over connections with service providers is taking place. In these network architectures network monitoring with basic networking tools is next to impossible, and often expensive Netflow collection and reporting software is used. JKFlow can deliver savings on lower/middle tier reporting and graphing, especially if you are familiar with Unix. For high-end query or full data collection retrieval/billing I recommend commercial software.

JKFlow is used by several large firms and service providers , I don't have accurate numbers on how many, while it is not in thousands it must be around hundreds. I know of a case where it is used to monitor the collective network traffic of a small, but whole country.

Jurgen Kobierczynski

Contents

1: THEORY

1.1.1: Definition of a Flow

Flow definition (as stated by the definition of the "IPFIX requirements"):

"A Flow is a set of packets passing an observation point in the network during a certain time interval. All packets belonging to a particular flow have a set of common properties derived from the data contained in the packet and from the packet treatment at the observation point."

(There are several different defintions circulating on the Internet, but there different in specifying of the common field properties, like Src/Dst IP-addres, and port numbers.)

1.1.2: Flow accounting

There isn't much agreement between network companies on a generic flow accounting protocol. With this, this resulted in a wildgrowth in flow accounting protocols. These are some well-known flow accounting protocols:

NetFlow (Cisco)
sFlow (InMon, RFC3176)
LFAP, Light-weight Flow Accounting Protocol (Riverstone, Cabletron)
CRANE, Common Reliable Accounting for Network Element (XACCT)
IPDR, Internet Protocol Detail Records
RTFM, Real Time Flow Monitor (IETF)
IPFIX (IETF)

Cisco is the largest router provider, resulting in a defacto standaard of its own flow accounting protocol, named NetFlow. NetFlow, sFlowd, LFAP nad CRANE are all propieraty flow accounting protocols.

Internet Protocol Detail Records (IPDR) is a open standard for network accounting and is been positioned as opposite to CDR (Call Detail Records), used in telephony for VoIP-applications. In IPDR XML Schemas are been used to make the format extensible.

The IETF has defined the Real Time Flow Monitor or RTFM, and the most prevalent implementation is NeTraMet. Also has the IETF defined the flow accounting protocol IPFIX (IP Flow Information Export), but the initiative is still in a draft-stage.

Following in the definition of a standard flow accounting protocol is in the IPFIX IETF Working Group the “IPFIX requirements” defined. These are specifications on which must a flow accounting protocol must comply to be complete, and is a good reference for comparison of flow accounting protocols.

1.1.3: Flow Accounting today

One tool in the toolset of a network manager to manage a large-scale network is network accounting. In history expensive network probes were most needed for this task but with the arrival of new features on Cisco routers as NetFlow processing, new network monitoring tools are becoming prevalent.

The big pro behind NetFlow processing is that you retrieve network traffic monitoring data from Cisco router directly. Because these day most used networking equipment is Cisco anyway, most people can enable NetFlow monitoring out of the box, without extra investments.

1.2: NetFlow Accounting

NetFlow is a Cisco IOS implemented technology for measurement of network traffic, passing over routers and switches. The analyse of the data provides the network manager an insight of the network traffic passing over the networks. Netflow enables very detailed analyses. The implementation of a NetFlow accounti ng system is a cheap solution because the network accounting functionality is already delivered with most low and highend Cisco routers.

1.2.1: NetFlow terms

NetFlow Exporter

A NetFlow exporter analyses and collects all IP network traffic and keeps all data in cache before its flow records are exported. The flows are kept in the cache until it meets one of these conditions:

Flows of TCP connections closed with a FIN or a RST-flag.
Flows of idle connecties after a certain timeout.
Flows of long during connections will expire default after 30 minutes. (*)
If the cache fills up, heuristics will determine the expiring flow records.

(* TIP: set this timeout to the flowscan reporting period, to get correct reporting of longduring connections!)

Flow Cache

The flowrecord of expired flow records are placed in NetFlow UPD datagrams and directed to a collector. These records contains the aggregated data of the network traffic. A typical flowrecord contains the following data:

Source en destination IP address
Type of service (ToS)
Packet en byte counts
Start en end timestamps
Input and output interface numbers
TCP flags en encapsulation protocol (TCP/UDP)
Routing informatie (next-hop address, source autonomous system (AS),
destination AS, source en destination prefix mask)

Flow Collector

A Flow Collector is a node collecting the netflow exports, and filters these with predefined rules and places these records in a database.

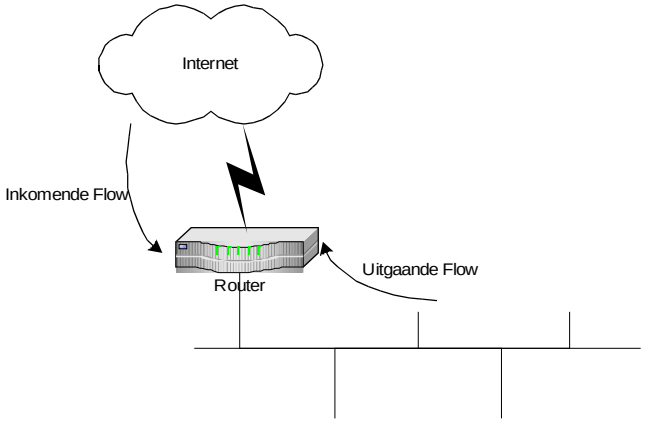
Netwerk Data Analyser

A Network Data Analyser provides the user with a GUI where all data collected and analysed by the flow collector to represent a graphic of a desired format or with aid of modules profides the users a billing structure.

1.2.2: Interpretation of a Flow

Some explanation is needed to relate the generated flows with the network traffic.

Flows are only generated on inbound traffic on the interfaces of the router/switch. A TCP session exists only of 2 connections: one connection is from the internal host to the external host and the other comes from the external host to the internal host. Because the traffic is only on the inbound interfaces considerated for flow accounting, this will not result in double-counting of network traffic.

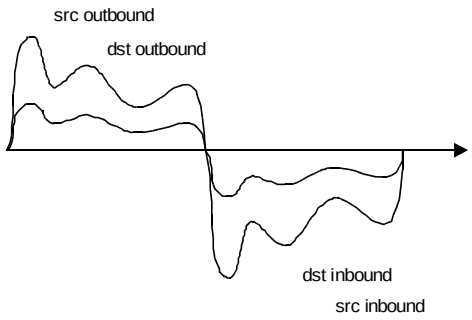


Depended of the source/destination IP-address and the source/destination ports of the flow and our subnet address we can ask ourself if the flow resulted from inbound or outbound traffic, and if the service is located internal of external:

Src IP	Src port	Dst IP	Dst port	Type
INSIDE	-	OUTSIDE	Service	Internal request to external service
OUTSIDE	Service	INSIDE	-	Answer on internal request
OUTSIDE	-	INSIDE	Service	External request to internal service
INSIDE	Service	OUTSIDE	-	Answer on external request

If you have to interprete these results, you must make the meaning of these flow directions clear for yourself, to recognise the differences in the inbound and outbound traffic, and the inside and outside services.

Outbound traffic can be an upload from internal (=dst outbound) or a download from external (=src outbound), while inbound traffic can be a download from internal (=dst inbound) or an upload from external (=src inbound).



1.2.3: NetFlow versions

NetFlow information can be exported in different format releases. Version 1 is the first version defined by Cisco and is not recommended anymore but for compatibility reasons. Version 5 adds BGP autonomous system information and flow sequence numbers. Version 7 is only supported on Cisco Catalyst 5000 Series switches with a NetFlow Feature Card (NFFC).

NetFlow version 1 (No sequence numbers, AS, or mask)
NetFlow version 5
NetFlow version 7 (Catalyst switches)

NetFlow Version 8 adds router-based aggregation schemes, allowing aggregating NetFlow data on the router itself:

NetFlow AS Aggregation
NetFlow Proto Port Aggregation
NetFlow Source Prefix Aggregation
NetFlow Destination Prefix Aggregation
NetFlow Prefix Aggregation
NetFlow Destination (Catalyst switches)
NetFlow Source Destination (Catalyst switches)
NetFlow Full Flow (Catalyst switches)
NetFlow ToS AS Aggregation
NetFlow ToS Proto Port Aggregation
NetFlow ToS Source Prefix Aggregation
NetFlow ToS Destination Prefix Aggregation
NetFlow ToS Prefix Aggregation
NetFlow ToS Prefix Port Aggregation

During configuration you must keep an eye on the used version, and make sure it meets the supported the version of NetFlow in the collector. The Cflow module used in Flowscan supports NetFlow version 1,5,6, and 7 using flow-tools, and NetFlow version 5 using cflowd.

Cisco supports these Netflow versions

Cisco IOS version	NetFlow version	Platform
11.1CA, 11.1CC	v1, v5	Cisco 7200, 7500, RSP7000

Cisco IOS version	NetFlow version	Platform
11.2, 11.2P	v1	Cisco 7200, 7500, RSP7000
11.2P	v1	Route Switch Module (RSM)
11.2(10)P and later, 11.3, 11.3T	v1	Cisco 7200, 7500, RSP7000
12.0	v1, v5	Cisco 2600, 3600, 4500, 4700, AS5800, 7200, uBR7200, 7500, RSP7000, RSM
12.0T	v1, v5	Cisco 1000,1600,1720, 2500, 2600, 3600, 4500, 4700, AS5800, 7200, uBR7200, 7500, RSP7000, RSM, MGX8800 RPM
12.0(3)T and later	v8	Cisco 1000, 1600, 1720, 2500, 2600, 3600, 4500, 4700, AS5800, 7200, uBR7200, 7500, RSP7000, RSM, MGX8800 RPM
N/A	v7	Catalyst 5000 NetFlow Feature Card (NFFC)

1.2.4: Configuration of NetFlow on a Cisco router

On the routers where you want to configure NetFlow, you must add the next rules to its configuration:

-in Interface mode on the router:

ip route-cache flow

-in Command mode on the router:

ip flow-export version { 1 | 5 [origin-as | peer-as] }
ip flow-cache entries 65536 (default = 65536)
ip flow-export destination 10.0.0.1 2055
ip flow-export source loopback0
ip flow-cache active-timeout 5

The version of the NetFlow exports are designated with the “version” keyword, while the “destination” keyword the IP-address/port asigns to which the router directs its exports.

The active timeout is the time the router reports current passing and not ended flows, The active timeout has to be set the same as the sample time of FlowScan, to avoid spikes on continuous running network traffic.

On multilayer switches (MLS) you have to configure netflow monitoring on both the switch and the router module. During a flow the first packet passes the router which will result in a

single packet flow being reported from the router module, the following packets are further processed by the switch, and are separately in another flow reported by the switch.

You have to enable netflow monitoring on all interfaces, because flows are only reported on the inbound router interface. If you ommit interfaces you will lose some outbound traffic on interfaces. Also when you use virtual interface tunnels endpoints like VPN and frame-relay, you will notice that the index of the interface changes over router reloads. To avoid this behaviour use the **snmp-server ifindex persistent** command.

2: PRACTICAL

2.1: Overview of Flow accounting tools

Before implemention of a NetFlow accounting system, I've considerated several solutions.

There exists commercial solutions, but I've looked at open-source tools in this project. Because I have already 7 years of experience with Linux, and used the included tools a lot on my job, so I know the pros and cons (are there any? ;-)) of using these open-source tools.

Because the project has to be implemented in Solaris, so it was invetable that I would browse the Internet in seeking of such tools, which could provide me some means to build a NetFlow accounting system. I have found several open-source tools which are related to flow accounting systems.

URL's and pointers:
You can find with a query for NetFlow a lot on http://www.sourceforge.net
CAIDA has several network monitoring tools on http://www.caida.org
Switch has a complete list on http://www.switch.ch/tf-tant/floma/software.html

CFLOWD	NETFLOW COLLECTOR
EHTN	The Extreme Happy NetFlow Tool
F.L.A.V.I.O.	Flow Loader And Virtual Information Output
fprobe	NetFlow probe http://psi.home.ro/flow/
Ntop	sFlow/NetFlow probe/collector
FlowScan	NetFlow framework
RRDTool	Round-Robin Database + Graphing tool
Flow-tools	NetFlow collector + conversion
Pantopis	NetFlow anti-DDoS tool
sFlow Toolkit	Conversion of sFlow to NetFlow

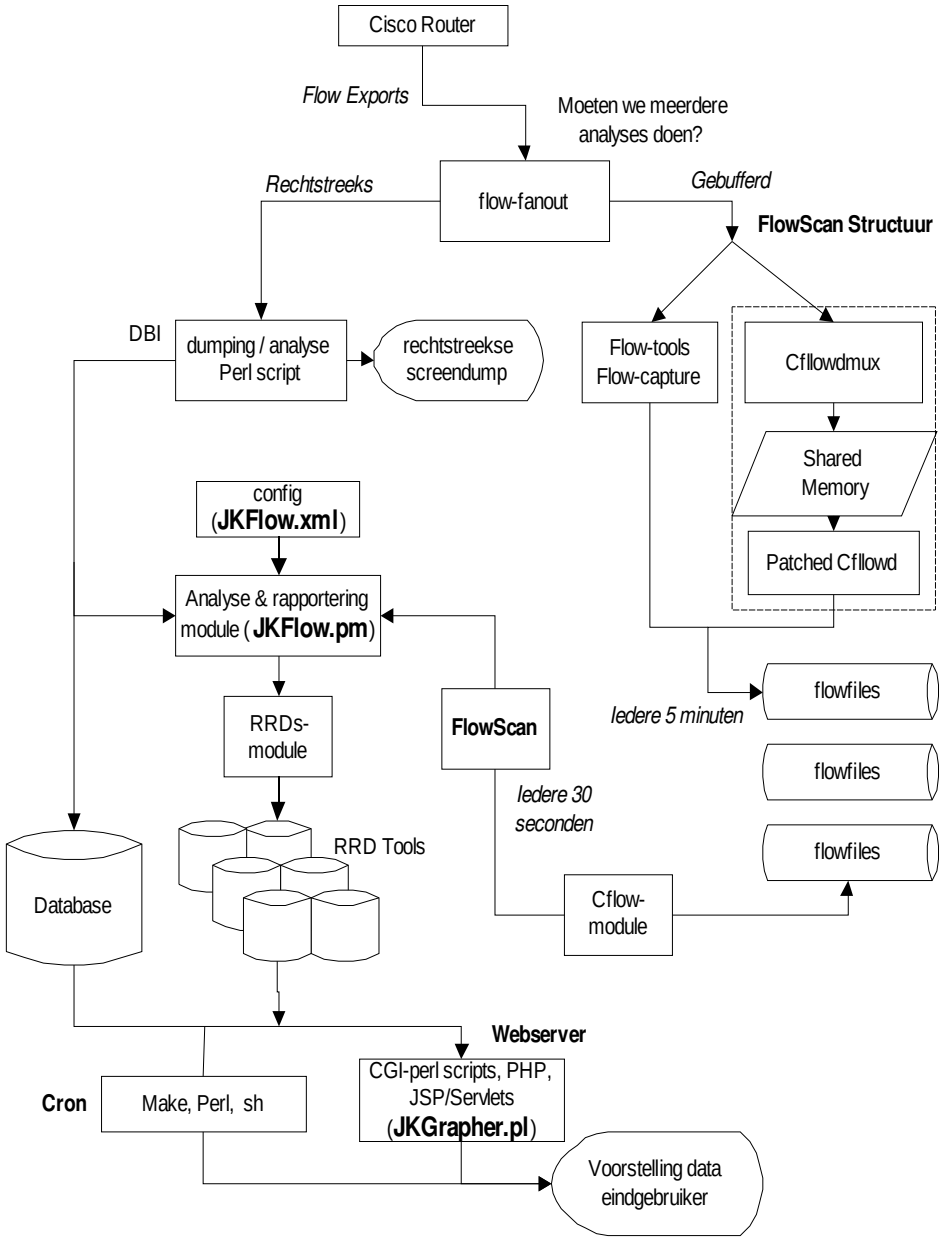
2.2: Implementation of a Flow accounting system

The implementation of a NetFlow collector is quite easy, but it can be a hard problem to analyse the right data for the collected records. Also the implementation of a presentation of the analysed data in its right format is not easy.

I had some critical questions:

Which data do I want to collect?
(Flowdata = order of Gigabytes have to be reduced to order of Megabytes)
Must the analyse be done buffered of continue?
Must the analyse be done statefully or stateless?
Do you want to keep the raw data, or do you want only to keep the resulted analyses?
Do I want to keep the data/analyse in a database, and what database?
How do I have to represent the data/analyse to the end user?

After I have evaluated some tools, I gained some insight on the possibilities of the setup of a flow accounting system.:



On the figure the JKFlow/JKGrapher modules written by me are bold mentioned. Note that for implementing this module you don't have to use a database, because RRDTool will deliver a timeseries-value round robin database for you. The “flow-fanout” split above is also not necessary, but usefull in situations where you want to combine serveral NetFlow monitoring tools. Also displayed in this graph is the option to replace cflowd with Flow-tools' capture. Using different components makes the setup of the flow accounting system easier:

- A collector which dumps all flow exports to flow files
- An analyser for the analyse of these flow files
- A report of the resulted analyse to some database
- A presentation of these data in one or another form (graphics, tables)

Within the open-source world FlowScan provides you with a framework which combines these components to a complete flow accounting system. Each component could be filled in with a

tool delivering these functionalities, and could possible be exchanged with another tool which provides the same or better features.

FlowScan gives your the following options during setting up of a flow accounting system:

- A collector Cflowd collects and dumps every 5 minutes a flowfile. Cflowd could possible be replaced with a different collector called flow-capture of Flow-tools, or possible even a collector for a different flow protocol than NetFlow (sFlow, LFAP, RTFM, ...)
- A report module (like CampusIO, SubNetIO) analysing the flow files statefull / stateless an writes the resulted data to RRDTool databases. There are different report modules available for FlowScan, but in this project I've written a report module , called JKFlow, allowing very flexible configurations for reports.
- A “make”-script for creation of graphics from RRDTool databases. Here are variations possible, like CGI-scripts as RRDGrapher. I've written a CGI-script named JKGrapher for this project.

So far I've only seen report modules using RRDTool for reporting data, which I see a gap because I would also want to report to a real database. The situation can be well explained with the easyness of using RRDTool. It combines the functionality of a small round-robin database, and a graphing functionality to create trending, and consilidation of data over larger timeframes. I've looked on the Internet to see if there has been some work for a database backend for RRDTool, but so far I didn't find anything usefull yet.

A interesting option during setup is that you could make use of a tool Flow-fanout. This is a tool included in Flow-tools which gives you the possibility to replicate flowexports in 2 or more identical flow exports. If you redirect these flows to different ports on the loopback address of your machine, you can implement several Flow analysing tools on your system.

At last, if you don't want to make use of FlowScan, can starting coding in Perl. In the sourcecode of Ntop there is a Perl-script included for capturing and dumping of flow records in a MySQL database. This script is easy to modify to dump those records to the console. I've used this script to monitor the flow exports in real-time in my project.

2.3: NetFlow Tools

I describe here some important tools I've used with the setup of FlowScan: Perl, Cflowd, RRDTools

2.3.1: PERL

PERL, “Practical Extracting and Reporting Language” is almost the most prevalent scripting language, aside Python in Unix. Perl is created in 1987 by Larry Wall in discomfort with the restrictions of the Awk-language, and is widely used for Unix systeem administration en web programming.

Perl is suprisingly fast for a scripting language, because every script passes a precompilation-step before its execution. Perl is executing the precompiled code in a “virtual machine”. In this aspect you can compare Perl a little with Java, which also uses a Java virtual machine for its execution of bincode. Perl is in this way different, because its virtual machine must be seen as a “logical” virtual machine in contrast to Java, which implements a “processor” virtual machine.

The benefit for Perl is that's faster than Java (actually I've read conflicting reports about this, but Perl is for a scripting language fast), but as a drawback it doesn't offer the write-once / run-every feature of Java. A lot of Perl's speed, can be accounted on using Perl modules, which are mostly written in the native C-language. This makes Perl as scripting language fast, but portings problems of the Perl modules will remain as a side effect.

Java doesn't have these drawbacks, and has grown into a widely used platform for e-business applications. Perl is mostly used in areas where scripting speed, logfile parsing and flexibility is at stake, like many areas in Unix system administration. Perl has also regular expression evaluation features, handy for patern matching. A supprising area where Perl is gaining ground in the Bio-informatics industry.

More information about Perl is available on the Perl-site of O’Reilly, well known from their Perl-books: http://www.perl.com.

2.3.2: Cflowd: Traffic Flow Analysis Tool

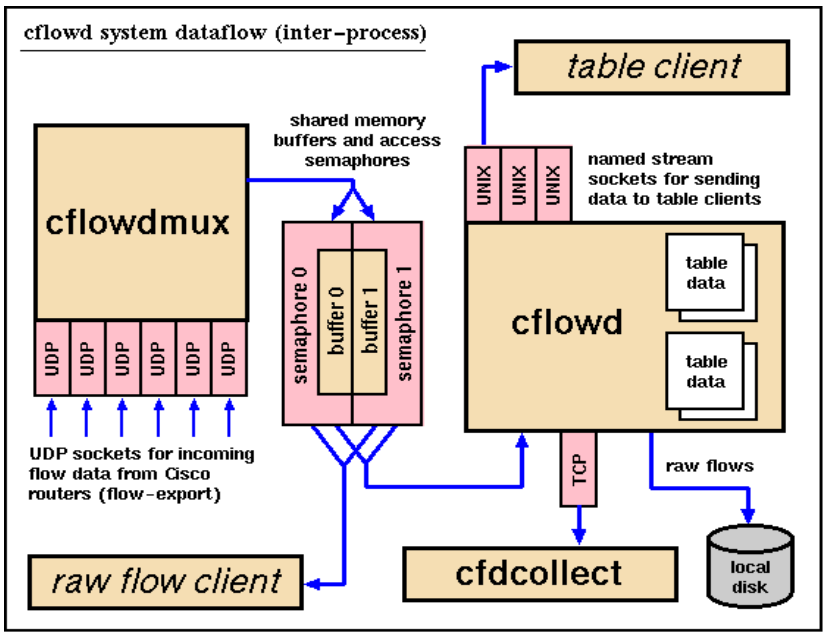
Is still maintain this section because the FlowScan homepage links to this tool. Only for this, I rather recommend you to use flow-tools as NetFlow collection tool, because it's much easier to compile and install.

Cflowd is a tool based on the Arts++ library (both available on the website of CAIDA: http://www.caida.org) Cflowd is used in FlowScan for the collection of the NetFlow exports directed from the Cisco routers in dump files, sometimes called as “flow files”.

For FlowScan you must patch Cflowd, so you can better download the Cflowd source and the Cflowd patch from the same URL: http://net.doit.wisc.edu/~plonka/cflowd/?M=D
This patch add a extra option to Cflowd so it closes the current flow dump file after a certain timeframe, to rename it and start dumping to the next dump file. In Flowscan is default a timeframe of 5 minutes used.

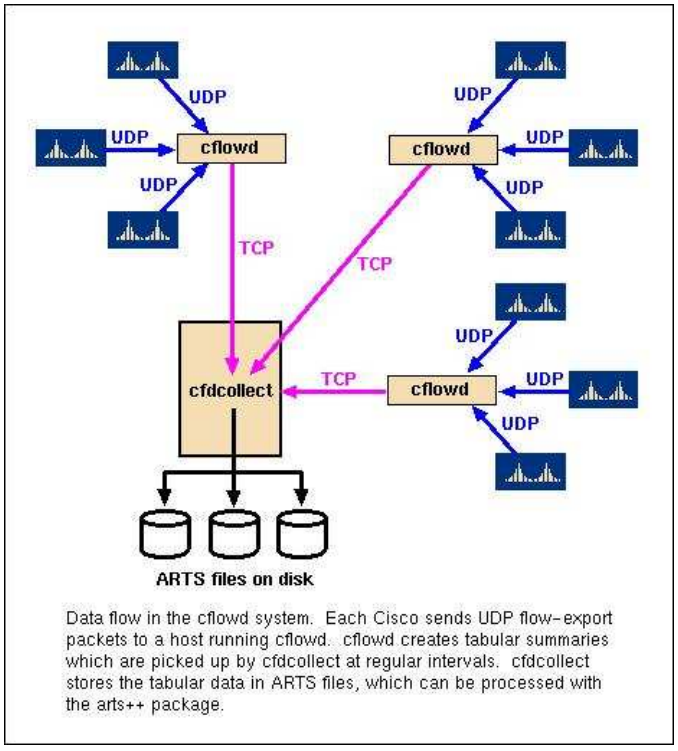
Cflowd exist from the following components:

Cflowmux: This daemon collect the NetFlow datagrams and places the flow records in a shared memory segment.
Cflowd: This daemon places the NetFlow data collected with Cflowmux in the shared memory segment in flow dump files. This deamon can also forward flow records to a central flowcollector named Cfdcollect over a TCP connection.
Cfdcollect: Central daemon for colection of flow records from cflowd daemons over TCP.



scheme of cflowd

In a simple setup are only 2 daemons used: **cflowdmux** is the daemon collecting the NetFlow exports from the Cisco routers and places these records in a shared memory segment. A patched **cflowd** reads these flowrecords from the shared memory segment and dumps the records in subsequented flow files.



central collector in cflowd

In larger networks you can configure **Cflowd** on more places as a NetFlow collector, which each collecting NetFlow exports from several Cisco's. Cfdcollect receives the flow records from the Cflowd daemon over the connection oriented TCP connection, so the connection won't drop flow records in a bottle-necked network(which is well possible in a backbone network).

2.3.3: Flow-Tools (OSU)

I recommend to use flow-tools as primairy NetFlow collection tool. The flow-tools package is not only good for NetFlow collection but also handy for solving some NetFlow related problems. Flow-tools exists of the following tools:

Collection: flow-capture, flow-fanout, flow-mirror, flow-rsync, flow-expire, flow-receive
Display: flow-cat, flow-print, flow-sort, flow-merge, flow-filter, flow-search
Aggregation: flow-stat, flow-profile
Security: flow-dscan, flow-scan-report, flow-host-profile

2.3.3.1: Replication of flows

What I sometimes used is the flow-fanout tool. It gives you the possibility to replicate a flow export to several flow collectors. This functionality gives you the possibility to use several tools form analysing the same flow export.With this tool I could monitor the flow exports with my Perl flow dump script while using FlowScan for analysing the flow exports.

This is an example:

Replicate the flows received at the local IP-address 10.0.0.1 port 2055 directed from the router exporter 10.1.1.1 to the loopback address on port 2056, and the IP-address 10.5.5.5 port 2057 comming from IP-address 10.0.0.5.

```
# flow-fanout 10.0.0.1/10.1.1.1/2055  0/0/2056
10.0.0.5/10.5.5.5/2057
```

2.3.3.2: Replacement of cflowd

If it is in some way undesired or impossible to install Arts++ or Cflowd, there is a way around: You could use *flow-capture* to replace Cflowd in using FlowScan. To make so, you must do the following things::

1:- Install Flow-tools and recompile/reinstall the Cflow shared Perl-module, delivered with the sources of flow-tools, so FlowScan can read the dump format of the dumpfiles of Flow-tools. *You must compile and install flow-tools*, in order to let the Perl-module use correct header files!

2:- Use the flow-capture tool of Flow-Tools for dumping of flow files. This tool replaces Cflowd/CflowMux.

```
# flow-capture -z0 -N0 -V5 -n287 -w/var/flows/flows
0/10.1.1.1/2055
```

The options are:
z=compression (0=no compression)
N=nesting degree (0=no directories)
V=version (5=NetFlow version 5)
n=rotations for each day (287 = every 5 minutes must there a dump file rotation)
w=write directory

localip/remoteip/port (0/10.1.1.1/2055 = receive exports from 10.1.1.1 on port 2055)

3:-patch FlowScan.pm with the patch included in JKFlow package:

```
# patch -p0 /usr/local/bin/FlowScan.pm <
FlowScan.pm_flow-tools_patch
```

4:-Specify in “flowscan.cf” the flow-capture dump files using this directive:

```
FlowFileGlob ft-v*[0-9]
```

2.3.4 RRDTool

RRDTool is a tool written by Tobias Oetiker for logging of time-series data (like network bandwidth) to generate trendings. It collects therefor data in a “Round-Robin” fashion in database-files, so these files remains in size. RRDTool has the possibility to keep the collected data in recent history more detailed, while it consilidates the data further in history, making it possible to keep the size of the RRDTool database files acceptable.

RRDTool can be downloaded from: http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/

In contrairy to MRTG, where all acquisition of data is done with SNMP, is the end user fully responsible for the acquision of the data.

RRDTool has the following functions:

Create	= Create a new RRDTool database
Update	= Update the latest values in a RRDTool database
Graph	= Create a graph from one or more RRDTool databases
Dump	= Dump the values of a RRDTool database to an ASCII-file
Restore	= Restore the values of a RRDTool database from an ASCII-file
Fetch	= Fetch data from a RRDTool database
Tune	= Change the configuration of a RRDTool database
Last	= Get the latest update time of a RRDTool database
Rrdresize	= Resize a RRDTool database

This command creates a database ping.rrd with a update period of 5 minutes:

```
# rrdtool create ping.rrd \
  --step 300 \
  DS:trip:GAUGE:600:U:U \         # 2 means: trip & lost
  DS lost:GAUGE:600:U:U \
  RRA:AVERAGE:0.5:1:600 \         # 600 values * 5 = 50 h
  RRA:AVERAGE:0.5:6:700 \         # 700 values of 6 means * 5 = 350 h
  RRA:AVERAGE:0.5:24:775 \ # 775 values of 24 means * 5 = 1550 h
  RRA:AVERAGE:0.5:288:750 \       # 750 values of 288 means * 5 = 750 d
  RRA:MAX:0.5:1:600 \             # 600 values of 5 = 50 h
  RRA:MAX:0.5:6:700 \             # 700 values of max 6 * 5 = 350 h
  RRA:MAX:0.5:24:775 \            # 750 values of max 24 * 5 = 1550 h
  RRA:MAX:0.5:288:750             # 750 values of max 288 * 5 = 750 d
```

DS = datasource, RRA = Round Robin Archive

The update of the latest values is mostly done from a Shell script with the RRDTool “update” keyword. In FlowScan is a RRDs-module used for updating of the RRDTool databases:

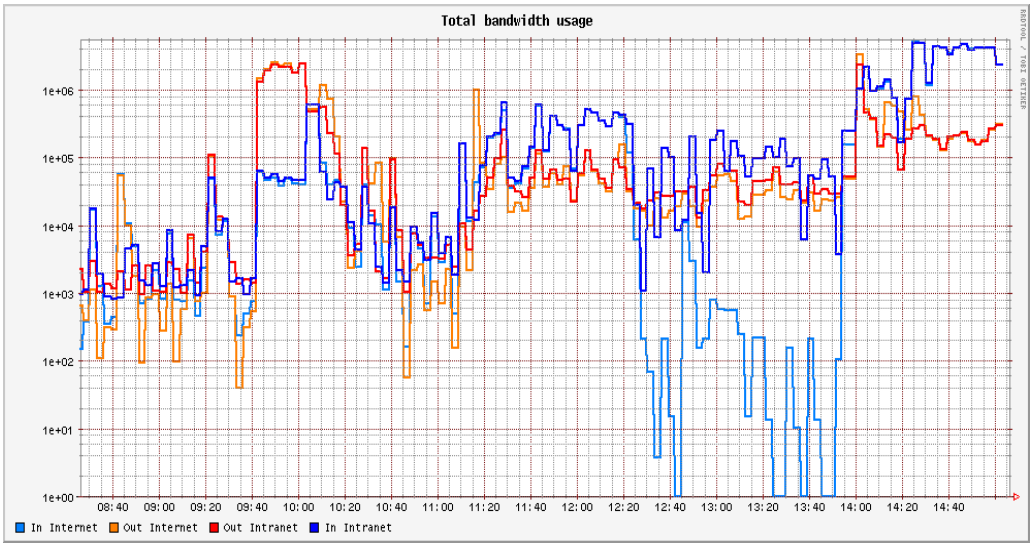
```
RRDs::update($file, $self->{filetime} . ':' . join(':', @values));
```

The source of this module is included with the sourcecode of RRDtool. The generation of the graphics is done using the RRDTool “graph” keyword. This command creates a graph from a cpu.rrd file:

```
# rrdtool graph cpu.gif \
  --title "CPU Performance" \
  DEF:la=cpu.rrd:la5:AVERAGE \
  CDEF:xla=la,10,* \
  DEF:np=cpu.rrd:nproc:AVERAGE \
  LINE2:xla\#0000FF:"la*10" \
 'GPRINT:la:AVERAGE:{avg=%.0lf' \
 'GPRINT:la:MIN:min=%.0lf' \
 'GPRINT:la:MAX:max=%.0lf)' \
  LINE2:np\#FF0000:"# procs" \
 'GPRINT:np:AVERAGE:(avg=%.0lf' \
 'GPRINT:np:MIN:min=%.0lf' \
 'GPRINT:np:MAX:max=%.0lf)'
```

DEF = selects a data source, CDEF= calculates a new variable,
LINE2 = draw a line, AREA = draws a full graph, GPRINT= print text

The following graph shows a RRDTool graph of a OpenBSD IPFilter firewall with accounting rules:



RRDTool graph from IPFilter accounting rules

2.3.4.1: RRDTool Frontends

There are several different frontends known for RRDTool graph generation:

- MRTG (This tool was a SNMP grapher tool on it's own, but uses RRDTool now also)
- Cricket
- Orca
- Smokeping (this is not a frontend, but a tool for measuring different kinds of latency)
- RRGrapher

Tobias Oetiker has written a CGI-frontend for its RRDTool tool named RRGrapher. You can select a RRD-database in a browser, select the needed data sources, setup the definitions wherafter you can generate RRDTool graphs.

For my project project I've written a CGI-frontend, named JKGrapher, myself.

2.4: FlowScan

FlowScan is a NetFlow monitoring framework written by Dave Plonka existing of Perl-scripts, combining the functionality of these modules to a complete package for the collection and analysing of flow exports.

Stated as readed on the CAIDA website on http://www.caida.org, where you can download this tool:

“FlowScan analyzes and reports on Internet Protocol (IP) flow data exported by routers. Consisting of Perl scripts and modules, FlowScan binds together (1) a flow collection engine (a patched version of cflowd), (2) a high performance database (Round Robin Database - RRD), and (3) a visualization tool (RRDtool). FlowScan produces graph images that provide a continuous, near real-time view of the network border traffic.”

Several tools and modules are needed for installing FlowScan:

Arts++	http://www.caida.org/tools/measurement/cflowd/ (2-1-b1) A daemon for the collection of Flow exports from Cisco routers.
Cflowd	http://net.doit.wisc.edu/~plonka/cflowd/?M=D (2-1-b1-djp) Patch for dumping of flow files.
Kornshell	http://www.kornshell.com/ or http://www.math.mum.ca/~michael/pdksh/ a Unix Shell
RRDTool	http://ee-staff.ethz.ch/~oetiker/webtools/rrdtool/ (versie 1.0.42) Round robin logging database for generation of graphs
Perl	http://www.perl.com (version 5.8) Practical Extracting and Reporting Language
FlowScan	http://net.doit.wisc.edu/~plonka/FlowScan/ (version 1.006) The Perl scripts that combines all packages.
Perlmodules	RRDs, Shared library included in RRDTool source Boulder::Stream (1.30) http://search.cpan.org/search?dist=Boulder ConfigReader::DirectiveStyle (0.5) http://search.cpan.org/search?dist=ConfigReader HTML::Table (1.17) http://search.cpan.org/search?dist=HTML-Table Net::Patricia (1.010) http://net.doit.wisc.edu/~plonka/Net-Patricia/ Cflow (1.051) http://net.doit.wisc.edu/~plonka/Cflow/

2.4.1: Setup of FlowScan

Personaly I consider the installation of FlowScan in Linux Fedora Core 3 much easier than in Sun Solaris 8, because you won't have to update as much tools like patch, tar, m4, bison and don't have to update Perl because the native Perl implementation of Solaris cannot call the perl

modules, compiled with GCC. This issue can be resolved in Solaris with installing a new GCC -compiled Perl version,while it is easier to get FlowScan working out first hand in Linux.

I've noticed this issue of getting stuff compiled and smoothly running on Linux easier and on Solaris with some effort quite a lot. Maybe it's because I'm more biased to Linux, and the real benefit in using Solaris lies more in its 3th party application support, which we don't need here. Anyway, I guess that an installation of FlowScan in Linux would take 3-4 hours and in Solaris, it would take 1 or 2 days.

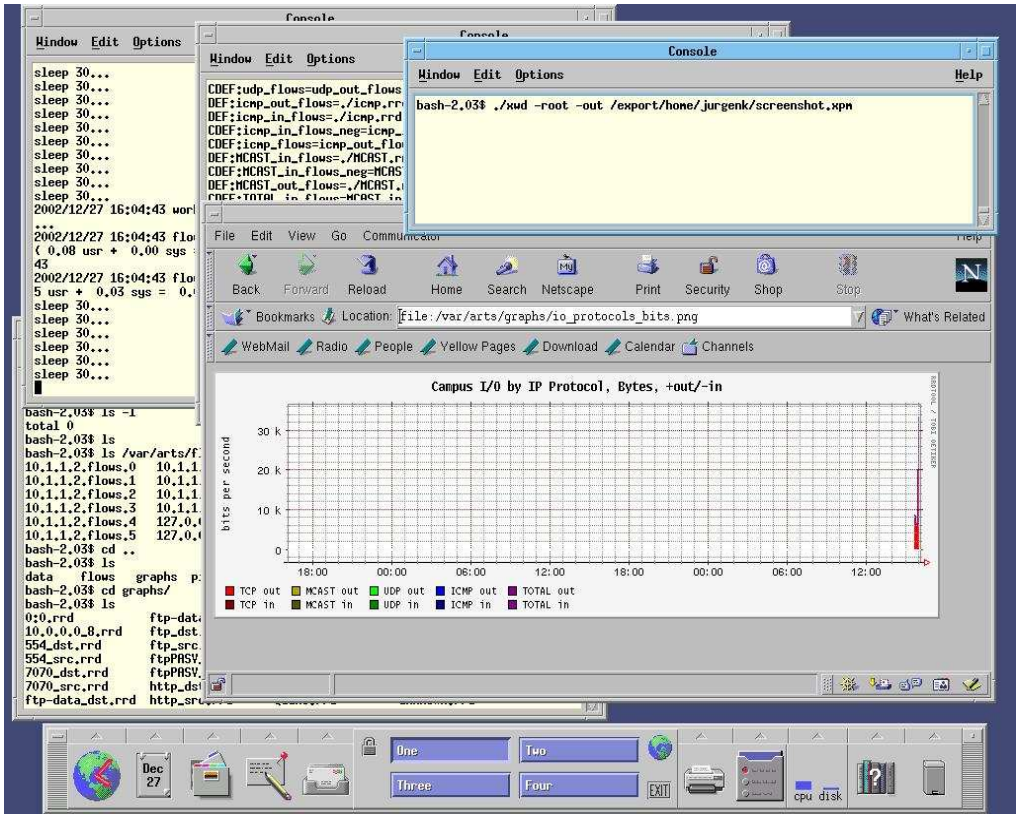
However, not everything in Linux works out at first. You must install Arts++ and Cflowd at first because these are the difficult ones. To compile Arts++ and Cflowd you **cannot** use GCC version 3.x! You must compile those packages with GCC 2.95.2 or GCC 2.95.3 (I've had succes with version 2.95.3, and not with 2.95.2, but my promoter had the opposite, so I suggest: If it doesn't work, try the other version)

To solve this problem you can download one of those versions of GCC. You must provide the –enable-shared flag with the ./configure script, issue a “make” and install it in its default path / usr/local. If you configure the sources of Arts++ and Cflowd, you must add /usr/local/bin before the PATH environment variable, like “export PATH=/usr/local/bin:$PATH”, and also provide the correct --libdir and --includedir flags to the right ones in /usr/local/lib and / usr/local/include.
Also when compiling you will notice the Arts++, cflowd binaries/libraries are very large. You can “strip” these files (see the FAQ at the end of this document)

If installing of Arts++ and/or Cflowd doesn't work out I suggest using Flow-tools flow-capture

If you install these tools, you will get shared objects located in /usr/local/lib and other directories. You must reconfigure the dynamic linker run-time libraries, or you would not be able to start daemons because the shared objects (or libraries, or how you call it) won't be found. Reconfiguring the library path in Linux can be done with modifying /etc/ld.so.config and running “ldconfig”. In Solaris library path reloading can be done with “crle” but this is a risky operation. When using this commend, you must try afterwards if you still can start CDE-applications or if you can resolve libraries to applications in /usr/dt/lib or /usr/ccs/lib with the “ldd”, before you leave CDE or Solaris, or you can lockout yourself to a plain terminal, you are warned! More instructions about this can be found in the troubleshooting section.

If you have any problems, and surely you will have problems if you install it on Solaris, I suggest you read the troubleshooting section at the end.



FlowScan on Solaris 8

After you have installed Arts++ and Cflowd or Flow-tools, you can install FlowScan. You don't need to install these packages at forehand, because FlowScan doesn't depend on these tools. However because FlowScan needs to read the flows dumped by Cflowd of Flow-tools' flow-capture Netflow capturing tool, you must install a cflow module corresponding to this Netflow capturing tool during FlowScan configuring.

The instructions for the setup of FlowScan are located in the included INSTALL file and are also online on the next URL: http://net.doit.wisc.edu/~plonka/FlowScan/INSTALL.html

FlowScan can be installed by untarring the sourcecode, and issuing a “./configure” in the unpacked directory. For each missing tool or module you must install the corresponding tool of module, until the configuration works. After configuring you don't need to compile FlowScan because it written fully in Perl, you only have to install it with “make install”.

You can at best make use of the Perl installation shell (perl -MCPAN -eshell) to install the perl modules, because installing modules with this shell will resolve dependencies to other modules automaticaly. If it doesn't work out, you can download and install these modules by hand. With the ConfigReader:DirectiveStyle module there is one catch. Because it is an old module the installer won't place the modules in a directory “ConfigReader” in the Perl lib directories, so you must create this directory by hand and move the files to this directory. (See troubleshooting section.)

This is what the ./configure shows at first:

```
[jurgenk@jurgenk FlowScan-1.006]$ ./configure
loading cache ./config.cache
checking host system type... i686-unknown-linux
```

```
checking for find... /usr/bin/find
checking for gzip... /usr/bin/gzip
checking for ksh... /usr/bin/ksh
checking for ln... /bin/ln
checking for ls... /bin/ls
checking for mkdir... /bin/mkdir
checking for perl... /usr/bin/perl
checking perl version... ok
checking for rm... /bin/rm
checking for rcs... /usr/bin/rcs
checking for rrdtool... /usr/local/rrdtool-1.0.38/bin/rrdtool
checking for sed... /bin/sed
checking for tar... /bin/tar
checking for touch... /bin/touch
checking for xargs... /usr/bin/xargs
checking for head... /usr/bin/head
checking for grep... /bin/grep
checking for mv... /bin/mv
checking for rm... (cached) /bin/rm
checking for cp... /bin/cp
checking for RRDs... yes
checking for Boulder::Stream... no
configure: error: Must be able to use Boulder::Stream!
```

Boulder::Stream is used for flowfiles reading. Now we will install this module with the CPAN shell. This shell must be configured when first runned. Configuring is easy because most values can be configured as default. The important values are the HTTP/FTP-proxy, the country and the nearest provider of the CPAN modules. CPAN stands for “Comprehensive Perl Archive Network” and is a online archive of most well-known Perl modules. It is mirrored world-wide. CPAN is configured while logged in as “root”:

```
[jurgenk@jurgenk FlowScan-1.006]$ su
Password:
root@jurgenk FlowScan-1.006]# perl -MCPAN -eshell

/usr/lib/perl5/5.8.0/CPAN/Config.pm initialized.


CPAN is the world-wide archive of perl resources. It consists of about
100 sites that all replicate the same contents all around the globe.
Many countries have at least one CPAN site already. The resources
found on CPAN are easily accessible with the CPAN.pm module. If you
want to use CPAN.pm, you have to configure it properly.

If you do not want to enter a dialog now, you can answer 'no' to this
question and I'll try to autoconfigure. (Note: you can revisit this
dialog anytime later by typing 'o conf init' at the cpan prompt.)

Are you ready for manual configuration? [yes] yes

...
(1) Africa
(2) Asia
(3) Central America
(4) Europe
(5) North America
(6) Oceania
(7) South America
Select your continent (or several nearby continents) [] 4
Sorry! since you don't have any existing picks, you must make a
geographic selection.

(1) Austria
(2) Belgium
(3) Bulgaria
```

```
(4) Croatia
(5) Czech Republic
(6) Denmark
(7) Estonia
(8) Finland
(9) France
(10) Germany
(11) Greece
(12) Hungary
(13) Iceland
(14) Ireland
(15) Italy
(16) Latvia
15 more items, hit ENTER
Select your country (or several nearby countries) [] 2
Sorry! since you don't have any existing picks, you must make a
geographic selection.

(1) ftp://ftp.cpan.skynet.be/pub/CPAN
(2) ftp://ftp.easynet.be/pub/CPAN/
(3) ftp://ftp.kulnet.kuleuven.ac.be/pub/mirror/CPAN/
Select as many URLs as you like,
put them on one line, separated by blanks [] 2

Enter another URL or RETURN to quit: []
New set of picks:
  ftp://ftp.easynet.be/pub/CPAN/

...

commit: wrote /usr/lib/perl5/5.8.0/CPAN/Config.pm
```

The line above is important: if you have trouble with configuring of the CPAN-shell, notable with entering wrong proxy settings, you can start over again by deleting this file. After configuring the CPAN-shell you get a CPAN-shell prompt, and start installing Boulder::Stream:

```
cpan> install Boulder::Stream
CPAN: Storable loaded ok
Fetching with LWP:
  ftp://ftp.easynet.be/pub/CPAN/authors/01mailrc.txt.gz
Going to read /root/.cpan/sources/authors/01mailrc.txt.gz
Fetching with LWP:
  ftp://ftp.easynet.be/pub/CPAN/modules/02packages.details.txt.gz
Going to read /root/.cpan/sources/modules/02packages.details.txt.gz
  Database was generated on Tue, 03 Dec 2002 03:53:47 GMT

  There's a new CPAN.pm version (v1.63) available!
  [Current version is v1.61]
  You might want to try
    install Bundle::CPAN
    reload cpan
  without quitting the current session. It should be a seamless upgrade
  while we are running...

Fetching with LWP:
  ftp://ftp.easynet.be/pub/CPAN/modules/03modlist.data.gz
Going to read /root/.cpan/sources/modules/03modlist.data.gz
Going to write /root/.cpan/Metadata
Running install for module Boulder::Stream
Running make for L/LD/LDS/Boulder-1.29.tar.gz
Fetching with LWP:
  ftp://ftp.easynet.be/pub/CPAN/authors/id/L/LD/LDS/Boulder-1.29.tar.gz
CPAN: Digest::MD5 loaded ok
Fetching with LWP:
  ftp://ftp.easynet.be/pub/CPAN/authors/id/L/LD/LDS/CHECKSUMS
```

```
Checksum for /root/.cpan/sources/authors/id/L/LD/LDS/Boulder-
1.29.tar.gz ok
Scanning cache /root/.cpan/build for sizes
Boulder-1.29/
Boulder-1.29/Boulder/
Boulder-1.29/Boulder/Swissprot.pm
Boulder-1.29/Boulder/Labbase.pm
Boulder-1.29/Boulder/Blast/
Boulder-1.29/Boulder/Blast/WU.pm
Boulder-1.29/Boulder/Blast/NCBI.pm
Boulder-1.29/Boulder/Omim.pm
Boulder-1.29/Boulder/Blast.pm
Boulder-1.29/Boulder/Unigene.pm
```

After installing Boulder::Stream you can exit the shell with Ctrl-D and check the FlowScan dependencies again using "./configure":

```
[jurgenk@jurgenk FlowScan-1.006]$ ./configure
loading cache ./config.cache
checking host system type... i686-unknown-linux
checking for find... /usr/bin/find
checking for gzip... /usr/bin/gzip
checking for ksh... /usr/bin/ksh
checking for ln... /bin/ln
checking for ls... /bin/ls
checking for mkdir... /bin/mkdir
checking for perl... /usr/bin/perl
checking perl version... ok
checking for rm... /bin/rm
checking for rcs... /usr/bin/rcs
checking for rrdtool... /usr/local/rrdtool-1.0.38/bin/rrdtool
checking for sed... /bin/sed
checking for tar... /bin/tar
checking for touch... /bin/touch
checking for xargs... /usr/bin/xargs
checking for head... /usr/bin/head
checking for grep... /bin/grep
checking for mv... /bin/mv
checking for rm... (cached) /bin/rm
checking for cp... /bin/cp
checking for RRDs... yes
checking for Boulder::Stream... yes
checking for Net::Patricia >= 1.010... no
configure: error: Must have Net::Patricia >= 1.010!
```

Net::Patricia is an important module. You need it for subnet matching. It contains a PATRICIA algorithm, written in C, for fast IP-address to subnet resolving. This is a example of how Perl can gets its speed: it depends on native in the C language written Perl-modules. For the case you don't know: C is the fastest language and used for operating system development. It is developed with the initial developement of UNIX.

```
[jurgenk@jurgenk FlowScan-1.006]$ su
Password:
[root@jurgenk FlowScan-1.006]# perl -MCPAN -eshell

cpan shell -- CPAN exploration and modules installation (v1.61)
ReadLine support available (try 'install Bundle::CPAN')
cpan> install Net::Patricia
CPAN: Storable loaded ok
Going to read /root/.cpan/Metadata
  Database was generated on Tue, 03 Dec 2002 03:53:47 GMT
Running install for module Net::Patricia
Running make for P/PL/PLONKA/Net-Patricia-1.010.tar.gz
CPAN: LWP::UserAgent loaded ok
Fetching with LWP:
```

```
  ftp://ftp.easynet.be/pub/CPAN/authors/id/P/PL/PLONKA/Net-Patricia-
1.010.tar.gz
CPAN: Digest::MD5 loaded ok
Fetching with LWP:
  ftp://ftp.easynet.be/pub/CPAN/authors/id/P/PL/PLONKA/CHECKSUMS
Checksum for /root/.cpan/sources/authors/id/P/PL/PLONKA/Net-Patricia-
1.010.tar.gz ok
Scanning cache /root/.cpan/build for sizes
Net-Patricia-1.010/
Net-Patricia-1.010/typemap
Net-Patricia-1.010/libpatricia/
Net-Patricia-1.010/libpatricia/patricia.c
Net-Patricia-1.010/libpatricia/patricia.h
Net-Patricia-1.010/libpatricia/Makefile.PL
Net-Patricia-1.010/libpatricia/copyright
Net-Patricia-1.010/libpatricia/credits.txt
Net-Patricia-1.010/Patricia.xs
Net-Patricia-1.010/MANIFEST
Net-Patricia-1.010/Patricia.pm
Net-Patricia-1.010/Makefile.PL
Net-Patricia-1.010/COPYING
Net-Patricia-1.010/Changes
Net-Patricia-1.010/demo/
Net-Patricia-1.010/demo/demo.c
Net-Patricia-1.010/test.pl
Net-Patricia-1.010/README
```

After installing Net::Patricia you have to install ConfigReader::DirectiveStyle:

```
[root@jurgenk FlowScan-1.006]# exit
[jurgenk@jurgenk FlowScan-1.006]$ ./configure
loading cache ./config.cache
checking host system type... i686-unknown-linux
checking for find... /usr/bin/find
checking for gzip... /usr/bin/gzip
checking for ksh... /usr/bin/ksh
checking for ln... /bin/ln
checking for ls... /bin/ls
checking for mkdir... /bin/mkdir
checking for perl... /usr/bin/perl
checking perl version... ok
checking for rm... /bin/rm
checking for rcs... /usr/bin/rcs
checking for rrdtool... /usr/local/rrdtool-1.0.38/bin/rrdtool
checking for sed... /bin/sed
checking for tar... /bin/tar
checking for touch... /bin/touch
checking for xargs... /usr/bin/xargs
checking for head... /usr/bin/head
checking for grep... /bin/grep
checking for mv... /bin/mv
checking for rm... (cached) /bin/rm
checking for cp... /bin/cp
checking for RRDs... yes
checking for Boulder::Stream... yes
checking for Net::Patricia >= 1.010... yes
checking for ConfigReader::DirectiveStyle... no
configure: error: Must be able to use ConfigReader::DirectiveStyle!
```

The installation of ConfigReader::DirectiveStyle module does not work out-of-the-box:
This module is quite old and designed while OOP-programming in Perl was in its early stages. As a result the CPAN-module will install the ConfigReader::DirectiveStyle.pm files directly in the Perl-lib directories, but this is not the way Perl-classes works. For each object-class (like ConfigReader) there has to be a directory created in its Perl-lib directory. This is not done by the CPAN-module or the install script of the ConfigReader::DirectiveStyle module:

```
cpan> install ConfigReader::DirectiveStyle
CPAN: Storable loaded ok
Going to read /root/.cpan/Metadata
  Database was generated on Tue, 03 Dec 2002 03:53:47 GMT
Running install for module ConfigReader::DirectiveStyle
Running make for A/AM/AMW/ConfigReader-0.5.tar.gz
CPAN: LWP::UserAgent loaded ok
Fetching with LWP:
  ftp://ftp.easynet.be/pub/CPAN/authors/id/A/AM/AMW/ConfigReader-
0.5.tar.gz
CPAN: Digest::MD5 loaded ok

...

Running make install
Installing /usr/lib/perl5/site_perl/5.8.0/ConfigReader.pod
Installing /usr/lib/perl5/site_perl/5.8.0/DirectiveStyle.pm
Installing /usr/lib/perl5/site_perl/5.8.0/Values.pm
Installing /usr/lib/perl5/site_perl/5.8.0/Spec.pm
Installing /usr/share/man/man3/DirectiveStyle.3pm
Installing /usr/share/man/man3/ConfigReader.3pm
Installing /usr/share/man/man3/Values.3pm
Installing /usr/share/man/man3/Spec.3pm
```

You must create a ConfigReader directory in the directory /usr/lib/perl5/site_perl/ 5.8.0 and move the files DirectiveStyle.pm, Values.pm and Spec.pm to this directory:

```
[root@jurgenk FlowScan-1.006]# mkdir /
usr/lib/perl5/site_perl/5.8.0/ConfigReader
[root@jurgenk FlowScan-1.006]# mv /
usr/lib/perl5/site_perl/5.8.0/DirectiveStyle.pm /
usr/lib/perl5/site_perl/5.8.0/ConfigReader
[root@jurgenk FlowScan-1.006]# mv /
usr/lib/perl5/site_perl/5.8.0/Values.pm /
usr/lib/perl5/site_perl/5.8.0/ConfigReader
[root@jurgenk FlowScan-1.006]# mv /
usr/lib/perl5/site_perl/5.8.0/Spec.pm /
usr/lib/perl5/site_perl/5.8.0/ConfigReader
```

The next dependency is for Cflow. It is very important that you configure and install the correct Cflow module corresponding with the used flow collection tool cflowd/flow-tools. For more information see the setup instructions for flow-tools. (Problems with this causes one of the most current asked questions on the FlowScan mailing lists).

```
[root@jurgenk FlowScan-1.006]# ./configure
loading cache ./config.cache
checking host system type... i686-unknown-linux
checking for find... /usr/bin/find
checking for gzip... /usr/bin/gzip
checking for ksh... /usr/bin/ksh
checking for ln... /bin/ln
checking for ls... /bin/ls
checking for mkdir... /bin/mkdir
checking for perl... /usr/bin/perl
checking perl version... ok
checking for rm... /bin/rm
checking for rcs... /usr/bin/rcs
checking for rrdtool... /usr/local/rrdtool-1.0.38/bin/rrdtool
checking for sed... /bin/sed
checking for tar... /bin/tar
checking for touch... /bin/touch
checking for xargs... /usr/bin/xargs
checking for head... /usr/bin/head
checking for grep... /bin/grep
```

```
checking for mv... /bin/mv
checking for rm... (cached) /bin/rm
checking for cp... /bin/cp
checking for RRDs... yes
checking for Boulder::Stream... yes
checking for Net::Patricia >= 1.010... yes
checking for ConfigReader::DirectiveStyle... yes
checking for Cflow >= 1.024... no
configure: error: Must have Cflow >= 1.024!
```

This module can be downloaded from http://net.doit.wisc.edu/~plonka/Cflow/ or is included with the source code of flow-tools and can be compiled with "./configure", "make" and "make install".

```
[root@jurgenk download]# tar -ztvf Cflow-1.051.tar.gz
drwxr-xr-x dplonka/staff     0 2002-01-31 07:09:18 Cflow-1.051/
-r--r--r-- dplonka/staff 32647 2002-01-31 07:07:19 Cflow-1.051/Cflow.xs
-r--r--r-- dplonka/staff  2880 2001-02-21 18:47:24 Cflow-1.051/cflow5.h
-r--r--r-- dplonka/staff 28077 2002-01-31 02:08:21 Cflow-1.051/Cflow.pm
-r--r--r-- dplonka/staff    93 2001-03-23 21:54:58 Cflow-1.051/MANIFEST
-r--r--r-- dplonka/staff  2298 2002-01-11 23:23:52 Cflow-
1.051/Makefile.PL
-rw-r--r-- dplonka/staff 38267 2002-01-31 07:09:07 Cflow-1.051/Changes
-r--r--r-- dplonka/staff 17982 1996-12-17 21:59:17 Cflow-1.051/COPYING
-r--r--r-- dplonka/staff   925 2001-02-14 14:55:18 Cflow-1.051/test.pl
-r--r--r-- dplonka/staff  2674 2002-01-11 23:48:56 Cflow-1.051/README
-r--r--r-- dplonka/staff 23038 2002-01-31 07:09:00 Cflow-
1.051/flowdumper.PL
[root@jurgenk download]# tar -zxvf Cflow-1.051.tar.gz
Cflow-1.051/
Cflow-1.051/Cflow.xs
Cflow-1.051/cflow5.h
Cflow-1.051/Cflow.pm
Cflow-1.051/MANIFEST
Cflow-1.051/Makefile.PL
Cflow-1.051/Changes
Cflow-1.051/COPYING
Cflow-1.051/test.pl
Cflow-1.051/README
Cflow-1.051/flowdumper.PL
[root@jurgenk download]# cd Cflow-1.051
[root@jurgenk Cflow-1.051]# ls
cflow5.h  Cflow.pm  Cflow.xs  Changes  COPYING  flowdumper.PL
Makefile.PL  MANIFEST  README  test.pl
[root@jurgenk Cflow-1.051]# perl Makefile.PL
Checking if your kit is complete...
Looks good
Writing Makefile for Cflow
[root@jurgenk Cflow-1.051]# make install
cp Cflow.pm blib/lib/Cflow.pm
AutoSplitting blib/lib/Cflow.pm (blib/lib/auto/Cflow)
/usr/bin/perl /usr/lib/perl5/5.8.0/ExtUtils/xsubpp  -typemap /
usr/lib/perl5/5.8.0/ExtUtils/typemap  Cflow.xs > Cflow.xsc && mv
Cflow.xsc Cflow.c
gcc -c   -D_REENTRANT -D_GNU_SOURCE -fno-strict-aliasing
-D_LARGEFILE_SOURCE -D_FILE_OFFSET_BITS=64 -I/usr/include/gdbm -O2
-march=i386 -mcpu=i686   -DVERSION=\"1.051\" -DXS_VERSION=\"1.051\"
-fpic "-I/usr/lib/perl5/5.8.0/i386-linux-thread-multi/CORE"   Cflow.c
```

At last we get the dependency HTML::Table. This module is needed for the Top-talkers reports, if you use it. Also the script checks for a 80/tcp service called "http" in /etc/services. Test with the next command if you have it:

```
$ perl -le "print scalar(getservbyport(80, 'tcp'))"
```

If not add a service http as 80/tcp in /etc/services.

```
[root@jurgenk FlowScan-1.006]# ./configure
loading cache ./config.cache
checking host system type... i686-unknown-linux
checking for find... /usr/bin/find
checking for gzip... /usr/bin/gzip
checking for ksh... /usr/bin/ksh
checking for ln... /bin/ln
checking for ls... /bin/ls
checking for mkdir... /bin/mkdir
checking for perl... /usr/bin/perl
checking perl version... ok
checking for rm... /bin/rm
checking for rcs... /usr/bin/rcs
checking for rrdtool... /usr/local/rrdtool-1.0.38/bin/rrdtool
checking for sed... /bin/sed
checking for tar... /bin/tar
checking for touch... /bin/touch
checking for xargs... /usr/bin/xargs
checking for head... /usr/bin/head
checking for grep... /bin/grep
checking for mv... /bin/mv
checking for rm... (cached) /bin/rm
checking for cp... /bin/cp
checking for RRDs... yes
checking for Boulder::Stream... yes
checking for Net::Patricia >= 1.010... yes
checking for ConfigReader::DirectiveStyle... yes
checking for Cflow >= 1.024... yes
checking for HTML::Table... no
configure: warning: Must be able to use HTML::Table for Top Talker
reports!
checking that service name for 80/tcp is http... yes
updating cache ./config.cache
creating ./config.status
creating Makefile
creating flowscan
creating graphs.mf
creating example/crontab
creating util/locker
creating util/add_ds.pl
creating util/add_txrx
creating util/event2vrule
creating util/ip2hostname
[root@jurgenk FlowScan-1.006]# perl -MCPAN -eshell

cpan shell -- CPAN exploration and modules installation (v1.61)
ReadLine support available (try 'install Bundle::CPAN')

cpan> install HTML::Table
```

After "`./configure`" works flawlessly you can install it with "`make install`":

```
[root@jurgenk FlowScan-1.006]# make install
test -d /usr/local/bin || /bin/mkdir -p /usr/local/bin
/home/jurgenk/download/FlowScan-1.006/install-sh -c flowscan /
usr/local/bin
/home/jurgenk/download/FlowScan-1.006/install-sh -c FlowScan.pm /
usr/local/bin
/home/jurgenk/download/FlowScan-1.006/install-sh -c CampusIO.pm /
usr/local/bin
/home/jurgenk/download/FlowScan-1.006/install-sh -c SubNetIO.pm /
usr/local/bin
/home/jurgenk/download/FlowScan-1.006/install-sh -c util/locker /
usr/local/bin
```

```
/home/jurgenk/download/FlowScan-1.006/install-sh -c util/add_ds.pl /
usr/local/bin
/home/jurgenk/download/FlowScan-1.006/install-sh -c util/add_txrx /
usr/local/bin
/home/jurgenk/download/FlowScan-1.006/install-sh -c util/event2vrule /
usr/local/bin
/home/jurgenk/download/FlowScan-1.006/install-sh -c util/ip2hostname /
usr/local/bin
[root@jurgenk FlowScan-1.006]#
```

2.4.2: Location/Configuring of Arts++, cflowd, RRDTool files

These are the default locations of the FlowScan dependent tools:

FlowScan, the report modules, and its configuration files are located in
`/usr/local/bin`

Arts++, cflowd & RRDTool are located in:
`/usr/local/arts/bin`
`/usr/local/arts/etc`
`/usr/local/rrdtool-...`

The corresponding header-files are located in:
`/usr/local/include`

libraries/shared object files are located in:
`/usr/local/lib`

You must first configure cflowd/cflowmux to capture Netflow data. You can do this by editing `/usr/local/arts/etc/cflowd.conf`

Also you have to configure the running Unix system to support a shared memory segment big enough for buffering Netflow records between the Cflowd daemons. In Solaris, as root place the following rule in /etc/system and restart the system:
`shmsys:shminfo_shmmax=16777216`. In Linux, run as root the following command: `#echo 16777216 >`
`/proc/sys/kernel/shmmax`

You must configure cflowd/cflowdmux from which flow exporters to receive netflow data, the shared memorysegment used by cflowd and where to dump flow files:

```
# vi /usr/local/arts/etc/cflowd.conf

OPTIONS {
  # syslog to local6 facility.
  LOGFACILITY:          local6
  # Listen for connections from cfdcollect on port 2056.
  TCPCOLLECTPORT:       2056
  # Use a 2 megabyte packet buffer in shared memory.
  PKTBUFSIZE:           2097152
  # Use /usr/local/arts/etc/cflowdtable.socket as named stream socket
  # for connections from local clients (cfdases et. al.)
  TABLESOCKFILE:        /var/flows/socket/cflowdtable.socket
  # Keep raw flow files in /usr/local/arts/data/cflowd/flows directory.
  FLOWDIR:              /var/flows/flows
  #  Each raw flow file should be 1000000 bytes in length.
  FLOWFILELEN:          1000000
  #  Keep 10 raw flow files per router.
  NUMFLOWFILES:         10
  #  Log total missed flows from a router if it exceeds 1000 between
  #  connections from cfdcollect.
```

```
  MINLOGMISSED:         1000
}

...

CISCOEXPORTER {
  HOST:         10.1.1.1        #  IP address of Cisco sending data.
  ADDRESSES:    { 10.1.1.1 }    #  Addresses of interfaces on Cisco
  CFDATAPORT:   2055            #  Port on which to listen for data.
  COLLECT:      { protocol, portmatrix, ifmatrix, nexthop, netmatrix,
                  asmatrix, tos, flows }
}
```

Now you can start cflowd/cflowdmux:

```
# cflowd -s 300 -O 0 -m /usr/local/arts/etc/cflowd.conf
# cflowdmux /usr/local/arts/etc/cflowd.conf
```

If cflowd complains with a invalid -s option it is a sign that the cflowd source was not patched. If so, patch and recompile cflowd.

2.4.3: Configuration of FlowScan

Inside the FlowScan distribution folder there is a cf directory located with the necessary configuration files. You must copy the flowscan.cf file to the /usr/local/bin folder where the flowscan script is located. The other configuration files are not needed for JKFlow. These are files for the included CampusIO and SubnetIO reporting modules.

```
# ls
CampusIO.html    config.log     flowscan      INSTALL.pod  README.pod
CampusIO.pm      config.status  flowscan.in   install-sh   SubNetIO.html
CampusIO.README  config.sub     FlowScan.pm   Makefile     SubNetIO.pm
cf               configure      graphs.mf     Makefile.in  SubNetIO.README
Changes          configure.in   graphs.mf.in  rc           TODO
config.cache     COPYING        INSTALL       README       util
config.guess     example        INSTALL.html  README.html  VERSION
[root@jurgenk FlowScan-1.006]# cd cf
[root@jurgenk cf]# ls
CampusIO.cf  local_nets.boulder       our_subnets.boulder
flowscan.cf  Napster_subnets.boulder  SubNetIO.cf
# cp flowscan.cf /usr/local/bin
```

Inside flowscan.cf you must configure were flowscan has to find the flowdump files and the reportmodule used for reporting. You must specify the JKFlow reporting class, and you may change the FlowFileGlob attribute to reflect the absolute path to the flowdump files:

```
# flowscan Configuration Directives
###########################################

# FlowFileGlob (REQUIRED)
# use this glob (file pattern match) when looking for raw flow files to
be
# processed, e.g.:
# FlowFileGlob /var/local/flows/flows.*:*[0-9]
FlowFileGlob /var/flows/flows/flows.*:*[0-9]

# ReportClasses (REQUIRED)
# a comma-seperated list of FlowScan report classes, e.g.:
# ReportClasses CampusIO
# ReportClasses SubNetIO
ReportClasses JKFlow

# WaitSeconds (OPTIONAL)
```

```
# This should be <= the "-s" value passed on the command-line to
cflowd, e.g.:
# WaitSeconds 300
WaitSeconds 30

# Verbose (OPTIONAL, non-zero = true)
Verbose 1
```

FlowScan can also archive processed flows. To configure this add a “saved” directory in the flow directory. This would be in our case:

`/var/flows/flows/saved`

Every processed file will be moved into this directory, after parsing. I've use this feature of FlowScan a lot while debugging JKFlow.

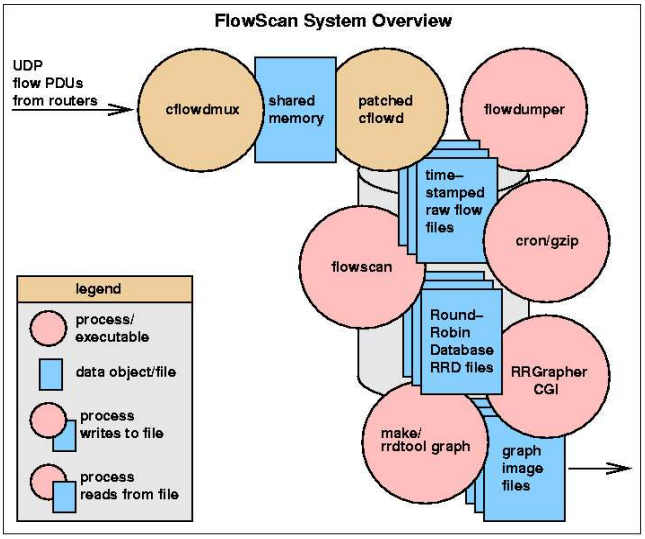
2.4.4: Installing JKFlow

JKFlow.pm and JKFlow.xml should be located in /usr/local/bin, like FlowScan in the default installation. If not, you will have to change the path to JKFlow.xml in JKFlow.pm because it assumes JKFlow is located in /usr/local/bin. JKGrapher.pl has to be copied in the cgi-bin directory of your webserver, normaly is this /var/www/cgi-bin. Also when the RRD directory is located in a different directory than /var/flows/report/rrds, you will have to change it in the JKGrapher.pl source too. Also note that the perl modules Net::Patricia, XML::Simple and HTML::Table are needed.

When installing on a system using SELinux (like Fedora 2, 3, REHL4, CentOS4), you may have to enable CGI binaries in Apache. This can be configured using a boolean flag. If things don't work out, you can also disable SELinux with the command 'setenforce 0'.

2.4.5: Principe of FlowScan

Here is a scheme describing the principe design of FlowScan. Note that not every component in this scheme is not actually used: flowdumper, cron/gzip, CGI-RRGrapher en the “make” step at end are all optional components. However 3 processes are important: the cflowdmux / patched cflowd for the NetFlow capture (also optional, because they can be replaced) and the flowscan Perl-script for the analyse of the flowfiles. The modules that FlowScan uses for reading the flowfiles (Cflow), to write the RRDTool files (RRDs) and at last the report module, which is quite important, are not shown in this figure.



FlowScan is actually a bunch of Perl script to combine a flow collector, with a reporting module , to write the resulting data to RRDTool databases, and another script for generation of graphs from these RRDTool databases. In this way you can see FlowScan thus as a pipeline of processes to process all NetFlow data into a usable set of trendings. The individual functions are:

1. Collection of NetFlow data to flow files
2. Reading and analysing of these records from the flow files
3. Writing resulting data from the analyse to some form like a RRDTool database
4. Creation and presentation of graphs like with a CGI Perl-script

In the drawing above takes Cflowd de function of the flow collector. Actually this doesn't have to be: FlowScan reads the flow records with a perl module called Cflow, and if you use a different flow collector you should use the corresponding Cflow module for this collector.
By using the correct flowreading module you can configure FlowScan to monitor a different flow protocol:

Argus	http://www.qosient.com/argus/
Flow-tools	http://www.splintered.net/sw/flow-tools/
CAIDA's Cflowd	http://www.caida.org/tools/measurement/cflowd/
LFAPd	http://www.nmops.org/

FlowScan is a Perl-script which checks every 30 seconds if there are any closed flow files available, and read the flow records from those available. These flow record are handled by a report module, whichafter it deletes those.

The report module is responsible for the analyse of all flows, and keeping statistics for the whole flowfile. With this role, takes the report module a critical part in FlowScan: This modules determinates to which data there must be analysed for.
In FlowScan there are standard 2 report modules included:

- CampusIO
- SubnetIO

The report module writes the resulted statistics data to RRDTool databases, or possible another database. You can log the statistics in principe to whatever you want, but the report modules that I've seen always report to RRDTool databases. Not supprising, because the RRDTool database is also very flexible for the generation of graphs.

Standard is in FlowScan a “Makefile”-script provided, which generates the graphs by using a “make” command. This tool is not very handy for a user who want to browse the trending by web, because these graphs as semi-statical. You can add a cron script to update the graphs regulary, but the end user had no flexibility in selecting the right data. Therefor I've looked for CGI Perl-scripts, and I've written my own CGI-script, JKGrapher.pl

2.4.6: Report modules for FlowScan

Internal are 3 functions called from within the report module in FlowScan:

- *parse*: This function reads the configuration file, parses the content of the configuration, and initialise the data structures.
- *wanted*: This function processes every flow record, and updates counters for every flow record depending on the configuration.
- *report*: This function places every counter in the RRDTool databases.

The report module is configured separately from the FlowScan script. The report module processes every flow record and updates matching counters based on the configuration of the module. The module will write the counters to its corresponding RRDTool databases after processing of all records.

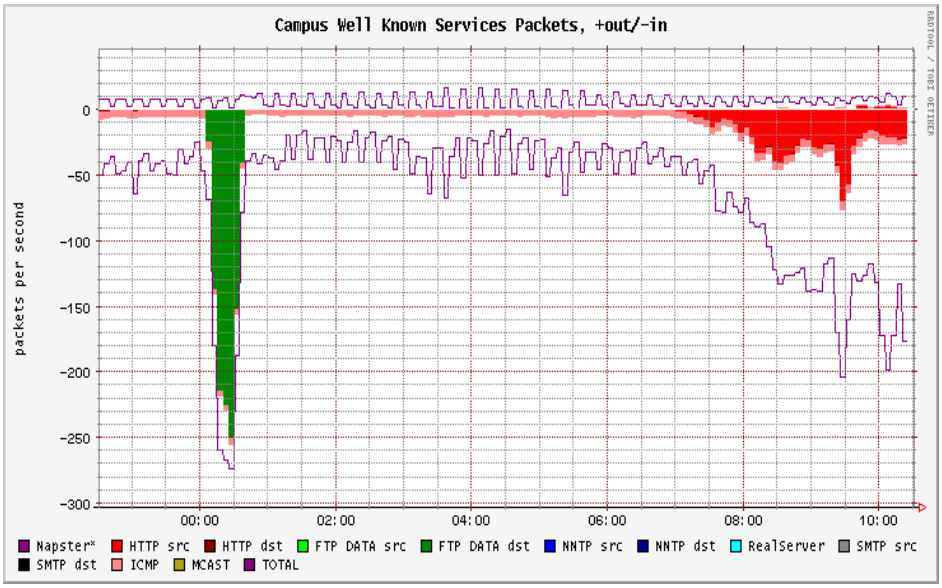
If you by example would like to monitor for 80/tcp or http, you simply add the configuration to the module and restart FlowScan. What happens is that the report module would start counting for webtraffic and after counting all records the module tries to open the corresponding RRDTool database, but doesn't find it, so it would create a new RRDTool database file, and starts writing webtraffic counters to it.

I didn't find any report module which allows me to write the counters to another database like MySQL, PostgreSQL or Oracle, but I looks very interesting to log the collected statistics with DBI to a database. Then I could generate graphs using the same DBI using the Gd-library or ImageMagick-tool. The option for using aggregation with SQL will become a option.

Internal within FlowScan there are 2 report modules available: **CampusIO** and **SubnetIO**

2.4.6.1: Campus IO (FlowScan)

CampusIO is a report module for the identification of the services and/or protocols passing over the network. Some extra feature are a statefull inspection on Passive FTP, Realaudio and Napster, and a Top10 webpage. Here's an example graph reported from of a CampusIO module.



A CampusIO Graph

2.4.6.2: Subnet IO (FlowScan)

Subnet IO is another report module which is delivered with FlowScan and is in comparison with CampusIO a lot simpler programmed. It basically only reports the total amount of traffic that is passing over a IP-subnet, and nothing more. This report module is only usable for subnet monitoring where no services of protocols monitoring is needed.

This module is very resticted in possibilities, and I didn't even bother testing it.

Beside the standard report modules CampusIO and SubnetIO, there are other report modules written by other people available. I've spitted the Internet out in search for report modules and I've found 2 other modules: **CarrierIn** and **CUFlow**

2.4.6.3: CarrierIn (Cablecon)

CarrierIn is a report module written for Internet Service Providers, and is developed by Stanislav Sinyagin for Cablecom in China, and the module is based on CampusIO.pm. The differences versus CampusIO are:

- Only inbound traffic is counted
- Everything is counted in bits and not in bytes
- There is no state monitoring, like FTP, Napster, Realvideo
- Support of flow sampling
- Only inbound traffic is counted.
- Subnet can be nested.

CarrierIn leave some functionality behind to make some gains in speed. You can use this module for monitoring of services and protocols, but only globaly like CampusIO. You “can” nest subnets, but while you configure nesting subnets, in reality the flow matching is done on a linear list. (see code)

```
foreach my $ptrie ( @CarrierIn::subnets_pt )
    {
       my $name = $ptrie->match_integer($dstaddr);
       if( $name )
       {
           $cargo = $CarrierIn::subnets_nm{$name};

           # keep inbound totals for subnet...
           $cargo->{bits} += $CarrierIn::bits;
           $cargo->{pkts} += $pkts;

           my $hr; # hashref to dst host stats
           if( !($hr = $cargo->{dst_pt}->match_integer($dstaddr)))
           {
              $hr = $cargo->{dst_pt}->add_string($dstip,
                                          { addr => $dstip,
                                            bits => 0,
                                            pkts => 0 });
              die unless ref($hr);
           }
           # keep stats by dst address within subnet
           $hr->{bits} += $CarrierIn::bits;
           $hr->{pkts} += $pkts;
           $identified = 1;
       }
    }
```

2.4.6.4: CUFlow (Columbia University)

CUFlow can be downloaded from this URL:
http://www.columbia.edu/acis/networks/advanced/CUFlow/CUFlow.html

The description of CUFlow on the website of Columbia University:

"CUFlow.pm is a FlowScan module designed to combine the features of CampusIO and SubNetIO as we needed them and also hopefully process data more quickly. CUFlow allows you to differentiate traffic by protocol, service, TOS, router,and network and then generate TopN reports over 5 minutes periods and over an extended period of time. CUGrapher is the companion graphing tool and is designed as a CGI which generates images on the fly based on user input with data supplied by CUFlow."

CUFlow is very interesting report module:

- You can monitor several routers on a set of protocols, services, type-of-services or the total amount of traffic.
- You can define networks (these are groupings of subnets) to monitor the total amount of traffic.
- You can the module let generate webpages, like a Top10.
- A easy usable CGI-script CUGrapher.pl is added, which provide a lot of flexibility in the generation of graphs.

Although I very much liked the module, after some testing, I've noticed some shortcomings in CUFlow:

- You can monitor the total amount of network traffic on subnets, but not specific for services, protocols, type-of-services. Basically this module was meant for monitoring of separate routers. I was in the situation where the network traffic of whole Europe passes

a group of 3-4 routers, but didn't had the possibility to split the traffic based on the router exporter IP to its matching country.

- Each router is monitored on the same set of protocols and services. If I would like to monitor the X400 Exchange directory replication traffic between 2 countries, it would imply that I have to monitor whole Europe for this X400 traffic.
- CUGrapher shows in its CGI-forms all routers, with all protocols, services, etc... If I would define several routers, it would imply that I have to navigate through a impressive webform.

Because there was no report module available for the features that I wanted I've decided to write a report module myself. My initial decision was to simply add the subnet monitoring functionality and leave the rest untouched.

By some unfortunate pursuit of accidents I was however noticed by my employer of a termination of my contract, I don't want to tell you the full details of it (*), but I had suddenly much spare time at hand :-(

This changed for me that I could implement a lot of stuff in CUFlow, which would not only improve the module a little, but it would change the whole internal structure. After that I've noticed that the rewrite was becoming a fully new module, I've decided to rename the module to JKFlow. I've written a separate chapter on JKFlow, because a lot of the project was spend on writing of JKFlow.

(*: I can however poke a lot of fun by telling that the manager who was responsible for the termination, was called one month later by the British Army on duty to fight in Irak, yes it's really true!)

3: JKFlow (Jurgen Kobierczynski)

JKFlow is a self written report module on basis of CUFlow, and
can be downloaded from my website located on these URL's:
http://users.telenet.be/jurgen.kobierczynski or http://jkflow.sourceforge.net

The code that I wrote is very tight modular, en structured, and allows for modification.
I've solved the problems that I had with CUFlow. Here is a comparison:

1:
-CUFlow can only split flows on basis of router exporters.
-JKFlow can also split the flows on basis of subnets.

2:
-CUFlow monitors protocols and services globally defined on every router.
-JKFlow makes it possible to monitor a different set of protocols and services on every router, on every subnet, and globaly .

3:
-CUFlow can only monitor subnets or networks on the total amount of traffic.
-JKFlow introduces directions, which allows definitions of source subnets and destination subnets, so it is possible to monitor on specific carriers or destinations. These directions can be

defined recursively defined, and the recursion is used during the evaluation of flowrecord. Only the flowrecords matching the upper subnet/router/direction are evaluated. In each direction you can define a different set of protocols and services.

4:
-CUFlow is configured with “Directives” in CUFlow.cf.
-JKFlow is configured with XML in JKFlow.xml, and allows easy definition of subnetted directions, with on each direction a different set of protocols and services to monitor. Thanks to the XML the configuration is very structured and writing the parsing code of the XML configuration file was a breeze.

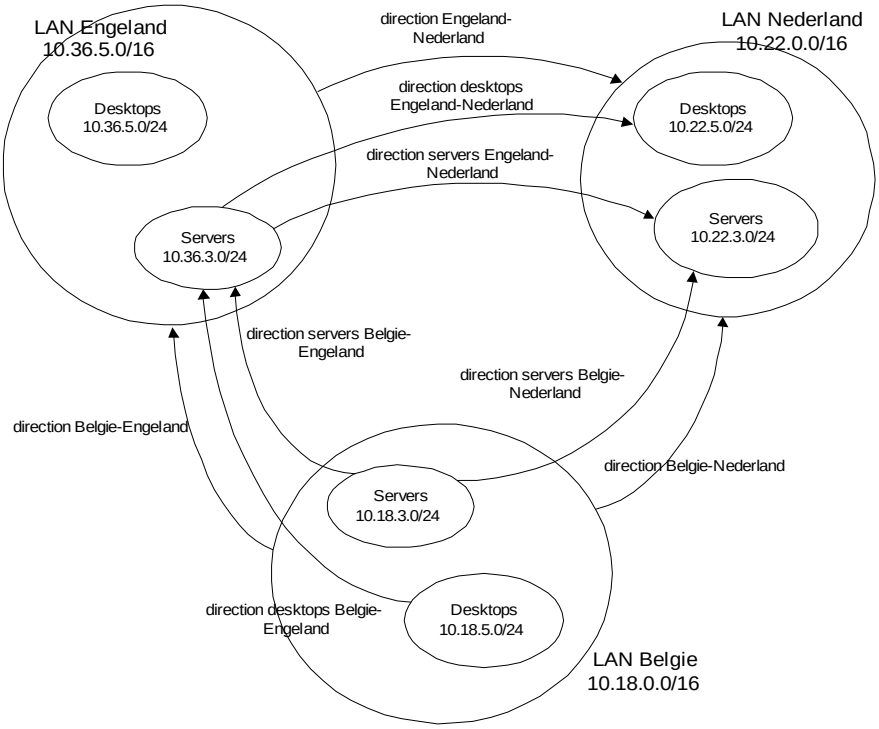
5:
-CUGrapher doesn't allow you for selection of routers, so you have to navigate a huge webform if you have a lot of routers.
-JKGrapher allows choosing which subnets/routers/directions you want to select trending from.

6:
-CUGrapher don't let you any choise between “stacked” and “unstacked” graphs.
-JKGrapher lets you to select to view the protocols, services, TOS and total in the graphs stacked and unstacked. Also protocols,services, TOS and total doesn't stack on eachother.

3.1: JKFlow directions explained

The next example will give you a good idea what you can do with the JKFlow report module:

A English company has a WAN-network connecting 3 brach offices between England, Netherlands, and Belgium. The networkadministrator of England wants to know how much network traffic is passing over the carriers to its neighbor offices in Netherland and Belgium, including specific services like Windows networking and Echange mail traffic. Futher he want's to split out the network traffic generated between the servers and the traffic of clients connecting to a server over a carrier (like remote workers, sales on trip).



example of a WAN network

You could define 2 subnets, or 2 routers in this situation, in my situation I choosed the England and Belgium subnets. Therefor I would need the flow exports of the VPN-routers of England and Belgium.

Then I would define “directions” within those subnets or routers. These are sections of source and destination subnets. In England I have defined a direction “England-Netherlands” as source subnet 10.36.0.0/16 and destination subnet 10.22.0.0/16. In Belgium I have defined 2 directions, one called “Belgium-England” as source subnet 10.18.0.0/16 and destination subnet 10.36.0.0/16, and one called Belgium-Netherlands as source subnet 10.18.0.0/16 and destination subnet 10.22.0.0/16

Directions can be nested.

Within England I have defined 2 directions in the direction England-Netherlands: one for the servers communicating between England and Netherlands, and one for the Dutch clients connecting to the servers in England. Because I want to define all directions in England as origin I have defined the direction “Servers England-Netherlands” with source subnet 10.22.3.0/24 and destination subnet 10.33.3.0/24, and I have defined the direction “Desktops England-Netherlands” with source subnet 10.22.3.0/24 and destination subnet 10.33.5.0/24.

Within Belgium I have defined 2 directions in the direction “Belgium-England”: one for de servers communicating between Belgium and England, and one for the Belgian clients connecting to the England servers. Here, I've took Belgium as origin and defined a direction “Servers Belgium-England” as source subnet 10.18.3.0/24 and destination subnet 10.22.3.0/24 and a direction “Desktops Belgium-England” as source subnets 10.18.5.0/24 and destination subnet 10.22.3.0/24

Also within Belgium I wanted to monitor the traffic between the servers in Belgium and Netherlands. Here I've defined in the direction “Belgium-Netherlands” the direction “Servers Belgium-Netherlands” as source subnet 10.18.3.0.24 and destination subnet 10.22.3.0/24.

Within each direction you can define a different set of protocols, services, TOS, and total. For desktops I would monitor Windows networking, Webbrowsing, FTP, etc... For servers I would rather monitor Windows networking, WINS, Exchange X400, DHCP, FTP, etc...

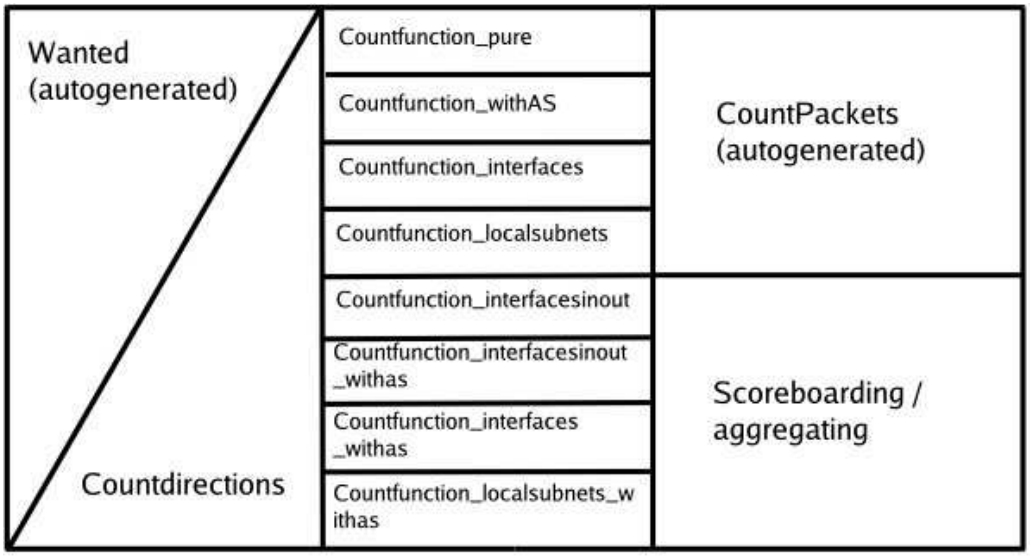
3.2: JKFlow configuration rules

Important: JKFlow has been improved a lot. There are 3 major releases: JKFlow version 1, 2 & 3. I recommend to use JKFlow version 3 only, because performance and configuration improvements were done. With the improvements, changes and additions in the configuration rules were neccesary.

Structure of the direction matching code

This picture outlines the internal working of JKFlow (updated to version 3.4.1) related to it's configuration. Understanding this helps you in configuring JKFlow. A mayor part in understanding lies in to understanding the difference between the logic to split traffic to directions, and the logic to measure services/protocols/tos and to maintain the scoreboarding inside the directions. This reflects directly in the configuration rules where services, protocols, tos, total, ftp and other elements are always defined inside directions, possible with aid of definsets and sets. Sets are templates of services/protocols/tos/total/ftp/etc...

To lookup these directions as fast as possible, I have programmed Countdirections wich calls the countPackets on basis of source/destination IP-address to subnets to directions Net::Patricia matching. This works as described for source/destination subnets. Combining this functionality with the matching of flows on basis of routers, interfaces, autonomous systems too is however not easy. Introducing such matching code into source/destination subnets matching code would lay a heavy burden on the performance of the code, so another way is implemented. In the picture below there is a second layer between the “Countdirections” and the “CountPackets”, consisting out of “Countfunction_*” functions. These functions will process every matched direction on further constraints defined within every direction.



This is not actually not complete: what to do when no source/destination subnets/or sites where defined within a direction? In the wanted function is this solved by evaluating these directions using these Countfunction_* functions direction outside Countdirections. So the whole picture consists of Wanted calling Countdirections calling Countfunctions calling Countpackets, Wanted calling Countfunctions calling Countpackets, Wanted calling Countdirections calling

CountPackets and at last Wanted calling CountPackets. The whole structure on what has to call what is defined at startup using autogeneration of the wanted code, and registrating the references of the correct Countfunction on the directions. The Countfunctions at last calls the Countpackets which are also autogenerated on startup for every direction, so asymetric configurations with directions with just </all> and others with lots of services, protocols, tos, etc... are processed as fast as possible.

These measures explains why JKFlow, while so flexible, is still very efficient in processing.

On XML

The JKFlow.xml configuration file has to be written in correct XML. When I speak about elements, I mean XML elements, and thus I mean the correct syntax is to define XML elements, like this: <tag> ... </tag> or <tag/>. Also when I speak about attributes I means here the additional value=”xxx” definitions inside the tags like <tag value=”xxx”> ... </tag>. Sometimes I receive mails asking questions about error messages within XML::Simple, which in fact are the results of invalid XML. Writing basic XML is fairly easy, and there are plenty of resources on the Internet available.

Overall Structure

The real cup of configuring JKFlow.xml is not to write valid XML, but to follow the DTD or schema used to interpret this XML file. Over time this schema has changed in making JKFlow more efficients and efficient, so when configuring try to upgrade to the latest version.

JKFlow.xml contains a root element <config>, in which the elements <definesets>, <sites>, <routergroups>, <all>, <directions>, <rrddir>, <scoredir>, <sampletime> may be defined. The element <scoredir> is mandatory, but you will have to add extra elements to make use of JKFlow. I added here a basic skeleton JKFlow.xml layout:

```
<config>
        <definesets>
                ...
        </definesets>
        <sites>
                ...
        </sites>
        <routergroups>
                ...
        </routergroups>
        <all>
                ...
        </all>
        <directions>
                ...
        </directions>
        <rrddir> ... </rrddir>
        <scoredir> ... </scoredir>
        <sampletime> ... </sampletime>
</config>
```

Inside the definesets, sites, routergroups, all and directions elements you add further elements, and that the reason why the opening and closing tag of the elements are put over multiple lines. The rrddir, scoredir and sampletime elements are written a single line.

A Very Basic JKFlow.xml

I begin with a very basic configuration. This config doesn't define any directions, but only a overall “all” , capturing every flow. This configuration is intended to show the various services monitoring elements you can define in every direction. The elements are in this basic example the blue elements located between the <all ...> <all/> elements.

A very basic JKFlow.xml example:

```
<config>
  <all localsubnets="10.0.0.0/8">
    <application name="web">80/tcp,8080/tcp</application>
    <application name="secureweb">443/tcp</application>
    <application name="mailreading">110/tcp,143/tcp</application>
    <application name="windows">137-139/tcp,137-139/udp</application>
    <application name="dns">53/udp,53/tcp</application>
    <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
    <protocols>tcp,udp,icmp</protocols>
    <otherservices/>
    <otherprotocols/>
    <multicast/>
    <ftp/>
    <tos/>
    <total/>
  </all>
  <rrddir>/var/flows/reports/rrds</rrddir>
  <scoredir>/var/flows/score</scoredir>
  <sampletime>300</sampletime>
</config>
```

The service monitoring elements possible are:

Services: Define the services to monitor, each service is monitored separate. You may define the <services> element which contains a comma separated list of protocols to enable logging of services. You may also define rages using "-", but be cautious with defining large ranges. You may use service names defined in /etc/services.

```
<services>25/tcp,119/tcp</services>
```

Application: Group several services together, and are reported as a single service. You may define one or more <application> elements which contains a comma separated list of services which you want to account as a single service. You may define ranges using "-". The name of this "service" is defined with the "name" attribute.

```
<application name="webbrowsing">www/tcp,8080/tcp</application>
<application name="mailreading">110/tcp,143/tcp</application>
<application name="windows">137-139/tcp,137-139/udp</application>
<application name="dns">53/tcp,53/udp</application>
```

FTP: Monitors ftp sessions. You may specify a <ftp/> element to enable state monitoring for active/passive FTP-sessions for logging FTP. (Do not log ftp in services if you use this, or you will end up logging this service twice.)

```
<ftp/>
```

Protocols: Defines the protocols to monitor. You may define the <protocols> element in which contains a comma separated list of protocols to enable logging of protocols. You may use protocol names defined in /etc/protocols.

```
<protocols>tcp,udp,icmp,ospf</protocols>
<protocols>1,6,17,89</protocols>
```

Multicasts: Matches all multicasts as a protocol.

```
<multicast/>
```

Tos: Tos montoring. Only BE (Best Effort) and other is supported

```
<tos/>
```

DSCP: Expands Tos values monitoring to BE, EF, all Afxx, all CSx, and other.
Note: The RRDTool databases are not created at startup, but only when traffic is monitored with these DSCP values, so be carefull with diskspace when defining DSCP monitoring for a large number of directories!

```
<dscp/>
```

Otherservices: Group all non services/applications/ftp matched traffic as a separate service.

```
<otherservices/>
```

Otherprotocols: Group all non protocol matched traffic as a separate protocol.

```
<otherprotocols/>
```

Total: Total amount of traffic. (You **must** define the <total/> element, JKGrapher.pl use the total for percentage calculations!)

```
<total/>
```

Also you can define scorebording inside at this level, but these are explained later.

A Basic JKFlow.xml with subnets

This example introduce directions. Directions selects flows on source and/or destination criteria and writes resulting RRDTool files in separate directories. This example demonstrates a configuration with 2 source/destination subnets based directions. The directions are highlighted in red, The following attributes defines subnet based directions: `fromsubnets` , `tosubnets`, `nofromsubnets`, `notosubnets`:

```
<config>
  <directions>
    <direction name="Belgium-Holland" fromsubnets="10.10.0.0/16"
        tosubnets="10.20.0.0/16">
      <application name="web">80/tcp,8080/tcp</application>
      <application name="mailreading">110/tcp,143/tcp</application>
      <application name="dns">53/udp,53/tcp</application>
      <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
      <protocols>tcp,udp,icmp</protocols>
      <otherservices/>
      <ftp/>
      <total/>
    </direction>
    <direction name="Belgium-England" fromsubnets="10.10.0.0/16"
        tosubnets="10.30.0.0/16">
      <application name="web">80/tcp,8080/tcp</application>
      <application name="mailreading">110/tcp,143/tcp</application>
      <application name="dns">53/udp,53/tcp</application>
      <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
      <protocols>tcp,udp,icmp</protocols>
```

```
      <otherservices/>
      <ftp/>
      <total/>
    </direction>
  </directions>
  <rrddir>/var/flows/reports/rrds</rrddir>
  <scoredir>/var/flows/score</scoredir>
  <sampletime>300</sampletime>
</config>
```

You can group several subnets together, and negate subnets, like defining all traffic to Internet excepts mached subnets (see the last direction defining Internet as 0.0.0.0/0, excluding subnets 10.20.0.0/16 and 10.30.0.0/16):

```
<config>
  <directions>
    <direction name="Belgium-Holland" fromsubnets="10.10.0.0/16"
       tosubnets="10.20.0.0/16">
      <application name="web">80/tcp,8080/tcp</application>
      <application name="mailreading">110/tcp,143/tcp</application>
      <application name="dns">53/udp,53/tcp</application>
      <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
      <protocols>tcp,udp,icmp</protocols>
      <otherservices/>
      <ftp/>
      <total/>
    </direction>
    <direction name="Belgium-England" fromsubnets="10.10.0.0/16"
       tosubnets="10.30.0.0/16">
      <application name="web">80/tcp,8080/tcp</application>
      <application name="mailreading">110/tcp,143/tcp</application>
      <application name="dns">53/udp,53/tcp</application>
      <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
      <protocols>tcp,udp,icmp</protocols>
      <otherservices/>
      <ftp/>
      <total/>
    </direction>
    <direction name="Belgium-Internet" fromsubnets="10.10.0.0/16"
       tosubnets="0.0.0.0/0" notosubnets="10.20.0.0/16,10.30.0.0/16">
      <application name="web">80/tcp,8080/tcp</application>
      <application name="mailreading">110/tcp,143/tcp</application>
      <application name="dns">53/udp,53/tcp</application>
      <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
      <protocols>tcp,udp,icmp</protocols>
      <otherservices/>
      <ftp/>
      <total/>
    </direction>
  </directions>
  <rrddir>/var/flows/reports/rrds</rrddir>
  <scoredir>/var/flows/score</scoredir>
  <sampletime>300</sampletime>
</config>
```

Reusing monitored services using Definesets/Sets in JKFlow

This was quite easy, but when creating larger configurations with the same monitored services inside each direction you may ask yourself: “Could there be a more efficient way to define monitored services, than copying these over all these directions?”. Yes, you can define a “Set” using a “Defineset” element. You can define more definesets, and use them multiple times over all directions. Here is an example, the “Defineset” and “Set” elements are highlighted in orange:

```
<config>
  <definesets>
    <defineset name="Common Services">
      <application name="web">80/tcp,8080/tcp</application>
      <application name="secureweb">443/tcp</application>
```

```
      <application name="mailreading">110/tcp,143/tcp</application>
      <application name="windows">137-139/tcp,137-139/udp</application>
      <application name="dns">53/udp,53/tcp</application>
      <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
      <otherservices/>
      <ftp/>
      <protocols>tcp,udp,icmp</protocols>
      <multicast/>
      <otherprotocols/>
      <tos/>
      <total/>
    </defineset>
  </definesets>
  <directions>
    <direction name="Belgium-Holland" fromsubnets="10.10.0.0/16"
       tosubnets="10.20.0.0/16">
      <set name="Common Services"/>
    </direction>
    <direction name="Belgium-England" fromsubnets="10.10.0.0/16"
       tosubnets="10.30.0.0/16">
      <set name="Common Services"/>
    </direction>
    <direction name="Belgium-Internet" fromsubnets="10.10.0.0/16"
       tosubnets="0.0.0.0/0" notosubnets="10.20.0.0/16,10.30.0.0/16">
      <set name="Common Services"/>
    </direction>
  </directions>
  <rrddir>/var/flows/reports/rrds</rrddir>
  <scoredir>/var/flows/score</scoredir>
  <sampletime>300</sampletime>
</config>
```

You can even define sets inside other sets:

```
<config>
  <definesets>
    <defineset name="Common Services">
      <application name="web">80/tcp,8080/tcp</application>
      <application name="secureweb">443/tcp</application>
      <application name="mailreading">110/tcp,143/tcp</application>
      <application name="windows">137-139/tcp,137-139/udp</application>
      <application name="dns">53/udp,53/tcp</application>
      <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
      <total/>
      <set name="Other Services"/>
    </defineset>
    <defineset name="Other Services">
      <otherservices/>
      <ftp/>
      <protocols>tcp,udp,icmp</protocols>
      <multicast/>
      <otherprotocols/>
      <tos/>
    </defineset>
  </definesets>
  <directions>
    <direction name="Belgium-Holland" fromsubnets="10.10.0.0/16"
       tosubnets="10.20.0.0/16">
      <set name="Common Services"/>
    </direction>
    <direction name="Belgium-England" fromsubnets="10.10.0.0/16"
       tosubnets="10.30.0.0/16">
      <set name="Common Services"/>
    </direction>
    <direction name="Belgium-Internet" fromsubnets="10.10.0.0/16"
       tosubnets="0.0.0.0/0" notosubnets="10.20.0.0/16,10.30.0.0/16">
      <set name="Common Services"/>
    </direction>
  </directions>
  <rrddir>/var/flows/reports/rrds</rrddir>
  <scoredir>/var/flows/score</scoredir>
  <sampletime>300</sampletime>
</config>
```

Using sets doesn't exclude defining other services to monitor inside a direction:

```
<config>
  <definesets>
    <defineset name="Common Services">
      <application name="web">80/tcp,8080/tcp</application>
      <application name="mailreading">110/tcp,143/tcp</application>
      <application name="dns">53/udp,53/tcp</application>
      <services>22-23/tcp,25/tcp</services>
      <otherservices/>
      <protocols>tcp,udp,icmp</protocols>
      <multicast/>
      <otherprotocols/>
      <tos/>
      <total/>
    </defineset>
  </definesets>
  <directions>
    <direction name="Belgium-Holland" fromsubnets="10.10.0.0/16"
       tosubnets="10.20.0.0/16">
      <set name="Common Services"/>
      <application name="secureweb">443/tcp</application>
      <services>102/tcp,119/tcp</services>
      <ftp/>
    </direction>
    <direction name="Belgium-England" fromsubnets="10.10.0.0/16"
       tosubnets="10.30.0.0/16">
      <set name="Common Services"/>
      <application name="windows">137-139/tcp,137-139/udp</application>
      <services>119/tcp</services>
    </direction>
    <direction name="Belgium-Internet" fromsubnets="10.10.0.0/16"
       tosubnets="0.0.0.0/0" notosubnets="10.20.0.0/16,10.30.0.0/16">
      <set name="Common Services"/>
    </direction>
  </directions>
  <rrddir>/var/flows/reports/rrds</rrddir>
  <scoredir>/var/flows/score</scoredir>
  <sampletime>300</sampletime>
</config>
```

Make using subnets in JKFlow.xml easier with sites

This was quite easy, but what if you create larger configurations with a multitude of subnets? Entering these subnets and masks is hard work to remember. To make this task easier you can define “Sites” and use “from”/”to” (and also “nofrom”/”noto”) attributes in directories. The example here below shows an example highlighted in purple:

```
<config>
  <definesets>
    <defineset name="Common Services">
      <application name="web">80/tcp,8080/tcp</application>
      <application name="secureweb">443/tcp</application>
      <application name="mailreading">110/tcp,143/tcp</application>
      <application name="windows">137-139/tcp,137-139/udp</application>
      <application name="dns">53/udp,53/tcp</application>
      <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
      <otherservices/>
      <ftp/>
      <protocols>tcp,udp,icmp</protocols>
      <multicast/>
      <otherprotocols/>
      <tos/>
      <total/>
    </defineset>
  </definesets>
  <sites>
    <site name="Belgium" subnets="10.10.0.0/16"/>
    <site name="Holland" subnets="10.20.0.0/16"/>
    <site name="England" subnets="10.30.0.0/16"/>
    <site name="Internet" subnets="0.0.0.0/0"/>
```

```
  </sites>
  <directions>
    <direction name="Belgium-Holland"
               from="Belgium" to="Holland">
      <set name="Common Services"/>
    </direction>
    <direction name="Belgium-England"
               from="Belgium" to="England">
      <set name="Common Services"/>
    </direction>
    <direction name="Belgium-Internet" from="Belgium"
               to="Internet" noto="Belgium,Holland,England">
      <set name="Common Services"/>
    </direction>
  </directions>
  <rrddir>/var/flows/reports/rrds</rrddir>
  <scoredir>/var/flows/score</scoredir>
  <sampletime>300</sampletime>
</config>
```

Defining Autonomous Systems (AS) in JKFlow.xml

With JKFlow version 3.4.1 you can define AS-monitoring using fromas, toas attributes, as demonstrated in this example. You can use multiple AS'es, separated with commas.

```
<config>
  <definesets>
    <defineset name="Common Services">
      <application name="web">80/tcp,8080/tcp</application>
      <application name="secureweb">443/tcp</application>
      <application name="mailreading">110/tcp,143/tcp</application>
      <application name="windows">137-139/tcp,137-139/udp</application>
      <application name="dns">53/udp,53/tcp</application>
      <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
      <otherservices/>
      <ftp/>
      <protocols>tcp,udp,icmp</protocols>
      <multicast/>
      <otherprotocols/>
      <tos/>
      <total/>
    </defineset>
  </definesets>
  <directions>
    <direction name="provider1-provider2" fromas="1111" toas="2222">
      <set name="Common Services"/>
    </direction>
    <direction name="provider12-provider3" fromas="1111,2222" toas="3333">
      <set name="Common Services"/>
    </direction>
    <direction name="provider1-provider23" fromas="1111" toas="2222,3333">
      <set name="Common Services"/>
    </direction>
  </directions>
  <rrddir>/var/flows/reports/rrds</rrddir>
  <scoredir>/var/flows/score</scoredir>
  <sampletime>300</sampletime>
</config>
```

Samplerates

You can define samplerates in subnets. Samplerates can be defined on routers to decrease the load on the routers.

```
<config>
  <definesets>
    <defineset name="Common Services">
      <application name="web">80/tcp,8080/tcp</application>
      <application name="mailreading">110/tcp,143/tcp</application>
      <application name="dns">53/udp,53/tcp</application>
      <services>22-23/tcp,25/tcp</services>
      <protocols>tcp,udp,icmp</protocols>
```

```
      <total/>
    </defineset>
  </definesets>
  <directions>
    <direction name=”Belgium-Holland” fromsubnets=”10.10.0.0/16”
       tosubnets=”10.20.0.0/16” samplerate=”2”>
      <set name=”Common Services”/>
    </direction>
    <direction name=”Belgium-England” fromsubnets=”10.10.0.0/16”
       tosubnets=”10.30.0.0/16” samplerate=”3”>
      <set name=”Common Services”/>
    </direction>
  </directions>
  <rrddir>/var/flows/reports/rrds</rrddir>
  <scoredir>/var/flows/score</scoredir>
  <sampletime>300</sampletime>
</config>
```

Routergroup elements

Routergroups enables selecting flows based on router exporter ip or/and router interface number. You define a routergroup within a <routergroups> top elements. Inside the routergroup you define multiple router elements. These routers may define interfaces or localsubnets. Both types of router elements define exporter ip's. You may not combine interface router elements and localsubnets router elements within a single routergroup. This example demonstrates both types of routergroups:

```
<config>
  <definesets>
    <defineset name=”Common Services”>
      <application name=”web”>80/tcp,8080/tcp</application>
      <application name=”secureweb”>443/tcp</application>
      <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
      <protocols>tcp,udp,icmp</protocols>
      <ftp/>
      <tos/>
      <total/>
    </defineset>
  </definesets>
  <routergroups>
    <routergroup name="routers1">
      <router exporter="10.1.2.2" interface="3"/>
      <router exporter="10.1.2.2" interface="4"/>
      <router exporter="10.1.2.3" interface="3"/>
      <router exporter="10.1.2.3" interface="4"/>
    </routergroup>
    <routergroup name="routers2">
      <router exporter="10.1.2.5" localsubnets="10.0.0.0/8"/>
    </routergroup>
  </routergroups>
  <directions>
    <direction name="Routers-Group1" routergroup="routers1">
          <set name=”Common Services”/>
    </direction>
    <direction name="Routers-Group2" routergroup="routers2">
          <set name="Common Services"/>
    </direction>
  </directions>
  <rrddir>/var/flows/reports/rrds</rrddir>
  <scoredir>/var/flows/reports/score</scoredir>
</config>
```

Updated in 3.4.1: You may define multiple interfaces with the attribute interfaces:

```
    <routergroup name="routers1">
      <router exporter="10.1.2.2" interfaces="3,4"/>
      <router exporter="10.1.2.3" interfaces="3,4"/>
    </routergroup>
```

The first routergroup combines 2 routerinterfaces together in a routergroup “router1”. This group is assigned with the “routergroup” attribute in the direction “Routers-Group1”. In- and Outbound direction is seen from the router perspective. Notice that traffic is reported only once per router, on the ingress router interface, so it is important to enable netflow on all interfaces, to report all traffic.

The second routergroup takes the flow export from a whole router. Because all traffic is reported only once, no duplicate traffic is reported. Because no interface is defined, the Inbound/Outbound direction is found based on the “localsubnets” attribute. Traffic directed from the localsubnet is designated as outbound. Traffic directed to the localsubnet is inbound.

The different configuration possibilities allow you to configure the netflow monitoring infrastructure based on directions (from/to-subnets), router (subnet+localsubnet), or routerinterfaces (ingress/egress traffic)

Combinations of Routergroup, From/To/Fromsubnets/Tosubnets, AS attributes

It is possible to combine defining routergroups, from/nofrom, to, noto, fromsubnets, tosubnets, nofromsubnets, notosubnets, fromas and toas attributes in directions. This is used for avoiding reporting on redundant flows originating from different routers on traffic which is in fact caused by equal flows. Because Netflow doesn't correlate flows between routers, precautions must be made in avoiding counting twice or more reported flows.

Dropping of subnets/router/network elements in favor to directions/routergroup elements departs JKFlow from the distinct 'global' view of directions between subnets/router/network to multiple separate views: one sees network traffic from the prespective of source and direction subnets (directions), one sees network traffic from the perspective of the router interfaces (ingress/egress).

Here's an example of a configuration where 3 sites and 4 routergroups were defined. Notice the routergroups are different: 2 routergroups are composed out of router interfaces and the other 2 are composed of routers with localsubnets attributes:

```
<config>
  <definesets>
    <defineset name=”Common Services”>
      <application name=”web”>80/tcp,8080/tcp</application>
      <application name=”secureweb”>443/tcp</application>
      <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
      <protocols>tcp,udp,icmp</protocols>
      <ftp/>
      <tos/>
      <total/>
    </defineset>
  </definesets>
  <sites>
    <site name=”Belgium” subnets=”10.10.0.0/16” />
    <site name=”Holland” subnets=”10.20.0.0/16” />
    <site name=”France” subnets=”10.30.0.0/16” />
  </sites>
  <routergroups>
    <routergroup name="routers-to-holland">
      <router exporter="10.1.2.2" interface="1"/>
      <router exporter="10.1.2.3" interface="1"/>
    </routergroup>
    <routergroup name="routers-to-france">
      <router exporter="10.1.2.2" interface="2"/>
```

```
      <router exporter="10.1.2.3" interface="2"/>
    </routergroup>
    <routergroup name="router-isp1">
      <router exporter="10.1.2.2" localsubnets="10.10.0.0/16"/>
    </routergroup>
    <routergroup name="router-isp2">
      <router exporter="10.1.2.3" localsubnets="10.10.0.0/16"/>
    </routergroup>
  </routergroups>
  <directions>
    <direction name="Belgium-Holland"
       from=”Belgium to=”Holland” routergroup="routers-to-holland">
      <set name=”Common Services”/>
    </direction>
    <direction name="Belgium-France"
       from=”Belgium to=”France” routergroup="routers-to-france">
      <set name=”Common Services”/>
    </direction>
    <direction name="Belgium-isp1" routergroup="router-isp1">
      <set name=”Common Services”/>
    </direction>
    <direction name="Belgium-isp2" routergroup="router-isp2">
      <set name=”Common Services”/>
    </direction>
    <direction name="Belgium-Holland-isp1"
       from=”Belgium to=”Holland” routergroup="router-isp1">
      <set name=”Common Services”/>
    </direction>
    <direction name="Belgium-France-isp2"
       from=”Belgium to=”France” routergroup="router-isp2">
      <set name=”Common Services”/>
    </direction>
  </directions>
  <rrddir>/var/flows/reports/rrds</rrddir>
  <scoredir>/var/flows/reports/score</scoredir>
</config>
```

Look at the different combinations of from/to and routergroup attributes. They are all allowed, but what do they mean?

```
name="Belgium-Holland”        from=”Belgium to=”Holland”
                              routergroup="routers-to-holland" =>
                              exporter="10.1.2.2" interface="1"
                              exporter="10.1.2.3" interface="1"

name="Belgium-France”         from=”Belgium to=”France”
                              routergroup="routers-to-holland" =>
                              exporter="10.1.2.2" interface="2"
                              exporter="10.1.2.3" interface="2"
```

The first 2 directions are a combination of source/destination sites and a routergroup which exists of 2 router interfaces. This means that traffic between those 2 sites passing over these 2 router interfaces will be reported.

```
name="Belgium-isp1"           routergroup="routers-isp1" =>
                              exporter="10.1.2.2" localsubnets=”10.10.0.0/16”

name="Belgium-isp2"           routergroup="routers-isp2" =>
                              exporter="10.1.2.3" localsubnets=”10.10.0.0/16”
```

The third and fourth direction use only a routergroup. This means the all traffic passing the router is reported, and like in the first direction the in/outbound direction is determinated from the router prespective.

```
name="Belgium-Holland-isp1"   from=”Belgium to=”Holland”
                              routergroup="routers-isp1" =>
                              exporter="10.1.2.2" localsubnets=”10.10.0.0/16”
```

```
name="Belgium-France-isp2"     from=”Belgium to=”France”
                               routergroup="routers-isp2" =>
                               exporter="10.1.2.3" localsubnets=”10.10.0.0/16”
```

The fifth and sixth direction are a combination of source/destination subnets and a routergroup which exists of a router including a localsubnet. This means that traffic between those 2 sites passing over the router will be reported.

In JKFlow 3.4 the direction of flows is determinated on the ingress/egress direction of the interface. The in/outbound direction is NOT determinated by the from/to attributes of the sites/subnets. You MUST add the localsubnets attribute in case you define routers in your routergroups.

In JKFlow version 3.4.1 you can also combine AS with routergroups and sites/subnets. This means that there are now a whore range of combinations of which the flows directions can be different, depending on the ingress/egress direction of the interfaces, the direction of the from/to sites and subnets, and now also on the direction from/to AS. To make it easier I have split interfaces into a: interfaces without direction and b: interfaces with a direction. The interfaces with a outbound directions have the same functionality as the interfaces in JKFlow 3.4, and in directions using routergroups containing interfaces without directions, the direction is determinated on AS direction and then on source/destination site/subnets. Here is a table:

Combination	***Direction based on***	
Source+Destination Sites/Subnets	Source->Destination=OUT Destination->Source=IN	
Source+Destination Sites/Subnets and FromAS+ToAS	FromAS->ToAS=OUT ToAS->FromAS=IN	
Source+Destination Sites/Subnets and Interfaces_In+Interfaces_Out	Egress Interfaces_Out=OUT Ingress Interfaces_Out=IN Egress Interfaces_In=IN Ingress Interface_In=OUT	
Source+Destination Sites/Subnets and Interfaces_In+Interfaces_Out and FromAS+ToAS	Egress Interfaces_Out=OUT Ingress Interfaces_Out=IN Egress Interfaces_In=IN Ingress Interface_In=OUT	
Interfaces_In+Interfaces_Out	Egress Interfaces_Out=OUT Ingress Interfaces_Out=IN Egress Interfaces_In=IN Ingress Interface_In=OUT	
Interfaces (with localsubnets)	From Localsubnet =OUT To Localsubnet=IN	
Source+Destination Sites/Subnets and Interfaces	JKFlow 3.4 Egress Interface=OUT Ingress Interface=IN	JKFlow 3.4.1 Source->Destination=OUT Destination->Source=IN
FromAS+ToAS and Interfaces	FromAS->ToAS=OUT ToAS->FromAS=IN	
Source+Destination Sites/Subnets and Interfaces (with localsubnets)	From Localsubnet =OUT To Localsubnet=IN	

Combination	*Direction based on*
FromAS+ToAS and Interfaces (with localsubnets)	From Localsubnet =OUT To Localsubnet=IN
FromAS+ToAS	FromAS->ToAS=OUT ToAS->FromAS=IN

Here is a example of using interfaces, with and without direction:

```
<config>
  <definesets>
    <defineset name="Common Services">
      <application name="web">80/tcp,8080/tcp</application>
      <application name="secureweb">443/tcp</application>
      <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
      <protocols>tcp,udp,icmp</protocols>
      <ftp/>
      <tos/>
      <total/>
    </defineset>
  </definesets>
  <sites>
    <site name="Belgium" subnets="10.10.0.0/16" />
    <site name="Holland" subnets="10.20.0.0/16" />
    <site name="France" subnets="10.30.0.0/16" />
  </sites>
  <routergroups>
    <routergroup name="routers-to-holland">
      <router exporter="10.1.2.2" interfaces_out="1,2"/>
      <router exporter="10.1.2.3" interface_out="1"/>
    </routergroup>
    <routergroup name="routers-from-france">
      <router exporter="10.1.2.2" interface_in="2"/>
      <router exporter="10.1.2.3" interfaces_in="1,2"/>
    </routergroup>
  </routergroups>
  <directions>
    <direction name="Belgium-Holland"
        from="Belgium to="Holland" routergroup="routers-to-holland">
      <set name="Common Services"/>
    </direction>
    <direction name="Belgium-France"
        from="Belgium to="France" routergroup="routers-to-france">
      <set name="Common Services"/>
    </direction>
  </directions>
  <rrddir>/var/flows/reports/rrds</rrddir>
  <scoredir>/var/flows/reports/score</scoredir>
</config>
```

Scoreboarding

Scoreboard are delivered with directories containing HTML-files. The directory is configured using the <scoredir> element:

Example:

```
...
        <rrddir>/var/flows/reports/rrds</rrddir>
        <scoredir>/var/flows/score</scoredir>
</config>
```

Scoreboarding is splitted in 2 parts: host-scoreboarding & port-scoreboarding. In configuring scoreboarding, you activate those with the “hosts” and/or “ports” attribute with the value 1 to the scoreboard element. (the value 1 resembles nothing special, btw) In the scoreboard directory a directory tree will be created equivalent to the rrdtool directory. For every direction

a directory will be created, with a hosts and a ports directory depending on the defined hosts/ports attributes. Inside these directories JKFlow will create datestamped directories, and hours directories, and create html-files with the corresponding TOP-10 statistics. 12 tables are printed on each webpage:

-bit/sec src inbound	-bits/sec dst inbound
-bit/sec src outbound	-bits/sec dst outbound
-pkts/sec src inbound	-pkts/sec dst inbound
-pkts/sec src outbound	-pkts/sec dst outbound
-flows/sec src inbound	-flows/sec dst inbound
-flows/sec src outbound	-flows/sec dst outbound

Here is an example how to configure this:

```
<config>
  <definesets>
    <defineset name=”Common Services”>
      <application name=”web”>80/tcp,8080/tcp</application>
      <application name=”secureweb”>443/tcp</application>
      <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
      <protocols>tcp,udp,icmp</protocols>
      <ftp/>
      <tos/>
      <total/>
    </defineset>
  </definesets>
 <sites>
    <site name=”Belgium” subnets=”10.10.0.0/16” />
    <site name=”Holland” subnets=”10.20.0.0/16” />
    <site name=”France” subnets=”10.30.0.0/16” />
  </sites>
  <directions>
    <direction name="Belgium-Holland"
       from=”Belgium to=”Holland” routergroup="routers-to-holland">
      <set name=”Common Services”/>
      <scoreboard hosts="1" ports="1"/>
    </direction>
    <direction name="Belgium-France"
       from=”Belgium to=”France” routergroup="routers-to-france">
      <set name=”Common Services”/>
      <scoreboard hosts="1" ports="1"/>
      <scoreboardother hosts="1" ports="1"/>
    </direction>
  </directions>
  <rrddir>/var/flows/reports/rrds</rrddir>
  <scoredir>/var/flows/reports/score</scoredir>
</config>
```

The first direction contains a <scoredboard>element, the second direction contains a <scoreboard> and a <scoreboardother> element. The scoreboardother element works exactly like a scoreboard element, but will create statistics on all not-matched service traffic in a direction.

You can not only create scoreboards for individual flowfiles but also aggregate scoreboards over multiple flowfiles. See this example:

```
<config>
  <definesets>
    <defineset name=”Common Services”>
      <application name=”web”>80/tcp,8080/tcp</application>
      <application name=”secureweb”>443/tcp</application>
      <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
      <protocols>tcp,udp,icmp</protocols>
```

```
      <ftp/>
      <tos/>
      <total/>
    </defineset>
  </definesets>
 <sites>
    <site name=”Belgium” subnets=”10.10.0.0/16” />
    <site name=”Holland” subnets=”10.20.0.0/16” />
    <site name=”France” subnets=”10.30.0.0/16” />
  </sites>
  <directions>
    <direction name="Belgium-Holland"
       from=”Belgium to=”Holland” routergroup="routers-to-holland">
      <set name=”Common Services”/>
      <scoreboard hosts="1" ports="1">
        <report count="12" hostsbase="agghostshour" portsbase="aggportshour"/>
        <report count="72" hostsbase="agghosts6hour" portsbase="aggports6hour"/>
      </scoreboard>
      <scoreboardother hosts="1" ports="1">
        <report count="12" hostsbase="agghostshour" portsbase="aggportshour"/>
        <report count="72" hostsbase="agghosts6hour" portsbase="aggports6hour"/>
      </scoreboardother>
    </direction>
    <direction name="Belgium-France"
       from=”Belgium to=”France” routergroup="routers-to-france">
      <set name=”Common Services”/>
      <scoreboard hosts="1" ports="1">
        <report count="12" hostsbase="agghostshour" portsbase="aggportshour"/>
        <report count="72" hostsbase="agghosts6hour" portsbase="aggports6hour"/>
      </scoreboard>
      <scoreboardother hosts="1" ports="1">
        <report count="12" hostsbase="agghostshour" portsbase="aggportshour"/>
        <report count="72" hostsbase="agghosts6hour" portsbase="aggports6hour"/>
      </scoreboardother>
    </direction>
  </directions>
  <rrddir>/var/flows/reports/rrds</rrddir>
  <scoredir>/var/flows/reports/score</scoredir>
</config>
```

The first scoreboard elements will create aggregated scoreboard of all matched (defined in with the services/ftp elements) network traffic. The scoreboardother elements will match all non-matched services in the direction. The hosts=”1” and the ports=”1” attributes will let JKFlow to write the individual scoreboard files as well. The next example will not write individual scoreboard files, saving a lot of space:

```
<scoreboard>
    <report count="12" hostsbase="agghostshour" portsbase="aggportshour"/>
    <report count="72" hostsbase="agghosts6hour" portsbase="aggports6hour"/>
</scoreboard>
```

The count attribute will aggregate 12 and 72 times a flowfile. You can choose any amount, and the time boundaries of aggregate scoreboards are choosen to match date/hour boundaries. (You can modify these with a “offset” attribute. The “hostsbase” and “portsbase” attributes are used for naming the files. The next example sets the amount of values to keep in memory and the rows to write, so these are actually Top-20's. (Default is a Top10 page configured.)

```
<scoreboard>
    <report   count="12" hostsbase="agghostshour" portsbase="aggportshour"
              scorekeep="20" numkeep="100"/>
    <report   count="72" hostsbase="agghosts6hour" portsbase="aggports6hour"
              scorekeep="20" numkeep="100"/>
</scoreboard>
```

Attributes latesthosts/latestports updates a softlink to the latest scoreboard (unfortunaly not for the aggregated scoreboards jet.)

```
<scoreboard   hosts="1" ports="1"
              latesthosts="latesthosts.html" latestports="latestports.html">
```

```
    <report count="12" hostsbase="agghostshour" portsbase="aggportshour"/>
    <report count="72" hostsbase="agghosts6hour" portsbase="aggports6hour"/>
</scoreboard>
```

Update “Tuples” scoreboarding JKFlow 3.4.2 and 3.5:

Since JKFlow 3.4.2 you can define “tuples” within scoreboarding. Tuples are lists of flow attributes used for aggregating traffic. This can be srcip, dstip for host aggregation, and can be srcport, dstport for port aggregation. You can combine any number of attributes, and also use some other less used flow attributes. This is the list of available flowattributes:

(as defined in Cflow)

```
unix_secs, exporter, exporterip, srcaddr, srcip, dstaddr, dstip, input_if,
output_if, srcport, dstport, pkts, bytes, nexthop, nexthopip, startime,
start_msecs, endtime, end_msecs, protocol, tos, src_as, dst_as, src_mask,
dst_mask, tcp_flags, engine_type, engine_id, localtime, raw, reraw, Bps, pps,
TCPFlags, ICMPTypeCode, ICMPType, ICMPCode, duration_secs, duration_msecs
```

As it is possible to define aggregation on every attribute, you don't need to define port/host scoreboard anymore. This is a very basic example scoreboard config:

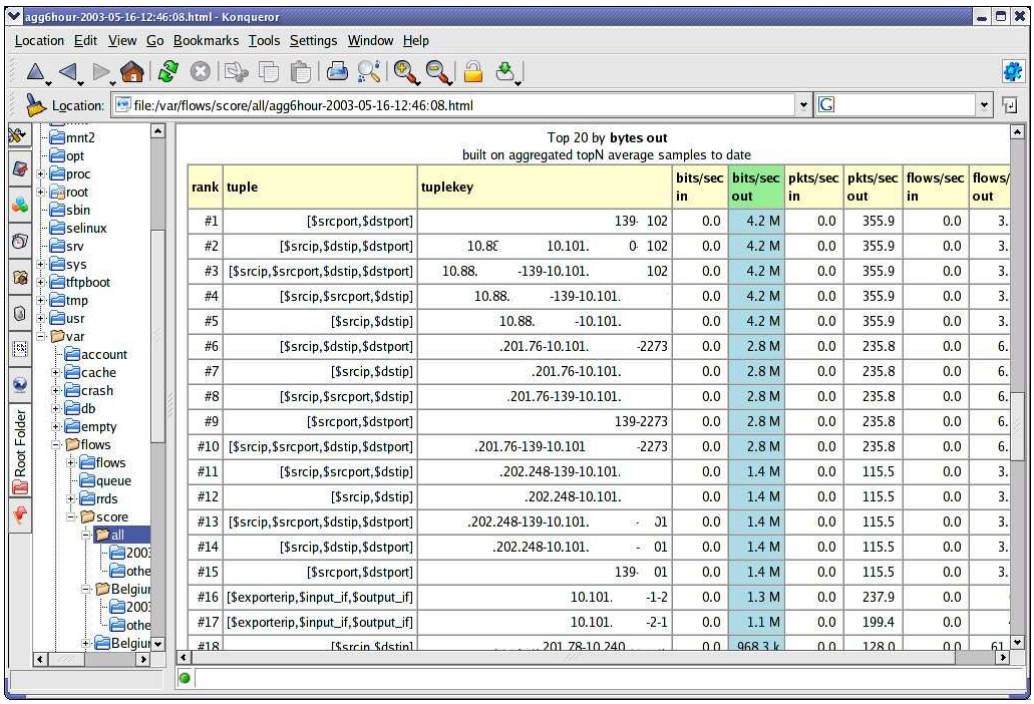
```
<scoreboard>
    <tuples>
         <tuple>srcip,dstip</tuple>
    </tuples>
    <report count="12" base="agghour"/>
    <report count="72" base="agg6hour"/>
</scoreboard>
```

This example reports on more tuples, like several combinations of src/dst ip and ports, but also on exporter ip, input and output interface. Note also that every flowfile is scoreboarded and a link is created to the latest scoreboard:

```
<scoreboard every=”1” latest="latest.html">
    <tuples>
         <tuple>srcip,dstip</tuple>
         <tuple>srcip,srcport,dstip,dstport</tuple>
         <tuple>srcport,dstport</tuple>
         <tuple>srcip,dstip,dstport</tuple>
         <tuple>srcip,srcport,dstip</tuple>
         <tuple>exporterip,input_if,output_if</tuple>
    </tuples>
    <report count="12" base="agghour"/>
    <report count="72" base="agg6hour"/>
</scoreboard>
```

Scoreboarding with tuples is possible on scoreboardother too:

```
<scoreboardother every=”1” latest="latest.html">
    <tuples>
         <tuple>srcip,dstip</tuple>
         <tuple>srcip,srcport,dstip,dstport</tuple>
         <tuple>srcport,dstport</tuple>
    </tuples>
    <report count="12" base="agghour"/>
    <report count="72" base="agg6hour"/>
</scoreboardother>
```



Aggregated Top-20 statistics

"Other" Direction

A direction named "other" (no other attributes) will catch all non-matched network traffic on all directions. This makes drill down traffic for network profiling possible.

Future Improvements

Issues still unsolved in JKFlow-3.5:

1: -The first aggregate scoreboard should aggregate flow files until the modula of the timestamp of the flowfile increments, this happens now for the 2nd which is incorrect. This results in n,lesser,n,n,n,... count aggregated scoreboards.

On the next pages are several configuration example scenario's addressed.

3.3: JKFlow Scenarios

Important: JKFlow has been changed a lot. Different new features and improvements were implementated, and this resulted in several changes in the original configuration rules. In JKFlow version 1 & 2, the improvements could be gradual implemented without big changes in the overall layout. In JKFlow 3, I've restructured the whole configuration file, to allow transparant configuration of routers, interfaces and subnets. Other new other features where implemented as well, so I recommend to look at the configuration scenario's of JKFlow3 after reading the configuration rules of JKFlow 3.

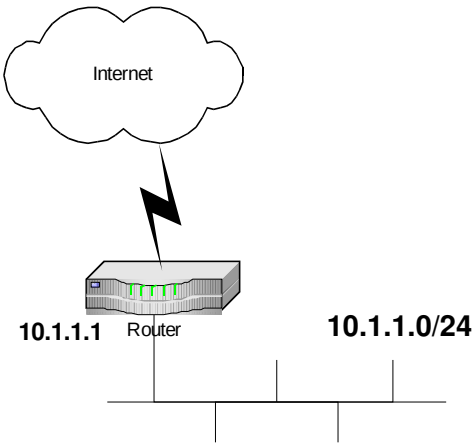
Here I provide the original scenario's for JKFlow 1,2.

3.3.1: JKFlow version 1,2 scenario's

Here are several scenarios adressed for users who wants to know the different configuration options. Configuring JKFlow is not hard, but you must have seen some examples to get an idea how you can write the most efficient configuration for your organisation.

Scenario 1: One single router for a single subnet

This is the easiest one: one single router is the access router to the Internet. This router connects the internal network to the Internet.



You have several options for configuring: 1 use the <all>, another use the <router>, and a last solution using <subnet>-element. The most efficient solution use the <all> element, because every flow passing the router is part of the network, so no filtering is needed:

```
<config>
       <all localsubnets="10.1.1.0/24">
              <application name="web">80/tcp,443/tcp</application>
              <services>25/tcp,53/udp,110/tcp,143/tcp</services>
              <tos/>
              <total/>
       </all>
       <outputdir>/var/flows/report/rrds</outputdir>
</config>
```

Another solution is this one:

```
<config>
       <routers>
              <router       name="router1"
                            routers="10.1.1.1"
                            localsubnets="10.1.1.0/24">
                        ...
              </router>
       </routers>
      <outputdir>/var/flows/report/rrds</outputdir>
</config>
```

The last solution is this:

```
<config>
       <subnets>
              <subnet       name="subnet1"
                            subnets="10.1.1.0/24">
                       ...
              </subnet>
       </subnets>
       <outputdir>/var/flows/report/rrds</outputdir>
</config>
```

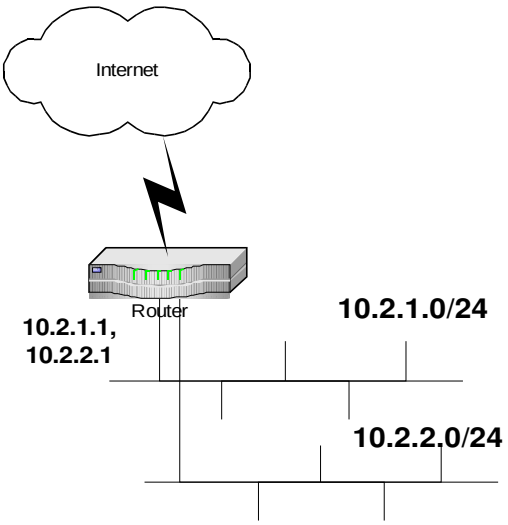
(A remark: The localsubnets attribute is not necessary needed, because JKFlow takes the subnets-attribute default as localsubnets, but you can define a different localsubnets attribute.)

The most efficient configuration is the configuration where you only have to monitor for services and don't have to monitor for specific router exporter or subnets. It is however possible, but the're less efficient because every evaluated flow would get an extra check on the exporter IP of subnet

In JKFlow is the localsubnet attribute optional for the subnets, and mandatory for the <all> and the <router> elements, because these elements don't have any information to determinate the local subnets to find which traffic is inbound or outbound.

Scenario 2: One router for a LAN network of 2 or more subnets

The network consists of two separate subnets connected to 1 router. Both subnets must be together/seperated reported.



In the case you need only 1 report, you can use an <all>, <router> or <subnet> element, where in the <subnet> element you use a subnets attribute with 2 subnets, like this:

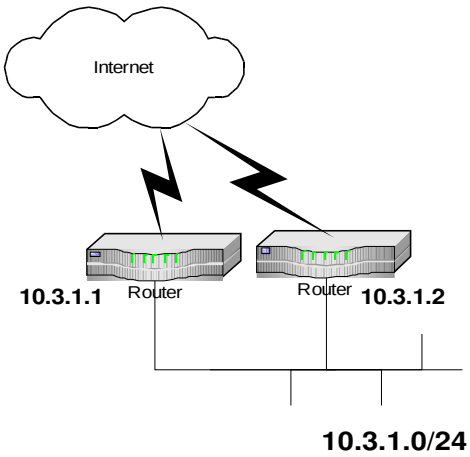
```
<config>
      <subnets>
            <subnet       name="subnet1"
                          subnets="10.2.1.0/24,10.2.2.0/24">
                        ...
            </subnet>
      </subnets>
      <outputdir>/var/flows/report/rrds</outputdir>
</config>
```

In the case you need 2 separated reports, one for each subnet, you must use 2 <subnet> elements:

```
<config>
       <subnets>
             <subnet       name="subnet1"
                           subnets="10.2.1.0/24">
                        ...
             </subnet>
             <subnet       name="subnet2"
                           subnets="10.2.2.0/24">
                        ...
             </subnet>
       </subnets>
       <outputdir>/var/flows/report/rrds</outputdir>
</config>
```

Scenario 3: LAN-network with 2 routers and 1 subnet

Here are 2 routers redundant configured to service a single subnet. In this case has the network traffic profiling not to split out over those routers, because we need only one profile for the subnet behind those routers.



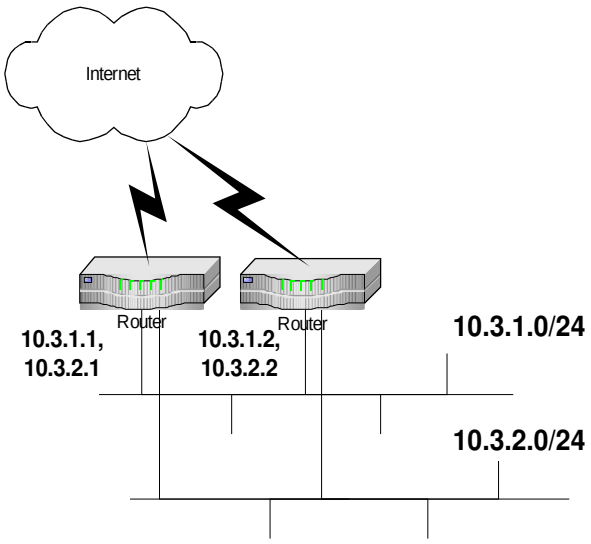
The configuration using <all> and <subnet> elements remains the same. In the <router> solution, despite that you collect flows from 2 different routers, you won't have to define a separate <router>-entry. You can define a <router> element with 2 or more router IP addresses:

```
<config>
       <routers>
             <router       name="router1"
                           routers="10.3.1.1,10.3.1.2"
```

```
                        localsubnets="10.3.1.0/24">
                    ...
            </router>
        </routers>
<outputdir>/var/flows/report/rrds</outputdir>
</config>
```

Also here remains the advice that can should use the <all> element, for efficiency. If you use <router> or <all> elements you must make use of the “localsubnet” attribuut, so the module can determinate inbound and outbound traffic.

Scenario 4: LAN-network with 2 routers and 2 subnets

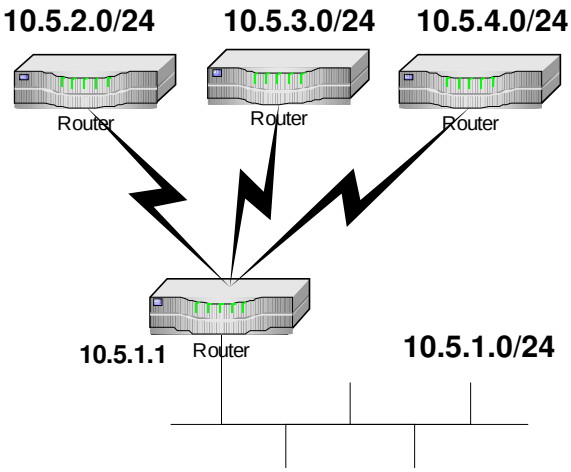


Here are 2 routers used for redundancy, but the difference with the previous example lies in that these 2 routers uniformly serves 2 subnets. In this scenario we use an <all> element like in the previous scenario.

In the case you would like to monitor the subnets with a diffent profile, you simply use 2 separate <subnet> elements and define the services to monitor in those elements, just like in scenario 2.

Scenario 5: WAN with 4 LAN-networks

A VPN infrastructure is configured between a single subnet network and 3 branch sites.
Here we want to monitor the Internet network traffic of the local site, and also the VPN traffic between the local site and the branch sites:



This is the first scenario where you have to define directions to monitor traffic passing to different sites. Directions are sections of source and destination subnets which has to match the flow IP-addresses. The flow is outbound if it goes along with the direction, and inbound if it goes contrary the direction.

You can make use of <subnet>, <network> or <all> elements in this example. The important point is that you must define 4 direction elements: 3 for the 3 branch offices, and 1 for the Internet traffic.

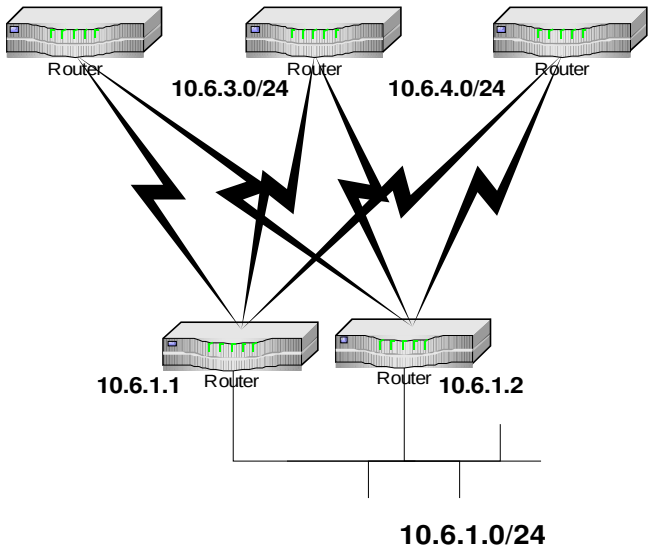
For each direction you must define the corresponding fromsubnets and tosubnets attributes. Note also that for monitoring the Internet traffic you must use a tosubnets="0.0.0.0/0" attribute and a notosubnets="10.5.2.0/24,10.5.3.0/24,10.5.4.0/24" attribute to negate the VPN traffic.

```
<config>
  <all localsubnets="10.5.1.0/24">
    <application name="web">80/tcp,443/tcp</application>
    <services>53/udp,110/tcp,143/tcp</services>
    <multicasts/>
    <tos/>
    <total/>
    <direction      name="site1"
                    fromsubnets="10.5.1.0/24"
                    tosubnets="10.5.2.0/24">
      <application name="windowe">
            137-139/tcp,137-139/udp
      </application>
      <services>25/tcp,110/tcp,143/tcp</services>
      <tos/>
      <total/>
    </direction>
    <direction      name="site2"
                    fromsubnets="10.5.1.0/24"
                    tosubnets="10.5.3.0/24">
      <services>102/tcp,110/tcp,143/tcp</services>
      <protocols>tcp,udp,icmp</protocols>
      <tos/>
      <total/>
    </direction>
    <direction      name="site3"
                    fromsubnets="10.5.1.0/24"
                    tosubnets="10.5.4.0/24">
      <application name="windowe">
            137-139/tcp,137-139/udp
      </application>
```

```
      <services>102/tcp,110/tcp,143/tcp</services>
      <tos/>
      <total/>
    </direction>
    <direction      name="Intranet"
                    fromsubnets="10.5.1.0/24"
                    tosubnets="0.0.0.0/0"
                    notosubnets="10.5.2.0/24,10.5.3.0/24,10.5.4.0/24">
      <application name="windowe">
            137-139/tcp,137-139/udp
      </application>
      <services>102/tcp,110/tcp,143/tcp</services>
      <tos/>
      <total/>
    </direction>
  </all>
  <outputdir>/var/flows/report/rrds</outputdir>
</config>
```

Scenario 6: WAN with 4 LAN networks over 2 redundant routers

In this scenario there is another router for redundancy added to the local subnet. The configuration remains however the same.

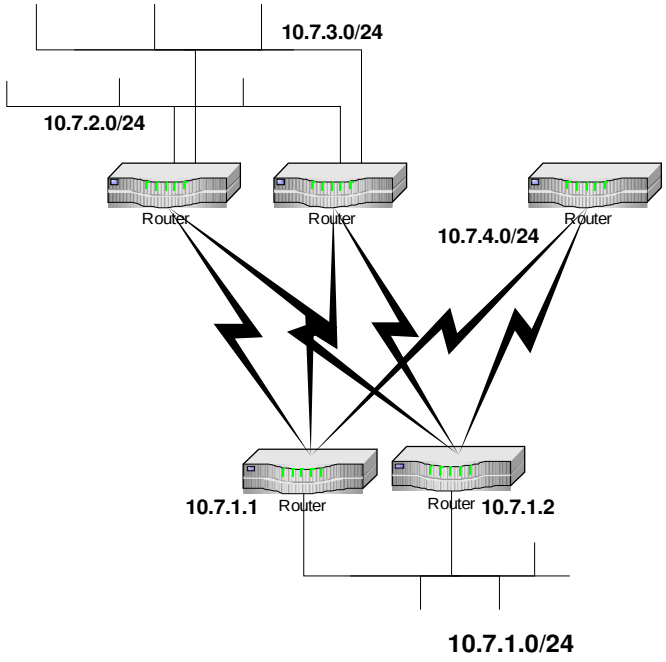



```
<config>
  <all localsubnets="10.6.1.0/24">
    <application name="web">80/tcp,443/tcp</application>
    <services>53/udp,110/tcp,143/tcp</services>
    <multicasts/>
    <tos/>
    <total/>
    <direction      name="VPN-site1"
                    fromsubnets="10.6.1.0/24"
                    tosubnets="10.6.2.0/24">
      <services>25/tcp,110/tcp,143/tcp</services>
      <total/>
    </direction>
    <direction      name="VPN-site2"
                    fromsubnets="10.6.1.0/24"
                    tosubnets="10.6.3.0/24">
      <services>102/tcp,110/tcp,143/tcp</services>
      <total/>
    </direction>
    <direction      name="VPN-site3"
```

```
                    fromsubnets="10.6.1.0/24"
                    tosubnets="10.6.4.0/24">
      <services>102/tcp,110/tcp,143/tcp</services>
      <total/>
    </direction>
    <direction      name="Internet"
                    fromsubnets="10.6.1.0/24"
                    tosubnets="0.0.0.0/0"
                    notosubnets="10.6.2.0/24,10.6.3.0/24,10.6.4.0/24">
      <services>102/tcp,110/tcp,143/tcp</services>
      <protocols>tcp,udp,icmp</protocols>
      <total/>
    </direction>
  </all>
  <outputdir>/var/flows/report/rrds</outputdir>
</config>
```

Scenario 7: Single subnet connected to the Internet and VPN-connections to 2 remote sites, with one site comprising two subnets

In this case we have 2 remote sites. In one remote there are 2 subnets, and different monitoring is needed for those 2 subnets.



We can choose in this situation:

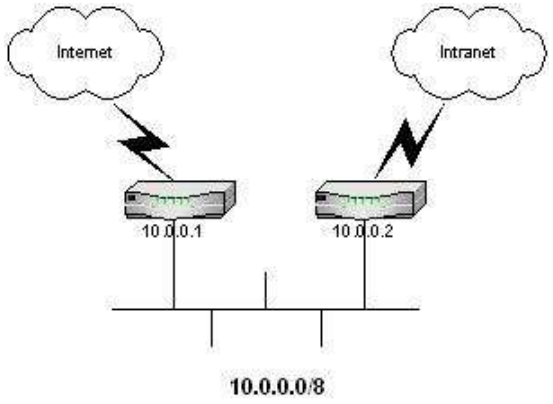
We can define 3 directions: 1 for the 10.7.2.0/24 subnet, another for the 10.7.3.0/24 subnet and a third for the 10.7.4.0/24 subnet on the second site.

You can also choose another method: define 2 directions, 1 for subnet 10.7.2.0/23 (which comprises 10.7.2.0/24 and 10.7.3.0/24) and a second for the 10.7.4.0/24 subnet. Inside the direction to subnet 10.7.2.0/23, we define 2 further directions to the subnets 10.7.2.0/24 and 10.7.3.0/24. This is an example:

```
<config>
       <all localsubnets="10.7.1.0/24">
              <services>53/udp,110/tcp,143/tcp</services>
              <total/>
              <direction    name="Site-1"
                            fromsubnets="10.7.1.0/24"
                            tosubnets="10.7.2.0/24,10.7.3.0/24">
                     <direction    name="Site-1 subnet1"
                                   fromsubnets="10.7.1.0/24"
                                   tosubnets="10.7.2.0/24">
                            <services>102/tcp,110/tcp,143/tcp</services>
                     </direction>
                     <direction    name="Site-2 subnet2"
                                   fromsubnets="10.7.1.0/24"
                                   tosubnets="10.7.3.0/24">
                            <services>102/tcp,110/tcp,143/tcp</services>
                     </direction>
              </direction>
              <direction    name="Site2"
                            fromsubnets="10.7.1.0/24"
                            tosubnets="10.7.4.0/24">
                     <services>102/tcp,110/tcp,143/tcp</services>
                     <protocols>tcp,udp,icmp</protocol>
                     <total/>
              </direction>
  </all>
  <outputdir>/var/flows/report/rrds</outputdir>
</config>
```

Scenario 8: Monitoring transit traffic

To monitor transit traffic (traffic not involving any hosts on the local subnet except for routing funktions) you can use a direction with a notosubnets attribute specified to the local subnet. Use intranet subnets for the fromsubnets attribute and a 0.0.0.0/0 subnet for the tosubnets attribute, and add the intranet subnet to the notosubnets attribute. This will enable the direction to monitor the direction of the traffic that is passing the subnet.



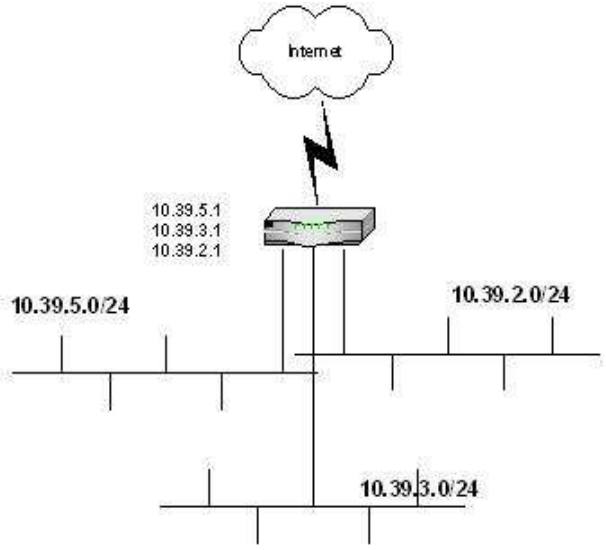
This example assumes the intranet uses subnet 192.168.0.0/16

```
<config>
       <all   name="subnet1">
              <direction    name="transit-to-Internet"
                            fromsubnets="192.168.0.0/16"
                            tosubnets="0.0.0.0/0"
                            notosubnets="10.0.0.0/8,192.168.0.0/16">
                     <services>25/tcp,80/tcp,110/tcp,143/tcp</services>
                     <protocols>tcp,udp,icmp</protocol>
```

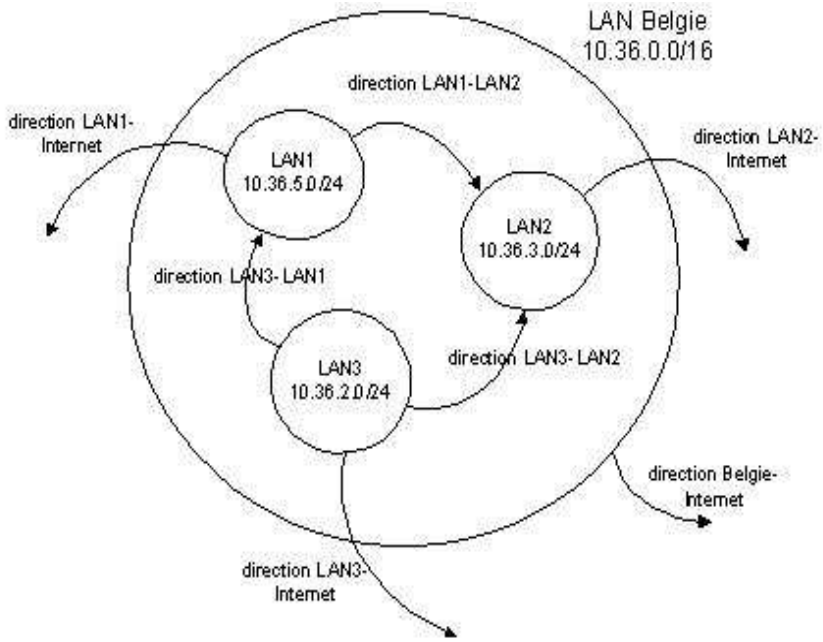
```
                <total/>
            </direction>
            <direction   name="local-to-Internet"
                         fromsubnets="10.0.0.0/8"
                         tosubnets="0.0.0.0/0"
                         notosubnets="10.0.0.0/8,192.168.0.0/16">
                <services>25/tcp,80/tcp,110/tcp,143/tcp</services>
                <protocols>tcp,udp,icmp</protocol>
                <total/>
            </direction>
        </all>
        <outputdir>/var/flows/report/rrds</outputdir>
</config>
```

Scenario 9: Monitoring traffic between traffic and Internet traffic of 3 subnets, and the Internet traffic as whole.

In this case we want to monitor the traffic between 3 separated subnets, and the Internet traffic of each subnet and the overall Internet traffic of the 3 subnets:



How can we define the directions? The following schema shows a possible layout of the directions that you can define. Remember that you must try to define as much as nested directions as possible, because it is efficient.



On the next page is a possible solution:

```
<config>
    <all localsubnets="10.36.0.0/16">
        <direction   name="Belgium-Internet"
                     fromsubnets="10.36.0.0/16"
                     tosubnets="0.0.0.0/0"
                     notosubnets="10.36.0.0/16">
            <direction   name="LAN1-Internet"
                         fromsubnets="10.36.5.0/24"
                         tosubnets="0.0.0.0/0"
                         notosubnets="10.36.5.0/24">
                ...
            </direction>
            <direction   name="LAN2-Internet"
                         fromsubnets="10.36.3.0/24"
                         tosubnets="0.0.0.0/0"
                         notosubnets="10.36.3.0/24">
                ...
            </direction>
            <direction   name="LAN3-Internet"
                         fromsubnets="10.36.2.0/24"
                         tosubnets="0.0.0.0/0"
                         notosubnets="10.36.2.0/24">
                ...
            </direction>
        ...
        </direction>
        <direction   name="LAN3-otherLANs"
                     fromsubnets="10.36.2.0/24"
                     tosubnets="10.36.5.0/24,10.36.3.0/24">
            <direction   name="LAN3-LAN1"
                         fromsubnets="10.36.2.0/24"
                         tosubnets="10.36.5.0/24">
                ...
            </direction>
            <direction   name="LAN3-LAN2"
                         fromsubnets="10.36.2.0/24"
                         tosubnets="10.36.3.0/24">
                ...
            </direction>
            ...
        </direction>
        <direction   name="LAN1-LAN2"
```

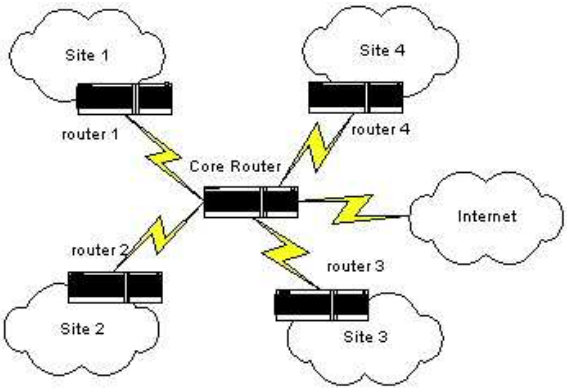
```
                        fromsubnets="10.36.5.0/24"
                        tosubnets="10.36.3.0/24">
                    ...
                </direction>
                ...
        </all>
        <outputdir>/var/flows/report/rrds</outputdir>
</config>
```

3.3.2: JKFlow version 3 scenario's

Here I describe some situations to look at to you an impression where to pay attention to, if you want to deploy JKFlow 3. These situations will not cover every possible configuration, because with the improved configuration possibilities, there much more possible, consider.

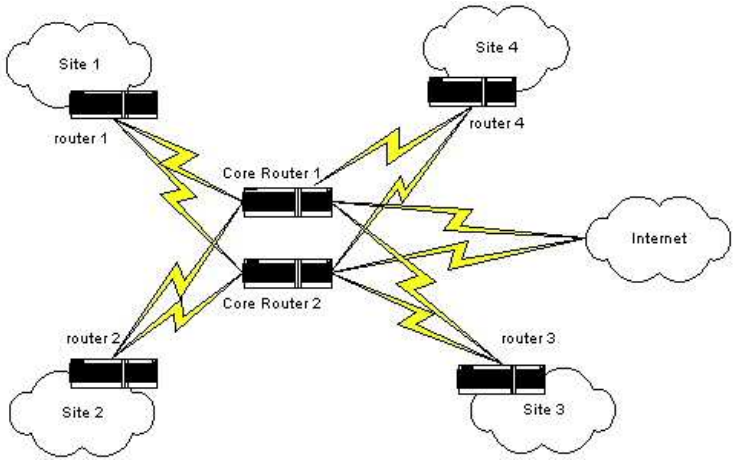
Scenario 1: Hub & Spoke with a single Core router

Here a central router is acting as a core router connecting a number of hub routers of branch sites. You want to monitor the traffic passing from and between each of the branch sites and the Internet. As an extra you also want to monitor the internal subnetwork traffic of each branch site.



If you only want to monitor the traffic from and between the branch sites and the Internet, you are easely off: Just define the sites and define the directions with from/to attributes between those sites. OR: define a routergroup for each interface on the core router and use for every direction the routergroup. You only enable Netflow on the core router, on all interfaces for both. If you want to monitor internal subnetwork traffic, you must enable netflow monitoring on routers of the desired branch sites too, which introduces another problem: how to avoid double counting flows? When you enable netflow monitoring on the branch sites and possible the core router, every traffic between the branch sites and with the Internet will generate flows from each router. In this case you must configure routergroups, and use the routergroups in the directions for every branch site.

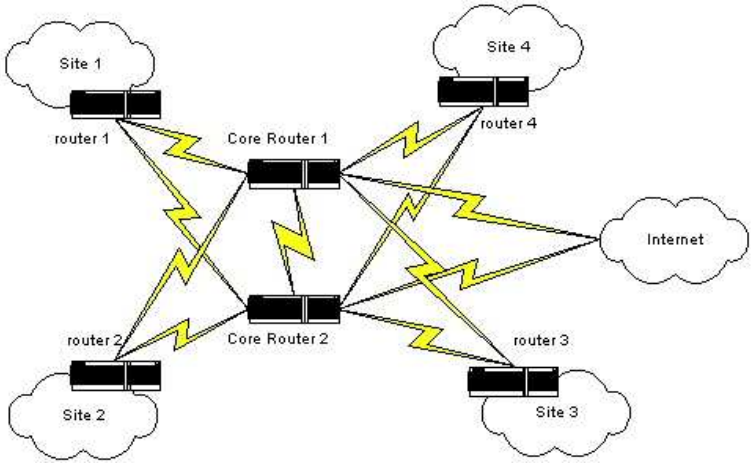
Scenario 2: Hub & Spoke with redundant non-interconnected core routers



This network has redundancy in network design (which is good), and traffic received on a core router will no pass the other core router (which is also good). Here you don't have to be afraid of double counting flows, and continue using directions with from/to attributes and subnets/sites.

Scenario 3: Hub & Spoke with redundant interconnected core routers

This network has redundancy in network design (which is good), but has redundancy in Netflow monitoring on the core router 2 due to the interconnection of these core routers (which is bad).



Here redundancy in network design has the possible outcome that network traffic can pass over multiple Netflow enabled routers. To avoid double-counting flows you have to configure routergroups including router interfaces of the branch routers and use these routergroups in the directions between the branch sites and the Internet.

(Warning: Don't think you can avoid the redundancy issue by disabling Netflow on the intermediary link between the core routers.)

Questions to ask yourself when configuring JKFlow:

- Do I monitor all the traffic with Netflow once?
- Do traffic traverse over multiple Netflow enabled routers?

Some basic strategic JKFlow / Netflow setup guidelines:

1: Determinate what you want to monitor: Is it the traffic of a subnet, a network, or a link? Ask yourself if it can be measured by monitoring from/to sites or subnets, or with grouping of router interfaces. Determinate your network monitoring model: Is it basically source/destination subnets, localsubnets, or ingress/egress network monitoring?

2: Follow the same rules where to place firewalls (like as short by the source as possible on source filtering) to determinate placement of Netflow agents.

3: Try to define source/destination subnets based directions if the subnet design match the physical infrastructure. (I mean subnets when using from/to/nofrom/noto/fromsubnets/ tosubnets/nofromsubnets/notosubnets attributes)

': In case of redundant links, if you have network paths through multiple monitored routers (like 2 firewalls), try to eliminate these links because redundancy should be provided between the core/distribution/access layers and not between individual distribution routers to reduce the amount of routes learned by dynamic routing.

4: If this is not possible, define routergroups consisting of router interfaces, to define demarcation points where all traffic from/to a network must pass only once. Define a direction with the routergroup and possible from/to attributes.

6: Use a central router if you want to centralize netflow monitoring on a central router. Enabling netflow on a central router, will increase the load of the central router, but the udp packets don't have to travel across links, and your monitoring infrastructure become more reliable.

Routergroups enabled a new kind of network monitoring which was previous not possible. You can now configure monitoring redundant networks with multiple routers connected with multiple sites over redundant links with each router. Now you can configure a routergroup grouping on a single router all links with every sites to monitor the traffic on a router, and configure a routergroup grouping all links to the same site to monitor the traffic to a site.

3.4: Flaws to be avoided with NetFlow/JKFlow

When implementing a NetFlow/JKFlow monitoring infrastructure you have to avoid some mistakes. Most problems I discuss here were reported to me, and are real case scenarios. The problems discussed here doesn't result from JKFlow errors, but often results of some basically misconceptions of NetFlow monitoring.

If I have to explain the internal working of NetFlow+JKFlow in a very few words, I would say:

1:NetFlow aggregates data of traffic-sessions, and sends data like this: source/destination IP, port, protocol, interfaces, AS, packets, bytes
2: JKFlow receives this data and increments counters on each matching defined directions/services/protocol elements.
3: After this any detailed data of the flow are gone.

Misinterpreted inbound/outbound traffic

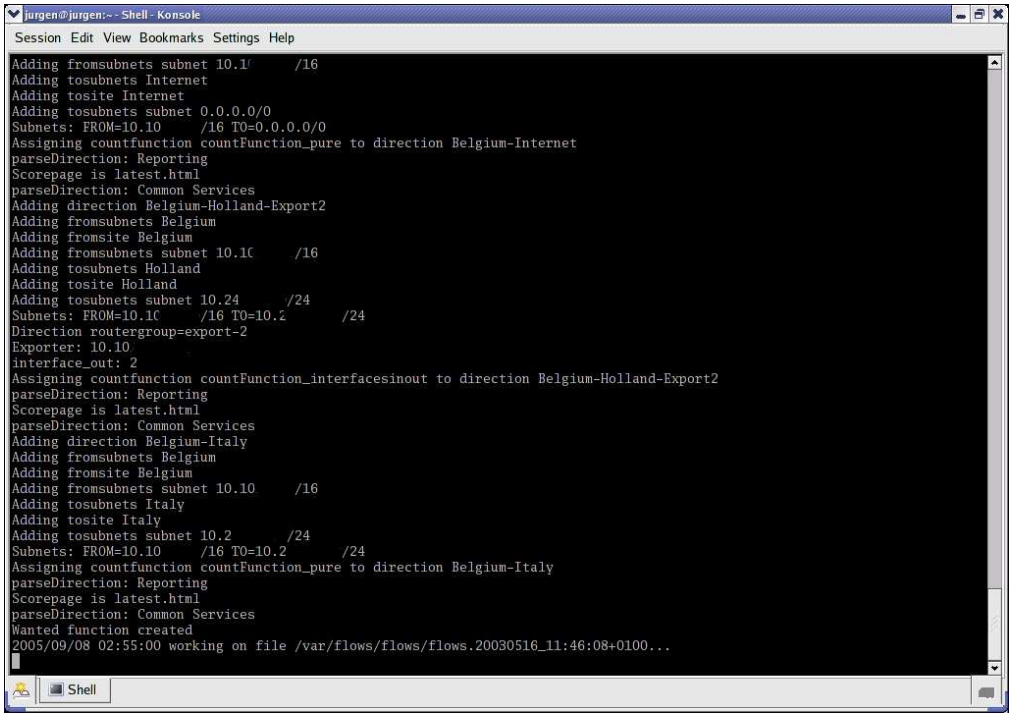
Cisco's NetFlow does not have the ability to mindread what you see as in-, or outbound. Rather than reporting in-/outbound directions, Netflow reports the inbound and the outbound interface of a flow. JKFlow designate the inbound and outbound direction using the source/destination IP addresses of the flow matching with the source/destination subnets of the directions, and the localsubnets of the “all”,”router” elements, and routergroups in JKFlow. In the case of routergroups with routerinterfaces JKFlow sees the direction of flows in the perspective of the router. A special caution: interpreting inbound and outbound in JKFlow is done by designating flows to be inbound in “all”/”router” elements if the destination address is located in the localsubnets. Thus will inside traffic directed inside passing a router be seen as inbound. Only in JKFlow3 version 3.01 and further this is corrected to check both source and destination IP-addresses.

NetFlow monitoring on single interfaces and double counting flows over routers.

NetFlow reports traffic only one. Traffic passing a router doesn't get reported both on inbound and outbound interfaces, but only on inbound interfaces, but with reporting of the outbound interface included. This means that if you only enable monitoring NetFlow on a single interface you want to report, you will probably miss flows orginating from other interfaces.
Also this will not prevent reporting traffic that is passing multiple routers to be reported multiple times. NetFlow and JKFlow doesn't provide facilities to correlate these flows.

3.5: Execution of FlowScan with JKFlow

Important: You must install the Perl-module **XML::Simple** to be able to run the report module , because this module is responsible for the parsing of the configuration file. Also you have to modify de setting ReportClasses to **JKFlow** in the **flowscan.cf** file. After you did this you can start the FlowScan script. I leaved some debugging messages, so you can easily verify to see if everything is running smoothly:



JKFlow running

3.6 Debugging Configuration JKFlow.xml

To debug the configured JKFlow.xml you should take a look at the output of the parsing while starting flowscan. Here are a few dumps (of JKFlow version 3.2.1) explained:

Here a direction is created with from/to attributes to sites. The subnets of the sites are added and the resulting subnets are printed. After the subnets the set “Common Services” is parsed:

```
...
Adding direction Engeland-Internet
Adding fromsubnets England
Adding fromsite England
Adding fromsubnets subnet 10.30.0.0/16
Adding tosubnets Internet
Adding tosite Internet
Adding tosubnets subnet 0.0.0.0/0
Adding notosubnets England
Adding tosite England
Adding notosubnets subnet 10.30.0.0/16
Subnets: FROM=10.30.0.0/16 TO=0.0.0.0/0
parseDirection: Common Services
...
```

Several routergroups are created and router interfaces are added:

```
...
Routergroup: ROUTERS1-LINE2
Exporter: 10.1.1.1, interface: 11
Exporter: 10.1.1.2, interface: 11
Routergroup: ROUTERS1-LINE1
Exporter: 10.1.1.1, interface: 10
Exporter: 10.1.1.2, interface: 10
Routergroup: ROUTERS1
Exporter: 10.1.1.1, interface: 10
Exporter: 10.1.1.1, interface: 11
Exporter: 10.1.1.2, interface: 10
```

```
Exporter: 10.1.1.2, interface: 11
...
```

Here is a direction ROUTERS1 created, linked to a routergroup, resulting in the adding of the exporters and at last the applications/services/protocols/etc... in the set “Common Services” is parsed:

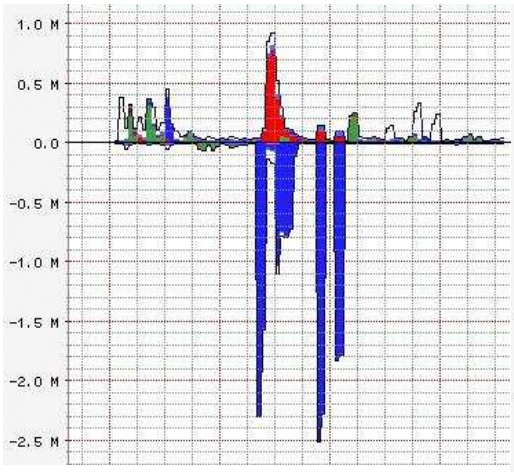
```
...
Adding direction ROUTERS1
Direction routergroup=ROUTERS1
Exporter: 10.1.1.1, interface: 10
Exporter: 10.1.1.1, interface: 11
Exporter: 10.1.1.2, interface: 10
Exporter: 10.1.1.2, interface: 11
Assign routergroup ROUTERS1 to Direction ROUTERS1
parseDirection: Common Services
...
```

Here is a direction ROUTERS1-LINE2 created, linked to a routergroup ROUTERS1-LINE2, resulting in the adding of the exporters and at last the applications/services/protocols/etc... in the set “Common Services” are parsed:

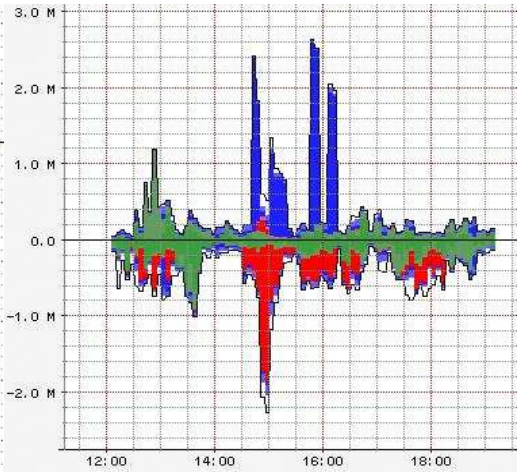
```
...
Adding direction ROUTERS1-LINE2
Direction routergroup=ROUTERS1-LINE2
Exporter: 10.1.1.1, interface: 11
Exporter: 10.1.1.2, interface: 11
Assign routergroup ROUTERS1-LINE2 to Direction ROUTERS1-LINE2
parseDirection: Common Services
...
```

3.7 Interpretation of Graphs

Here are a few example graphs created with JKFlow/JKGrapher of a direction passing a router. The 3 graphs shows the same direction, but on different interfaces on the router. 4 big blue spikes are cleary visible: windows filetransfers. The windows filetransfers enters the router on interface 1 and leaves the router on interface 2.

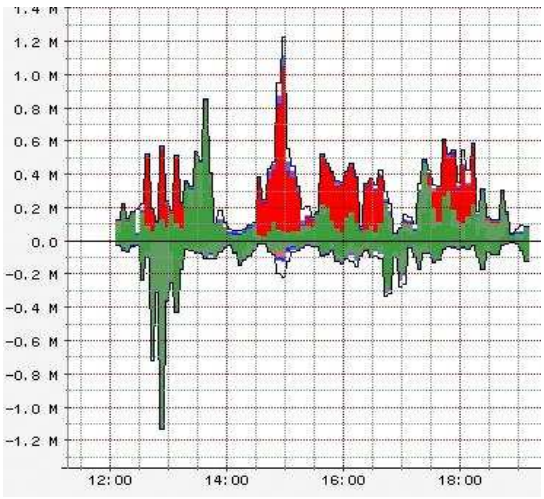


Router Interface 1



Router Interface 2

Further you can see that most other traffic of this direction passes the router on interfaces 2 and 3. This shows in a inbound/outbound mirrored traffic profile (web and mail traffic). Also, if you look closely you can see that 2 Mb webtraffic entering the router via interface 2 around 15h00 splits into 0.8 Mb leaving interface 1 and 1.2 Mb leaving interface 3.



Router Interface 3

3.8: Principe JKFlow

I've written this report module for FlowScan in Perl, starting from the already existing CUFlow module.

The rewrite of the module may not be underestimated, because of 2 reasons:

The first reason is that FlowScan is actually not a reasonable large program. The only functionalities that I'm confronted with were the reading of the flowfiles every 30 seconds, then the calling of the reporting routines, like the initialisation of the data structures, which is done only once at startup of FlowScan, and the counting of every flow record, for each flowrecord in the flow files. At end the FlowScan module will call another routine in the report module, writing all counters to the RRDTool databases. After processing the flow file it will be deleted, and the next flowfile will be processed. It seems that from my point of view that almost 90% of the functionality is located in the report module.

The second reason is that it is of great importance that the reportmodule is written as efficient (read: FAST!) as possible. This means practicaly that the “wanted” routine must process every flow record as fast as possible. You may not forget that FlowScan is widely used in WAN networks where lots of flows are generated. A simple routine for the processing of the flow records (like SubnetIO) will garantee a quick processing of the flow files, but it won't leave much room for configuration.

JKFlow is a rewritten module starting from CUFlow, where statistics are accumulated in one giant hash. The hash and the code is written optimal structured so redundacy in the code and the hash is eliminated as much as possible. Therefor relative flexible/big changes where possible at the end in the project.

The choise to base the program on hash-lookups is not ungrounded. Perl deliver the programmer the possibility to place data in a “associative array”. Here can the elements of a array be places with an hash-routine on the key on a specific place in the table. It is possible to find the value directly by using the hash-routine on the key. With this the access time of a hash value is unrelated or O(1) to the size of the hash-array.

(Within a hash table is sometimes a so-called “hash” collision possible. That is when the hash routines makes that 2 distinct values are placed behind the same place in the hash-array, because the hash routine generates to the same position in this array. This event is handled internally within the hash algorithm, and is invisible to the end user.)

Within the report module are “references” widely used. A “reference” can be compared with a “pointer” in the C language, and references a variable, than representing the variable by itself. Using references for arguments passing in functions makes programming flexible and is called “call by reference”, in contrast to directly passing of variables with functions, “call by value”.

Another possibility in perl is placing references directing to hashes as values in another hash. This can be done on different layers, creating a hash-tree. It is possible to using references located inside a hash-tree to pass as argument to functions, thereby passing a part of the hash-tree this way. Such techniques are common in Perl, and is used in several modules available on CPAN. As example, XML::Simple dumps the DOM model of a XML-file to a hash-tree.

The configuration of JKFlow is done using a XML file. This file is read in at startup in a hash-tree, and from the content another hash-tree is initialised with the counters. Based on existing hash-values in this tree the module processes each flow record on different criteria, and increments the counters.

Within the report module is this hash-structure widely reused:
(Every colums to the right points to a further brach, and the gray areas are fixed values in the hash-tree.)

Hash 1	Hash 2	Hash 3	Hash 4	Hash 5	Hash 6
Protocol	$protocol	**Total**	$which	**Flows** **Bytes** **Pkts**	
Multicast	**Total**	$which	**Flows** **Bytes** **Pkts**		
Service	$protocol	$service	**Src** **Dst**	$which	**Flows** **Bytes** **Pkts**
Total	$which	**Flows** **Bytes** **Pkts**			
Tos	$tos	$which	**Flows** **Bytes** **Pkts**		

In the functions “**countpackets”** and “**countmulticasts”** you can see how the counters in this structure are increased. For the ease I show you a piece of code of “countpackets”, responsible for the protocol and service counters (The 1st 2nd and the 5th hash within the showed structure.)

```
if ((defined $ref->{'protocol'}) && (defined $ref->{'protocol'}{$protocol})) {
        $ref->{'protocol'}{$protocol}{'total'}{$which}{'flows'}++;
        $ref->{'protocol'}{$protocol}{'total'}{$which}{'bytes'} += $bytes;
        $ref->{'protocol'}{$protocol}{'total'}{$which}{'pkts'} += $pkts;
        $ref->{'protocol'}{$protocol}{'tos'}{$typeos}{$which}{'flows'}++;
        $ref->{'protocol'}{$protocol}{'tos'}{$typeos}{$which}{'bytes'} += $bytes;
        $ref->{'protocol'}{$protocol}{'tos'}{$typeos}{$which}{'pkts'} += $pkts;
```

```
}
if ((defined $ref->{'service'}) && (defined $ref->{'service'}{$protocol})) {
        if (defined $ref->{'service'}{$protocol}{$dstport}) {
                $ref->{'service'}{$protocol}{$dstport}{'dst'}{$which}{'flows'}++;
                $ref->{'service'}{$protocol}{$dstport}{'dst'}{$which}{'bytes'} += $bytes;
                $ref->{'service'}{$protocol}{$dstport}{'dst'}{$which}{'pkts'} += $pkts;
        }
        elsif (defined $ref->{'service'}{$protocol}{$srcport}) {
                $ref->{'service'}{$protocol}{$srcport}{'src'}{$which}{'flows'}++;
                $ref->{'service'}{$protocol}{$srcport}{'src'}{$which}{'bytes'} += $bytes;
                $ref->{'service'}{$protocol}{$srcport}{'src'}{$which}{'pkts'} += $pkts;
        }
}
```

You can see that there is a check if a the hashvalue of a protocol of service exist, before it increases its protocol, and service counters. These hashvalues are filled in during parsing of the XML configuration file.

The $ref is the reference pointing to some subtree into the hash-tree, and all functions uses this reference exclusivly. Therefor it is possible to reuse the function for every subnet, router and direction that has to be monitored, and this is the also reason of the reoccuring hash-structure in the hash-tree. This is how the module works on the lowest level.

Now this is how the modules handles the hash-tree globally:

Down below you can see how the functions are called in the function **“wanted”** (used to process every flow record) Here are the functions **countpackets** & **countmulticasts** multiple time called for different routers routers, subnets, networks, etc... You can notice that every function pass a reference to a specific hash-structure inside the %JKFlow::mylist hash-tree:

```
# Counting for specific Routers
foreach my $routername (keys %{$JKFlow::mylist{'router'}}) {
    if (defined $JKFlow::mylist{'router'}{$routername}{'localsubnets'}) {
        if ($JKFlow::mylist{'router'}{$routername}{'localsubnets'}->match_integer($dstaddr)) {
            $which = 'in';
        } else {
            $which = 'out';
        }
    }
    if (defined $JKFlow::mylist{'router'}{$routername}{'routers'}{$exporterip}) {
        countpackets(\%{$JKFlow::mylist{'router'}{$routername}},$which);
        countApplications(\%{$JKFlow::mylist{'router'}{$routername}{'application'}},$which);
        countDirections(\%{$JKFlow::mylist{'router'}{$routername}{'direction'}},$which);
        #only $dstaddr can be a multicast address
        if (($dstaddr & $JKFlow::MCAST_MASK) == $JKFlow::MCAST_NET) {
            countmulticasts(\%{$JKFlow::mylist{'router'}{$routername}},$which);
        }
    }
}
```

... (This happens in the same way in the all-tree and the subnet-trees)

You can quicky in this way use this code to process such hash-trees:

Router	$IProuter	**Protocol**	$protocol	**Total**	$which	**Flows Bytes Pkts**	
Router	$IProuter	**Multicast**	**Total**	$which	**Flows Bytes Pkts**		
Router	$IProuter	**Service**	$protocol	$service	**Src Dst**	$which	**Flows Bytes Pkts**
Router	$IProuter	**Total**	$which	**Flows Bytes Pkts**			
Router	$IProuter	**Tos**	$tos	$which	**Flows Bytes Pkts**		

Subnet	$subnet	**Protocol**	$protocol	**Total**	$which	**Flows Bytes Pkts**	
Subnet	$subnet	**Multicast**	**Total**	$which	**Flows Bytes Pkts**		
Subnet	$subnet	**Service**	$protocol	$service	**Src Dst**	$which	**Flows Bytes Pkts**
Subnet	$subnet	**Total**	$which	**Flows Bytes Pkts**			
Subnet	$subnet	**Tos**	$tos	$which	**Flows Bytes Pkts**		

Within the different all, routers, subnets and networks are beside “countpackets” and “countmulticasts” also the functions “**countApplications”** and “**countDirections”** called.

```
# Counting ALL
if (defined $JKFlow::mylist{'all'}) {
    if (defined $JKFlow::mylist{'all'}{'localsubnets'}) {
        if ($JKFlow::mylist{'all'}{'localsubnets'}->match_integer($dstaddr)) {
            $which = 'in';
        } else {
            $which = 'out';
        }
    }
    countpackets(\%{$JKFlow::mylist{'all'}},$which);
    countApplications(\%{$JKFlow::mylist{'all'}{'application'}},$which);
    countDirections(\%{$JKFlow::mylist{'all'}{'direction'}},$which);
    if (($dstaddr & $JKFlow::MCAST_MASK) == $JKFlow::MCAST_NET) {
```

```
            countmulticasts(\%{$JKFlow::mylist{'all'}},$which);
        }
    }
```

countDirections is something special: It is a recursive function, and it calls the functions **countpackets, countApplications, countmulticasts** for each direction, so you can define services, protocols, TOS and total counters in each direction. Directions are grouping of source subnets and destination subnets with a specific name like “Belgium-France”. Within every direction you can define several other directions, each containing others directions and in every direction are the functions **countpackets, countApplications, countmulticasts** called again so you can recursively define directions, with each defining a different set of protocols, services, TOS and total to monitor.

```
sub countDirections {
my $ref=shift;
my $which=shift;

        foreach my $direction (keys %{$ref}) {
        …
                countDirections(\%{$ref->{$direction}{'direction'}},$which);
        }
}
```

Because of this you can subdivide the network subnets to monitor different sets of protocols, services, TOS and total. Here are the hash-trees for the different sets for deeper levels of directions:

Subnet	$subnet	**Direction**	$direction	**Protocol**	$protocol	**Total**	$which	**Flows**	
Subnet	$subnet	**Direction**	$direction	**Multicast**	**Total**	$which	**Flows**		
Subnet	$subnet	**Direction**	$direction	**Service**	$protocol	$service	**Src**	$which	**Flows**
Subnet	$subnet	**Direction**	$direction	**Total**	$which	**Flows**			
Subnet	$subnet	**Direction**	$direction	**Tos**	$tos	$which	**Flows**		

Subnet	$subnet	**Direction**	$direction	**Direction**	$direction	**Protocol**	$protocol	**Total**	$which	**Flows**
Subnet	$subnet	**Direction**	$direction	**Direction**	$direction	**Multicast**	**Total**	$which	**Flows**	
Subnet	$subnet	**Direction**	$direction	**Direction**	$direction	**Service**	$protocol	$service	**Src**	$which **Flows**
Subnet	$subnet	**Direction**	$direction	**Direction**	$direction	**Total**	$which	**Flows**		
Subnet	$subnet	**Direction**	$direction	**Direction**	$direction	**Tos**	$tos	$which	**Flows**	

With **countApplications** you can count several services as being one single service/application. If you define “Windows” as the services 137-139/udp, 137-139/tcp, 135/tcp and 445/tcp, you can counts these services as a single service. The result will be that you will only have a single service in your list of services on your webform, and not 7 separe windows related services.

Here are the hash-trees for applications for deeper levels of directions:

Subnet	$subnet	**Direction**	$direction	**FromSubn**	$fromsubn			
Subnet	$subnet	**Direction**	$direction	**ToSubnet**	$tosubnet			
Subnet	$subnet	**Direction**	$direction	**Applicati**	$applicat	**Src**	$which	**Flows**
Subnet	$subnet	**Direction**	$direction	**Applicati**	$applicat	**Service**	$protocol	$service

Subnet	$subnet	**Direction**	$direction	**Direction**	$direction	**FromSub**	$fromsubnet			
Subnet	$subnet	**Direction**	$direction	**Direction**	$direction	**ToSubnet**	$tosubnet			
Subnet	$subnet	**Direction**	$direction	**Direction**	$direction	**Applicat**	$application	**Src**	$which	**Flows**
Subnet	$subnet	**Direction**	$direction	**Direction**	$direction	**Applicat**	$application	**Service**	$protocol	$service

It is inevitable that programming such kind of code must be done following a tight scheme, because bug-free calling of the functions with the correct references is of great importance. A dump of the whole hash-tree can easely takes tens of pages, and an error in the code can causing a fault in the hash structure only after processing tens or hunders of flows.

3.9: Performance issues in JKFlow/JKFlow2

In the "wanted"-method of the module are several funktions with hash-lookups called. Will the processing of all the flow records, depending on the configuration of JKFlow, not get very slow?
There are a lot of hash-lookups possible, so I was also concerned if the speed of the hash-lookup would make the module painfully slow. Fortunaly I could do a profiling test in Perl of the module, allowing to make an estimate of the commulative time that Perl spends on specific funktions.

The command "perl –d:Dprof flowscan" starts FlowScan execution, but will also starts dumping a profiling dump file tmon.out with the necessary data for analyse afterwards. This file can be analysed afterwards with dprofpp.

Before I could start a profiling test, I had to update the Devel::DProf module also, because the original module crashed.

These are the results:

```
[root@homepc flows]# dprofpp tmon.out
Garbled profile is missing some exit time stamps:
Cflow::find
main::wanted
JKFlow::wanted
Try rerunning dprofpp with -F.
[root@homepc flows]# dprofpp -F tmon.out
Faking 3 exit timestamp(s).
Total Elapsed Time = -1.55103 Seconds
  User+System Time =        0 Seconds
Exclusive Times
%Time ExclSec CumulS #Calls sec/call Csec/c Name
 0.00   4.022 11.647  12438  0.0003 0.0009 JKFlow::wanted
 0.00   2.640  2.591  49097  0.0001 0.0001 JKFlow::countApplications
 0.00   2.100  1.582 517474  0.0000 0.0000 Net::Patricia::match_intege
 0.00   2.080  2.031  49097  0.0000 0.0000 JKFlow::countpackets
 0.00   1.967  3.257  89424  0.0000 0.0000 JKFlow::countDirections
 0.00   0.330  0.330     38  0.0087 0.0087 RRDs::create
 0.00   0.290  0.265  24876  0.0000 0.0000 Cflow::InetNtoA::FETCH
 0.00   0.210 11.838  12438  0.0000 0.0010 main::wanted
 0.00   0.170  0.169    679  0.0003 0.0002 RRDs::update
 0.00   0.090  0.645    679  0.0001 0.0009 JKFlow::reporttorrd
 0.00   0.090  0.550     16  0.0056 0.0344 main::BEGIN
 0.00   0.088 11.935      2  0.0439 5.9675 Cflow::find
 0.00   0.080  0.129      6  0.0133 0.0216 XML::LibXML::SAX::BEGIN
 0.00   0.050  0.050     58  0.0009 0.0009 Exporter::import
 0.00   0.050  0.833     25  0.0020 0.0333 JKFlow::reporttorrdfiles
[root@homepc flows]#
```

Here you can see the time FlowScan used to process the flowfile. As espected JKFlow::wanted took the largest portion of the execution time. The methods countApplications, countpackets, countDirections took all a none-exclusive portion of the total processing time. (This is because these functions depends on eachother, so the sum is bigger than the total processing time) What is important is that the function Net::Patricia::match_integer still takes more than +-50% of the total processing time. Because this function, a piece of BSD-radix routing code to find a matching subnet for a IP-address, is coded in native C, you can assume its execution is fast. This shows me that the influence of the hash-lookups is minor.

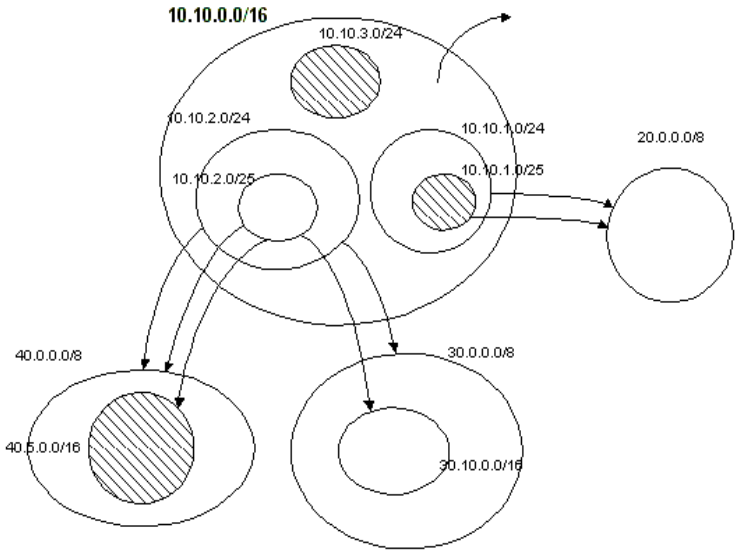
A bad design located in the code of JKFlow lies in the subsequent evaluation of every subnet and every direction. For every subnet and direction there is an individual source and destination IP matching done with Net::Patricia::match_integer. Because all the Net:Patricia::match_integer calls takes a major part of the processing time, I can better make sure that the Net::Patricia::match_integer would be called as less as possible. Therefor I can better make the code so that I would just call the Net::Patricia::match_integer for the source IP address and another for the destination IP address of the evaluated flow record. If I could return references of all the references of the subnets, and directions where I would have to increase the counters of, I would be settled.

This was my first idea to implement a faster algorithm:

I've took the idea to registrate all source subnets in a source Net::Patricia object (also called a “trie”). It is possible for every subnet in a source Net::Patricia trie to register the reference to a destination Net::Patricia trie as returnvalue. After each destination Net::Patricia trie I can place for every destination subnet a return value of a reference to a array. In that array I would place the references of all the datastructures from which I have to increase the couters.

This sounds quite abstract, but I've written some test code (unfortunate not functional):

This is an example of a network with subnets, and directions I want to monitor:



Source Net:Patricia Trie (you need only one source trie)

Source Subnet	Returned Trie
10.10.0.0/16	[0]
10.10.1.0/24	1
10.10.2.0/24	2
10.10.3.0/24 (not)	Undefined
10.10.2.0/25	3
10.10.1.0/25 (not)	[0]

Destination Net:Patricia Tries

Undefined: Don't count anything

Hash references to count:
empty

Trie [0]:
Matches everything: Count into subnet hash

Subnet/directions hash references stack
+subnet subnets="10.10.0.0/16"

Trie 1:

Subnet	Subnet/directions hash references stack
20.0.0.0/8	+direction fromsubnet="10.10.1.0/24" nofromsubnet="10.10.1.0/25" tosubnet="20.0.0.0/8" +subnet subnets="10.10.0.0/16"

Trie 2:

Subnet	Subnet/directions hash references stack
30.0.0.0/8	+direction fromsubnet="10.10.2.0/24" tosubnet="30.0.0.0/8,40.0.0.0/8" +subnet subnets="10.10.0.0/16"
40.0.0.0/8	+direction fromsubnet="10.10.2.0/24" tosubnet="30.0.0.0/8,40.0.0.0/8" +subnet subnets="10.10.0.0/16"

Trie 3:

Subnet	Subnet/directions hash references stack
30.10.0.0/16	+direction fromsubnet="10.10.2.0/25" tosubnet="30.10.0.0/16,40.0.0.0/8" notosubnet="40.5.0.0/16" +direction fromsubnet="10.10.2.0/24" tosubnet="30.0.0.0/8,40.0.0.0/8" +subnet subnets="10.10.0.0/16"
40.0.0.0/8	+direction fromsubnet="10.10.2.0/25" tosubnet="30.10.0.0/16,40.0.0.0/8" notosubnet="40.5.0.0/16" +direction fromsubnet="10.10.2.0/24" tosubnet="30.0.0.0/8,40.0.0.0/8" +subnet subnets="10.10.0.0/16"
40.5.0.0/16 (not)	+direction fromsubnet="10.10.2.0/24" tosubnet="30.0.0.0/8,40.0.0.0/8" +subnet subnets="10.10.0.0/16"

The code on the following pages have I wrote to demonstrate the idea, to bad the code is not fully bugfree.

These are the directions that I register in the code:

-fromsubnets="10.10.1.0/24,10.10.2.0/24", tosubnets="20.0.0.0/8", nofromsubnets="10.10.1.128/25", notosubnets="20.10.0.0/16"
-fromsubnets="10.20.0.0/16", tosubnets="30.0.0.0/8", nofromsubnets="10.20.10.128/25", notosubnets="30.5.0.0/16"
-fromsubnets="10.20.10.0/24", tosubnets="30.10.0.0/16", nofromsubnets="10.20.10.128/26", notosubnets="30.10.0.128/25"

This is the wanted output:

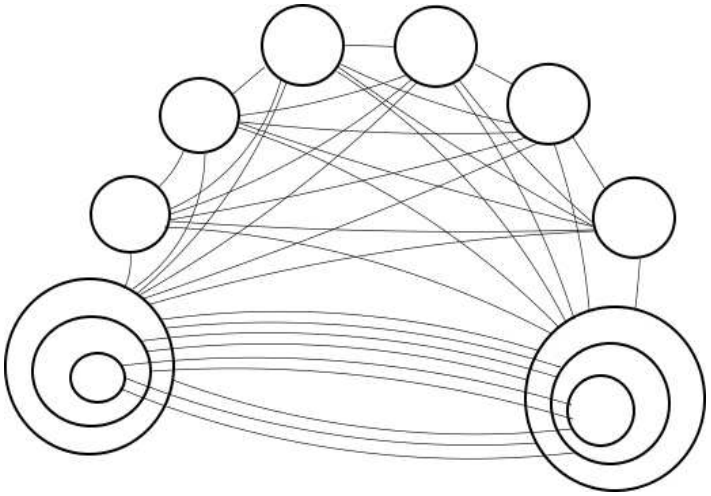
```
SRC=10.20.1.1 DST=30.5.1.1
SRC=10.20.1.1 DST=30.10.0.130
              Data:10.20.0.0/16:30.0.0.0/8
SRC=10.20.10.1 DST=20.1.1.1
SRC=10.20.10.1 DST=30.1.1.1
              Data:10.20.0.0/16:30.0.0.0/8
SRC=10.20.10.1 DST=30.10.1.1
              Data:10.20.0.0/16:30.0.0.0/8
              Data:10.20.10.0/24:30.10.0.0/16
SRC=10.20.10.1 DST=20.10.1.1
SRC=10.20.10.1 DST=30.5.1.1
SRC=10.20.10.1 DST=30.10.0.130
              Data:10.20.0.0/16:30.0.0.0/8
SRC=10.10.1.130 DST=20.1.1.1
```

There is only 1 problem with the code: I call a reference of a Net::Patricia destination trie, and I would like to copy this trie to a new equivalent trie, but copying a object is not possible in Perl. (default not supported, I know there are modules which can do this, but they didn't work either) The problem is in code jargon called a “shallow copy” problem called. It's when you write data to the same datastructure, because you didn't provided a real copy of the datastructure.

```
#!/usr/bin/perl

use Net::Patricia;
use Data::Dumper;

my $myref;

my $srctrie = new Net::Patricia;

sub pushsubnet {

my $srcsubnets=shift;
my $dstsubnets=shift;
my $nosrcsubnets=shift;
my $nodstsubnets=shift;

      foreach $nosrcsubnet (split /,/, $nosrcsubnets) {
            print "NOSRCSUBNET=".$nosrcsubnet."\n";
            foreach $dstsubnet (split /,/, $dstsubnets) {
                  print "DSTSUBNET=".$dstsubnet."\n";
                  my $nofromdsttrie = new Net::Patricia;
                  my $list = [];  # This should be a brand new array!
                  if (defined $srctrie->match_string($nosrcsubnet)) {
                        if (defined ${$srctrie->match_string($nosrcsubnet)}{patricia}->match_string($dstsubnet)) {
                              push @{$list}, @{${$srctrie->match_string($nosrcsubnet)}{patricia}->match_string($dstsubnet)};
                              print "Returned list = ".Dumper([@{$list}]);
                              print "Returned list = ".${$srctrie->match_string($nosrcsubnet)}{patricia}->match_string($dstsubnet)."\n";
                        }
                  }
                  print "Add NOSRCSUBNET to srctrie to notfromdsttrie\n";
                  $srctrie->add_string($nosrcsubnet,{ipsubnet=>$nosrcsubnet,patricia=>$nofromdsttrie});
                  print "Add DSTSUBNET to notfromdsttrie with list:".Dumper([@{$list}])."\n";
                  $nofromdsttrie->add_string($dstsubnet,[ @{$list} ]);
            }
      }
```

```
    foreach $srcsubnet (split /,/, $srcsubnets) {
        print "SRCSUBNET=".$srcsubnet."\n";
        my $dsttrie = new Net::Patricia;
        foreach $dstsubnet (split /,/, $dstsubnets) {
            print "DSTSUBNET=".$dstsubnet."\n";
            my $list = [];  # This should be a brand new array!
            if (defined $srctrie->match_string($srcsubnet)) {
                $dsttrie=${$srctrie->match_string($srcsubnet)}{patricia};      <-Here is an undeep copy problem...
                if (defined ${$srctrie->match_string($srcsubnet)}{patricia}->match_string($dstsubnet)) {
                    push @{$list} , @{${$srctrie->match_string($srcsubnet)}{patricia}->match_string($dstsubnet)};
                    print "Returned list = ".Dumper([@{$list}])."\n";
                    print "Returned list = ".${$srctrie->match_string($srcsubnet)}{patricia}->match_string($dstsubnet)."\n";
                }
            }
            foreach $nodstsubnet (split /,/, $nodstsubnets) {
                print "Add NODSTSUBNET to dsttrie to list:".Dumper([@{$list}])."\n";
                $dsttrie->add_string($nodstsubnet,[ @{$list} ]);
            }
            push @{$list}, "$srcsubnet:$dstsubnet";
            print "Ref List=".$list."\n";
            print "Ref dsttrie=".\$dsttrie."\n";
            print "Add SRCSUBNET to srctrie to dsttrie\n";
            $srctrie->add_string($srcsubnet,{ipsubnet=>$srcsubnet,patricia=>$dsttrie});
            print "Add DSTSUBNET to dsttrie with list:". Dumper([@{$list}])."\n";
            $dsttrie->add_string($dstsubnet,[ @{$list} ]);
        }
    }
}

pushsubnet ("10.10.1.0/24,10.10.2.0/24","20.0.0.0/8","10.10.1.128/25","20.10.0.0/16");
pushsubnet ("10.20.0.0/16","30.0.0.0/8","10.20.10.128/25","30.5.0.0/16");
pushsubnet ("10.20.10.0/24","30.10.0.0/16","10.20.10.128/26","30.10.0.128/25");
#pushsubnet ("10.20.0.0/16","30.0.0.0/8","","");
#pushsubnet ("10.20.10.0/24","30.10.0.0/16","","");


foreach my $srcip ("10.10.2.1", "10.20.1.1", "10.20.10.1", "10.10.1.130", "10.20.10.130","10.20.10.130") {
    foreach my $dstip ("20.1.1.1","30.1.1.1","30.10.1.1","20.10.1.1","30.5.1.1","30.10.0.130") {
        print "SRC=".$srcip." DST=".$dstip."\n";
        #print Dumper($srctrie->match_string($srcip))."\n";
        if (defined $srctrie->match_string($srcip)) {
            $srcsubnetmatchip=${$srctrie->match_string($srcip)}{ipsubnet};
            $srcsubnetmatchref=\${$srctrie->match_string($srcip)}{patricia};
            if (defined ${$srctrie->match_string($srcip)}{patricia}->match_string($dstip)) {
                $dstsubnetmatch=${$srctrie->match_string($srcip)}{patricia}->match_string($dstip);
                foreach $value (@{${$srctrie->match_string($srcip)}{patricia}->match_string($dstip)}) {
                    print "   Data:".$value."   ".$srcsubnetmatchip.",".$srcsubnetmatchref.",".$dstsubnetmatch."\n";
                }
            }
        }
    }
}
```

After a few weeks thinking about this problem I understood it was not necessary to copy the Net::Patricia objects at all, even better: I don't have to maintain a combined Net::Patricia source/destination trie at all! Why? Well, lets take a large example of 100 sites each with its own subnets. We want to monitor the traffic between every site so we have a combination of 100x99/2 = 4450 directions between all sites. If each site contains 2 further subnets, this will result in 300x299/2 = 44850 directions to monitor! This seems enormious but it is not so bad as it sounds. A Net::Patricia trie-match on the source and destination IP would only result in 2 source and 2 destination subnets if none of these 2 further source and destination subnets aren't subnets of eachother. These 2 source and destination subnets results in only 2x2=permutation of 4, and even if the all the subnets are subnets of eachother (source subnet inside a source subnet inside another source subnet and a destination subnet inside a destination subnet inside another destination subnet) this would result in only 3x3=permutation of 9. It is very unlikely that you would monitor 4 or more subnetted inside eachother subnets against eachother, so the amount of combinations resulting with the source and the destination Net::Patricia match subnets will not become as large as the combinations between all 100 sites.



In the new JKFlow2 report module I use now a hash containing a “included”-key pointing to a array of matching from/tosubnets and a “excluded”-key pointing to a array of matching nofrom/notosubnets.

```
Net::Patricia->match_integer( '10.101.10.5' )
       ->{    'included'=>[ '10.0.0.0/8','10.101.0.0/16' ],
              'excluded'=>[ '10.101.10.0/24' ]        }
```

This functionality is implemented for the source and destination Net::Patricia Tries.

The resulting 'included' source and destination subnets are in a permutation combined to source and destination subnet pairs. These pairs are registrated in $JKFlow::mylist{subnets} and contains the following structure:

```
$JKFlow::mylist{subnet}{$srcsubnet}{$dstsubnet} contains:
an array:           [
of hashes:            {    'nofromsubnets' => [ ],
                           'notosubnets' => [ '10.101.10.0/24' ],
                           'ref' => $ref             },
                           ...
                    ]
```

The 'ref'-key contains the reference of any direction matching to the source and destination subnet. The nofrom/notosubnets are compared with the returned 'excluded' subnets and discards the direction if subnets matches. The array makes it possible to register multiple directions behind a source/destination subnet pair. The returned reference points to the hash containing the hash structure explained before. The internal working of the JKFlow2 module remains the same.

The differences in JKFlow and JKFlow2 lies with this change in the evaluation of the directions. JKFlow2 contains beside the original and now unused countDirections a new countDirections2 function wich incorporates the new evaluation. Inside the 'wanted' function the old countDirections calls are commented, and a single countDirections2 call is placed a the end.

```
countpackets(\%{$JKFlow::mylist{'all'}},$which);
countApplications(\%{$JKFlow::mylist{'all'}{'application'}},$which);
#countDirections(\%{$JKFlow::mylist{'all'}{'direction'}},$which);
countDirections2();
```

A problem was implementing the population of the source and destination Net::Patricia tries. Critical was to registrate the subnets in the correct order. Because I had to registrate both source and destination subnets tries I could never manage this correctly by using a single DOM-tree of directions. Therefor I've created a fromsubnets and a tosubnets element inside the JKFlow.xml configuration file, containing the subnets used in the directions. You can take a look to the example configuration file provided within the jkflow2-v08092003.tgz. Defining these subnets is critical for correctly directions matching with flows:

```
<fromsubnets>
       <subnet       subnets="10.240.0.0/16"    nosubnets="">
       </subnet>
       <subnet       subnets="10.241.0.0/16"    nosubnets="">
       </subnet>
       <subnet       subnets="10.101.0.0/16"    nosubnets="">
       </subnet>
       <subnet       subnets="10.103.0.0/16"    nosubnets="">
       </subnet>
       <subnet       subnets="10.247.0.0/16"    nosubnets="">
       </subnet>
</fromsubnets>
<tosubnets>
       <subnet       subnets="0.0.0.0/0"
                     nosubnets="10.103.0.0/16,10.240.0.0/16,
                            10.241.0.0/16,10.101.0.0/16,10.247.0.0/16">
       <subnet       subnets="10.103.0.0/16"
                     nosubnets="">
       </subnet>
       <subnet       subnets="10.240.0.0/16"
                     nosubnets="">
       </subnet>
</tosubnets>
```

These subnets corresponds with the subnets of the directions, but splits these into fromsubnets and tosubnets. This was the only way to make it possible to make a DOM-tree (Document Object Model, the tree structure of the XML document) structure configuration possible.

In the latest version of JKFlow.pm in jkflow2-v10092003.tgz is the necessarity of defining these subnets removed, with the replacement of the pushDirections2 with the pushDirections3 function. JKFlow.pm automaticaly extracts the needed subnets from its configuration. This feature is not fully tested yet, but it works with my configuration without any problems.

To configure JKflow2:

1:- Configure the system with the original JKFlow module.
2:- Replace the JKFlow.pm module with the module of JKFlow2.
(JKGrapher.pl remains the same)
3:- Add the necessary fromsubnets and tosubnets elements in the JKFlow.xml file (as above)
(step 3 can be discared with jkflow2-v10092003.tgz)
4:- Test FlowScan!

Testing:

Some abilities are included within the code. It is possible to define a monitor=”yes” attribute inside a direction element, like this:

```
<direction   name="Engeland-Internet"
```

```
fromsubnets="10.241.0.0/16"
tosubnets="0.0.0.0/0"
notosubnets="10.241.0.0/16"
monitor="yes">
```

This will result in reporting of the matching flows to STDOUT.
Also it is possible to uncomment the countDirections calls in the “wanted” function. This will result in evaluating the flows in BOTH sequentialy,recursively like in JKFlow and with the new way of JKFlow2. Providing the monitor-attribute will result in reporting the direction matching in both ways to compare the result, for tracing errors in the configuration of the fromsubnets and tosubnets elements. For every match in the old JKFlow direction matching, tagged with “D1”, there should be another match with the new JKFlow2 direction matching, tagged with “D2”.

```
D1 SRC=10.101.18.113,SRCSUBNET=10.101.0.0/16,DST=10.240.72.69,DSTSUBNET=10.240.0.0/16
D2 SRC=10.101.18.113,SRCSUBNET=10.101.0.0/16,DST=10.240.72.69,DSTSUBNET=10.240.0.0/16
D1 DST=10.101.18.113,DSTSUBNET=10.101.0.0/16,SRC=10.240.72.69,SRCSUBNET=10.240.0.0/16
D2 DST=10.101.18.113,DSTSUBNET=10.101.0.0/16,SRC=10.240.72.69,SRCSUBNET=10.240.0.0/16,
EXPORTER=174455042
```

Note that it is not necessary that every D1 match will be followed by a D2 match. If a flow results in 3 D1 matches, the 3 following D2 matches may have a different order. There must be the same amount of D1 as D2 matches however.

In JKFlow3 you will notice you can only track the evaluations defined inside in the “all” element in this way, because router/subnet elements were replaced with routergroup, and it didn't make much sense to add the recursive direction definition in routergroups, and the top defined direction element . If you are in the case you would like to debug a top defined direction with directions inside, place the directions inside the 'all' element, and uncomment the countdirections inside the all evaluations in wanted().

The new method makes JKFlow2 about 30%-70% faster than JKFlow, depending on configuration of your JKFlow.xml. If you have a lot of directions inside same elements, you will notice that flowfiles parsing will get faster a lot!

Testing has shown that a configuration of 1 “all”, 1 “subnet” and 8 directions within this “subnet”, makes the JKFlow2 module 35% faster than JKFlow. This seems not as much, but the important point is that defining more directions won't make the module slower anymore. I've increased the amount of directions to 12, restarted the test, and noticed no difference in flow processing speed. If I've defined about 100-1000 directions the reporting of all the data in the RRDTool files would be the bottleneck of JKFlow2.

If I would compare JKFlow2 against CUFlow, I see however some nuances. CUFlow is good at parsing flowfiles FAST. JKFlow2 could keep with 30-50% CUFlow's speed at best, but this ignores where the modules would performs at best.

CUFlow is at best were you need raw processing power of network traffic of different routers, and you don't mind monitoring subnets of directions. Adding extra routers should not much degrade the speed. JKFlow2 is at best where you need fast processing power of network traffic and you need to monitor directions (better yet: lots of directions) of source / destination subnets. Adding extra directions should not much degrade the speed.

Tip: You could comment out the D2 reporting lines in countDirections2 to gain some extra speed.

3.10: Principes behind improvements of JKFlow version 3

Definesets / Sets

JKFlow version 3 removed the router/subnet/network elements and put the direction as the basis of the configuration. I've added routergroup, sites, and definesets elements.

Adding definesets was remarkable easy in front of parseDirection:

```
if (defined $refxml->{set}) {
        foreach my $set (keys %{$refxml->{set}}) {
                print "parseDirection: ".$set."\n";
                parseDirection(\%{$config->{definesets}{defineset}{$set}},$ref);
        }
}
```

Routergroups

Routergroups in JKFlow 3 allowed monitoring routers and router interfaces by linking those to a direction. To do this I've created In JKFlow 3.2 I've added the possibility to combine routergroups and directions. For directions without subnets from/to, nofrom/noto, fromsubnets/tosubnets, nofromsubnets/notosubnets, this is accomplished by using these hashes:

First a lookup of the matching routergroups is done on basis of the exporter address and the interface:

```
$JKFlow::mylist->{router}{$router}{interface}{$interface} = []
$JKFlow::mylist->{router}{$router}{localsubnets} = []
```

(array of routergroups)

Then all linked directions are found via the routergroups. Note that the directions with subnets are not added to this array, because they are evaluated anyway.

```
$JKFlow::mylist->{routergroup}{$routergroup} = []
```

(array of references to directions)

In the directions with subnets from/to, nofrom/noto, fromsubnets/tosubnets and nofromsubnets/notosubnets attributes, the routergroup matching is done after the direction evaluation in the function &countFunction2 from which reference is called from within the direction, if a routergroup attribute was found during parsing:

```
if (    defined $direction->{router}{$exporterip} &&
        defined $direction->{router}{$exporterip}{$input_if}) {
                countpackets(\%{$direction},'in');
                countApplications(\%{$direction->{'application'}},'in');
}
if (    defined $direction->{router}{$exporterip} &&
        defined $direction->{router}{$exporterip}{$output_if}) {
                countpackets(\%{$direction},'out');
                countApplications(\%{$direction->{'application'}},'out');
}
```

Combining routergroups

In JKFlow 3.2 I've added the possibility to combine routergroups and directions. With the introduction of the fast evaluation of directions based on a single source and destination Net::Patricia match, JKFlow 2 silently lost a feature: you couldn't monitor directions on specific routers anymore. This is because in JKFlow 2 every direction is evaluated, even when they are defined inside routers, subnets and even other directions, because the recursive evaluation was replaced. I've added the feature to evaluate the routers back with JKFlow 3.2 and added router interfaces evaluation also. The method used in JKFlow1 was: evaluate the router first, then evaluate the direction(s). The method implemented in JKFlow 3.2 is now: evaluate the directions, and in the directions, where routergroups are defined, evaluate the routers/interfaces.

This means that for every direction a different logic is implemented, based on if a routergroup attribute is defined or not. This is done by linking references to countFunction1/countFunction2 in the directions, and calling the right countFunction via the reference.

A problem was how to handle subnets based directions without from/to, nofrom/noto, fromsubnets/tosubnets, nofromsubnet/notosubnets attributes. These directions are not evaluated during direction evaluation, so these are explicit evaluated based on exporter address / interface. This happens here below. (In JKFlow 3.0 and 3.1 the usage of routergroups attribute in subnet based is invalid and will result in wrong data, but in JKFlow 3.2 this is okay)

```
# This may have some explanation... Why don't push references of directions into
# $JKFlow::mylist{routergroup} if any subnets are defined in the direction ? This is
# because in countDirections2() these directions will call countFunction2 (registered here
# above, because these directions contains routergroup attributes) and will call
# countPackets and countApplications by itself so we won't allow calling these functions
# from within the router section of wanted().

if (     defined $refxml->{$direction}{'routergroup'} &&
         !defined $refxml->{$direction}{'fromsubnets'} &&
         !defined $refxml->{$direction}{'tosubnets'} &&
         !defined $refxml->{$direction}{'nofromsubnets'} &&
         !defined $refxml->{$direction}{'notosubnets'} &&
         !defined $refxml->{$direction}{'from'} &&
         !defined $refxml->{$direction}{'to'} &&
         !defined $refxml->{$direction}{'nofrom'} &&
         !defined $refxml->{$direction}{'noto'}) {
         my $list=[];
         my $routergroup=$refxml->{$direction}{"routergroup"};
         $list=$JKFlow::mylist{routergroup}{$routergroup};
         print "Assign routergroup ".$routergroup." to ".$direction."\n";
         push @{$list},$ref->{$direction};
         $JKFlow::mylist{routergroup}{$routergroup}=$list;
         }
```

3.11: Parsing JKFlow.xml

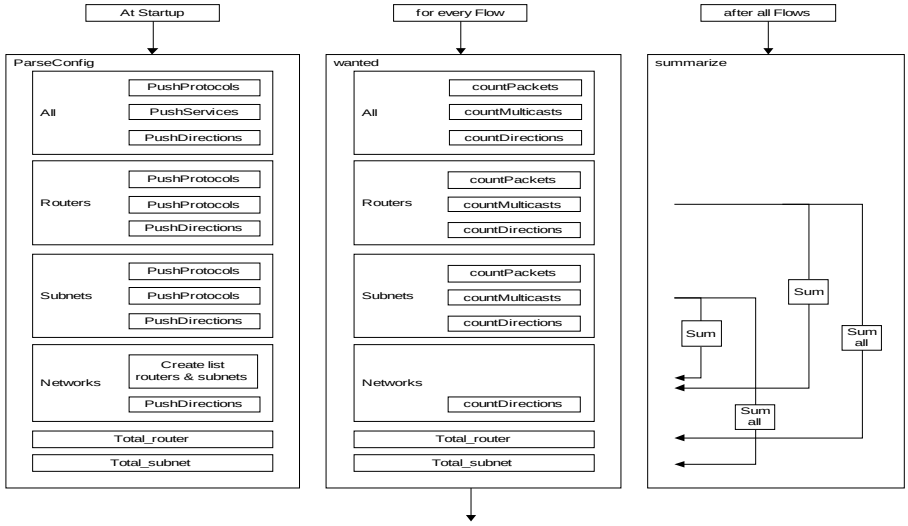
The configuration of JKFlow.pm is done using a XML file. Parsing of this file is done with XML::Simple, en is as its name says, easy. There is even no validation on a DTD or Schema done. I've used this module for having a framework for the configuration, and it helped me because I've written +-50% less code for parsing.

If you look into the code you can see that XML::Simple by itself will parse the XML-file to a hash-tree. Here you can ask yourself: “why creating a new tree while you have already a tree?”. 1:-I've started code this way, and added the configuration afterwards 2:-because changes in configuration layout may not affect the internal code, that's asking for problems.

Why not using XSLT to process this DOM-tree to the internal tree? 1:- I didn't wanted to go this far, and I didn't had the time either. 2:-XSLT doesn't support loops, I would have to write code myself to evaluate strings like 137-139/tcp, 137-139/udp.

3.12: Structure, development of JKFlow.pm

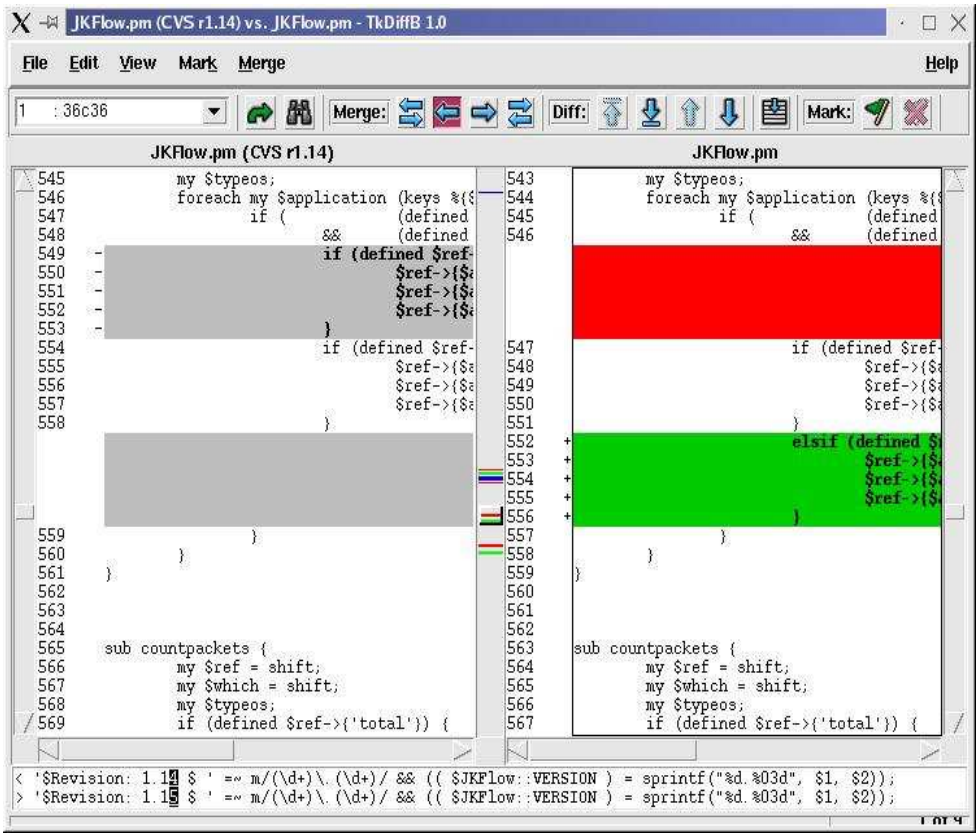
This is a scheme of the internal structure of the module. I consider that I've split the structure in its global and specific aspects very well out. There is one facet that bother me: during parsing of the configuration file there are 3 functions called: PushProtocols, PushServices and PushDirections, while during counting the flows the functions countPackets, countMulticast and countDirections are used. The functions PushProtocols and PushServices stands not in direct relationship with the countPackets en countMulticasts functions, but initialise the same datastructures.



Structure of JKFlow.pm

*(*remark: in the latest version is the function "countDirections" migrated inside "countPackets")*

For the development of the module I've used a versioncontrol tool known as "CVS" (Concurrent Version System). A well-known frontend for this tool is TkCVS, which allowed me to keep an eye on the different version: The red marked blocks are pieces of code which are removed, and the green ones are pieces of new code:



To give an impression on the development of JKFlow.pm:

> CUFlow.pm counted 1619 lines
> JKFlow.pm counted 898 lines, is 40 % smalled and 95% rewritten.
> (Update: The latest version of JKFlow 3.4.1 counts 2176 lines, and contains much more functionality)

I wrote and tested the code in RedHat Linux 8.0, 9.0, Fedora Core 1,2,3 and CentOS 4, and Solaris. Debugging is mostly done by inserting Data::Dumper lines. In the code are still some %mylist dump lines commented out with #.

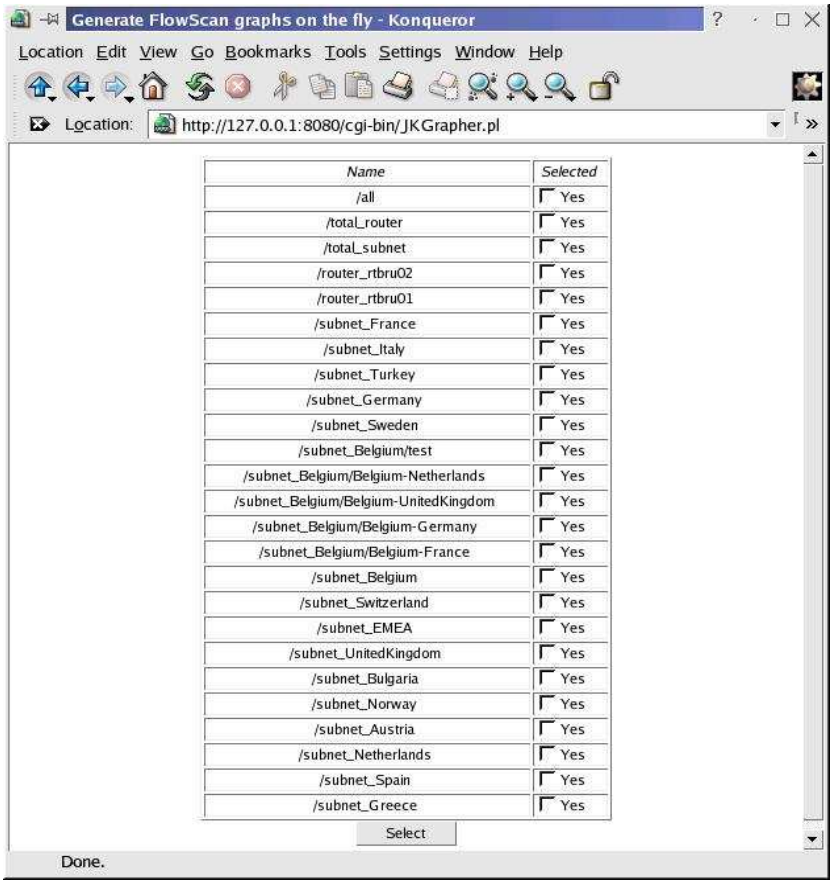
3.13: JKGrapher.pl

Rewriting of CUGrapher.pl to JKGrapher.pl was not easy. Within the original code where several functions dependent on each other, so it was hard to modify the code to make some critical changes.

- I have removed every reference to a configuration file. JKGrapher.pl doesn't need the configuration file, only the RRDTool database directories.
- A quite important problem I had to solve was the removal of a hash which contains all the filenames of the RRDTool files, and previously were filled in at forehand, via parameter passed by CGI. The same had I to do with the colors of the graphs.
- Also I had to closely watch in which order I passed RRDTool commands to RRDTool. If I made some mistake in the code, I received a unclear error message. Luckily I could examine the passed RRDTool command using an extra CGI-parameter: “...&debug=1”.

- Another problem was making sure that protocols, services, TOS and/or total of the graphs didn't get stacked to eachother, and this for all routers,subnets and directions. I could solved it by using an array to place the commands for the services, protocols, TOS and total separately.

It did take some time for getting JKGrapher.pl to work. Personaly I don't consider the code really important for discusion in this document, and it is not really needed to understanding the working of JKFlow.



JKGrapher.pl during selection of routers,subnets

In JKGrapher.pl is Perl-CGI extensively used. This choose is reasonable because the original CUGrapher.pl was already a CGI-script, and its execution was fast. There is not image-chache used like in some other frontends, because the image generation is fast.

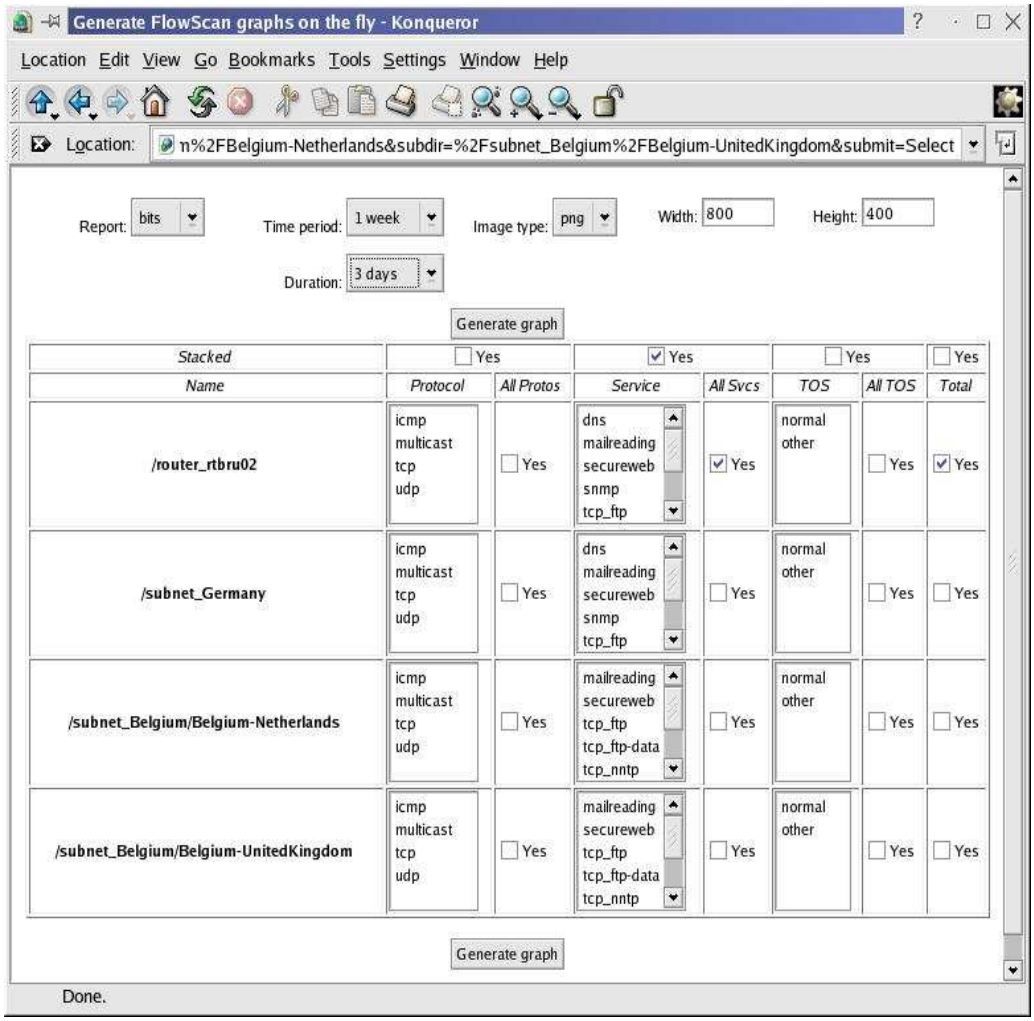
3.14: Automating creation & archive RRDTool graphs

If you want to create and display/archive graphs without using the JKGrapher.pl CGI-script you can make use of “wget”. Wget is a tool for webretrieval, and can be used for retrieving and saving of RRDTool graphs to files instead of creating dynamically via the webserver. You can use this if you don't want to expose your webserver using CGI-scripts. All you have to do is create a script containing wget-commands like these:

```
wget -O picture-`date +%d-%m-%Y`.png
'http://127.0.0.1/cgi-
bin/JKGrapher.pl?report=bits&hours=24&imageType=png&wid
```

```
th=800&height=400&duration=24&predefinedcolors=1&servic
e_stacked=1&%2Fall_all_services=1&%2Fall_total=1'
```

The wanted URL can be found using the webbrowser, and is placed inside tildes to prevent evaluation by the shell. The -O option writes the retrieved picture to a file. The filename is dynamically created using the date function and backwards tildes. This example could be used in a cron.daily script.



Screenshot of JKGrapher.pl after selection of routers, subnets, directions

3.15: Safety of CGI-scripts

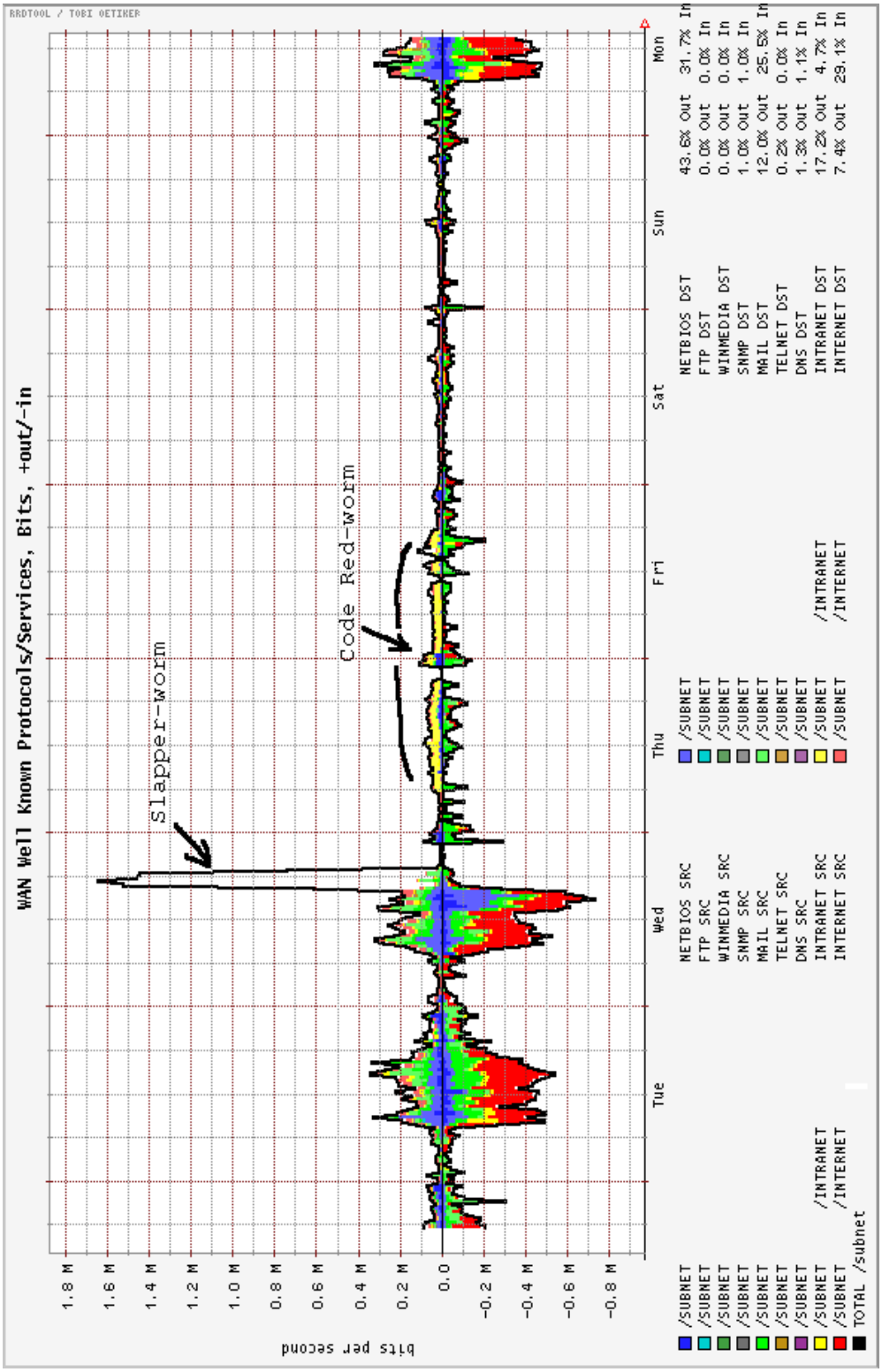
There is one thing you must be aware of in consequence of using Perl-CGI: security.

I did not taken any special precoursions while programming JKGrapher.pl. In the script are however no system-commands called with CGI-parameters, but internal in the script is a RRDTool command generated from parameters which are returned from a CGI-form.

If you're planning to use this tool on a puplic webserver, you should run the Apache-server under an account with minor user rights, and if you really wants to make sure nothing could happen, you should setup the Apache-server in a chrooted environment.
These remarks were actually also valid for CUGrapher.pl, btw...

3.16: Example Graph

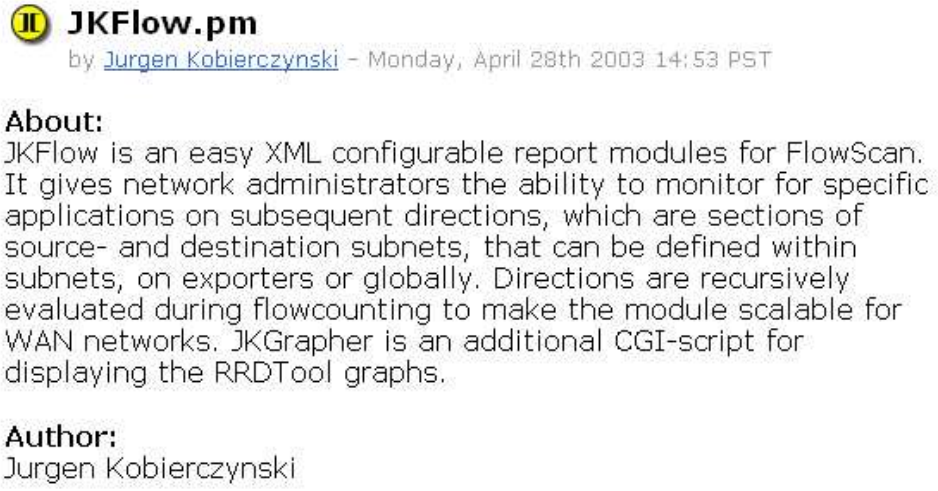
This graph shows an outbreak of the Slapper-worm (port 1443=MS-SQL) and the Code Red-worm (Port 80=HTTP) from the intranet from ... to the intranet of ...



3.17: JKFlow available on the Internet

Because this module is more flexible configurable that others, I've decided to place the JKFlow module on the Internet, submitted it to Freshmeat, and sended messages to the CUFlow and FlowScan mailing lists. There were some good reactions from people who were waiting on a module who could handle monitoring directions.

JKFlow.pm is available on: http://users.telenet.be/jurgen.kobierczynski



One last tip in using JKGarpher.pl: If you want to create graphs on regulary timesteps, you can make use of the “wget” tools, allowing file downloads of URL's

Conclusion

This project delivered to ... a working solution for network bandwidth profiling to monitor the different services over carriers. The result, besides a working implementation, is a new module which allows FlowScan a better solution to monitor network services.

I like to thanks Koen Depovere for the opportunity to do this project.

Jurgen Kobierczynski

Appendix A: FlowScan Troubleshooting

Versions and software:

- Use GCC compiler **2.95.2** or **2.95.3** for Arts++ and cflowd
- Use GCC compiler **3.3.x (not 3.4)** for flow-tools, or patch flow-tools with the provided patch in the JKFlow package
- I don't have tried RRDTool 1.2 yet.

Tips for Operating Systems:

I've installed Flowscan in several OS's, with changing succes:

- RedHat 8.0, 9.0, Fedora 1, 2, 3, Mandrake 9.1, 10.0 works
- Fedora 3 for AMD64 does not work :-((problems compiling of flow-tools)
- Solaris 8 en Solaris 9 works, but it will takes some effort
- FreeBSD 5 doesn't work, troubles with buiding Boulder:: Stream
- OpenBSD could work, I didn't get behind Arts++. The installation problems are much like RedHat 8.0

Tips for Solaris 9:

- Use for Solaris 9 if there is no gnu-tool available, the tool for Solaris 8.
- Use not the newest version of Bison in Solaris 9, like versie 1.75. (compilation problems for cflowd)

Tips for Solaris generaly:

- Compile GCC from the source and install it under /usr/local.
- Use the –enable-shared flag while configuring of the GCC source.
- Change the PATH so /usr/local/bin/gcc is default executed for compiling Arts++,Cflowd.
- Also use the –libdir and the –includedir options for compiling of Arts++, Cflowd.
- Install a GCC compiled version of Perl. Perl must call several modules which are also compiled with GCC, which the native Perl included in Solaris is not capable of. I've always used version 5.8

These are some problems you can get while installing FlowScan in Solaris and/or Linux. Every problem has a short description, and is followed by a description of the cause and the solution.

Problem:
not well-formed (invalid token) at line x, column yy, byte zz at /usr/lib/perl5/site_perl/5.8.0/i386-linux-thread-multi/XML/Parser.pm line xxx

Solution:
This is not a bug in JKFlow.pm of the Parser. The JKFlow.xml file has no valid XML syntax. Check the XML syntax in JKFlow.xml.

Problem:
During untarring of Cflowd I get a "Directory Checksum Error".

Solution:

Install GNU Tar, and put in the PATH environment variable the path /usr/local/bin before /usr/bin, so GNU Tar is called before Solaris Tar.

Problem:

During strating of cflowmux I get a message saying dat there isn't enough "shared memory" available.

Solution:

In the kernel must the maximum amount of memory be set big enough so the Cflowd daemons could work.

- In Solaris, as root place the following rule in /etc/system and restart the system:
  shmsys:shminfo_shmmax=16777216
- In Linux, run as root the following command:
  echo 16777216 > /proc/sys/kernel/shmmax

Problem:

libstd++.so.x.x is not found when starting of cflowd, cflowmux.

Solution:

There are 2 configuration solutions possible: one is session-based and the other is system-based. This is easy to configure, but the setting is not persistent, and disappears after closing of the shell where you've issued the command. Also the syntax of the command is different in different shells (bash, ksh, ...)
In the systemwide solution you reconfigure the librarypath with "crle". With this tool you reconfigure the "Runtime Linking Environment". These changes are permanent and must be carefully be done. A misconfiguration could result in that (re)starting of the desktop or the system will fail in starting the system and/or the windowsmanager, dumping you into "rescue" mode. Test before closing your session if you can start CDE applications, and try to resolve libraries using the "lld" tool.

These are some commands that you've to use:

#crle –l /usr/lib:/lib:/usr/local/lib:/usr/ccs/lib:/usr/dt/lib/ -i /usr/lib –i /lib –i /usr/local/lib –i /usr/ccs/lib/ -i /usr/dt/lib/
#crle (to check the changes)
#ldd <executable> (to check if you can resolve libraries)

In the case you've problems, and you gets into a rescue-session, I have the following tips:

- Check with 'ldd' if all libraries can be resolved of for binaries.
- There are several 'fsck'-s on the systeem, for different filesystemtypes. If you got trouble starting fsck for your filesystem, check the library resolving for the correct fsck of your filesystemtype.
- You will have to remount your / filesystem in read-write mode.

If you are ready with reconfiguring you must try restarting the system without problems starting the X Window Systeem or the CDE window manager. (golden rule: always reboot after important changes: you can better solve problems now, than someone else later!)

Problem:

After solving different problems in compiling cflowd you get several errormessages pointing to linkpaths and libraries.

Solution:

This problem is the result of several problems while compiling cflowd, which were solved but makes a new configuration/compilation needed. Delete the compilation tree/source, and starts over with a new cflowd source + patch. (this sound vague, but it works for me)

Problem:

You get an errormessage in the bigendian test in the configure script of Arts++.

Solution:

Remove the 'exit 1' clausule in the configure script. Look for the word "bigendian", and then for "exit".

```
fi
rm -f conftest*
else
  echo "configure: failed program was:" >&5
  cat conftest.$ac_ext >&5
fi
rm -f conftest*
if test $ac_cv_c_bigendian = unknown; then
if test "$cross_compiling" = yes; then
    { echo "configure: error: can not run test program while
cross compiling" 1>&2; exit 1; }
else
  cat > conftest.$ac_ext <<EOF
```

Problem:

I get an "internal" compiler error while compiling of Arts++.

Solution:

Try compiling of Arts++ and cflowd with the GCC version 2.95.2 or 2.95.3 downloadable from the ftp-server ftp.sunfreeware.com. I've noticed that things works out using this version of GCC. If not, download the source of GCC and reconfigure is with the --enable-shared option, so when you compile the compiler you compiles the library libstd++.so.x.x also.

Problem:

Compiling of Arts++ and/or cflowd eates all diskspace!

Solution:

The linked libraries in the apps subdirectory takes in Solaris about +-20 megabyte for each file, and they includes for every file a lot of symbolic information ment for debugging purposes. Because you have quite a few of these files, you will lose about 400 megabyte of diskspace. In Linux is this problem also there, but the size per file is about 8 megabytes.

Before you install these files, or between different “make” invocations, you must make room by “stripping” the binaries and libraries. This will remove the symbolic information.

#strip <binary>
You can restart the make process, with just entering “make” + Enter

Problem:
The installation of GNU m4 for Bison, doesn't solve the problem of compilation of cflowd caused by a missing -l option in m4 delivered by Solaris.

Solution:
M4 (a processing language) is hardcoded called by Bison (a parsing environment based on Yacc for C), and doesn't check the PATH environment variable. By this the / usr/ccs/bin/m4 is still been called, while “which m4” shows that the m4 default calls to / usr/local/bin/m4 ! The solution is to rename the old m4 binary to m4.old and to create a symbolic link m4 to /usr/local/bin/m4:
c++ -g –02 –I../include –I../../include –I/usr/local/arts/include –
I../../snmp++/classes/include –c CflowdConfig.cc –o CflowdConfig.o
echo timestamp > CflowdConfig.lo
bison –d –b flowfilt –p flowfilt FlowFilter.y **/usr/ccs/bin/m4**: bad option: -I
c++ -g –O2 –I../include –I../../include –I/usr/local/arts/include –c FlowFilter-Lex.cc –o FlowFilterLex.o
FlowFilterLex.cc:46: flowfilt.tab.h: No such file or directory
make[2] : *** [FlowFilterLex.lo] Error 1
make[2] : Leaving directory ‘/export/home/jurgenk/install/cflowd-2-1-b1/classes/src’
make[1] : *** [all] Error 2
make[1] : Leaving directory ‘/export/home/jurgenk/install/cflowd-2-1-b1/classes’
make: *** [all] Error 2
bash-2.03$ which m4
/usr/local/bin/m4

Problem:
In the end fase during the compilation of cflowd is the tool “ar” not found .

Solution:
“Ar” is the archive tool for creation of portable libraries of archives. It is located in / usr/ccs/bin and this directory is added to the path.

Problem:
Cflowd crashes regulary

Solution:
Checks if in the logfile /var/adm/log/messages, /var/log/messages contains this rule:
failed to release buffer lock: Resource temporarily unavailable
Cflowd can crash under heavy lead. The problem can be solved parly by increasing the maximum shared segment. If not solved try using flow-tools.

Problem:

The total traffic graph doesn't appear.

Solution:

Reported with Redhat 8.0, you may need to change the definition of the black color from #000000 to #00000F. (There are 2 places in JKGrapher.pl where you have to change it.)

Problem:

You have empty graphs, the error message appears: Invalid index in cflowd flow file: 0xCF100103! Version 5 flow-export is required with ***all*** *fields being saved.*

Solution:

Install the Cflow module after installing Flow-tools. You have still the Cflow module compiled for Cflowd. You have to install Flow-tools from the source before compiling Cflow, in using the correct include files delivered with Flow-tools. The file (Cflow-1.051.tar.gz) included with the sources of flow-tools is actually the same as the file pointed with the installation instructions of Cflowd, so it all depends on the installed include files of Cflowd/flow-tools.

Problem:

Help: the error message: "option '/var/flows/...'" appears when writing RRD-Files!

Solution:

Patch FlowScan.pm for Flow-tools.

```
patch -p0 /usr/local/bin/FlowScan.pm < FlowScan.pm_flow-tools_patch
```

Problem:

While compiling Flow-tools I get the error message "label at end of compound statement".

Solution:

Or install GCC 3.3.x (GCC 3.4.x introduced this problem)
Or patch flow-tools with the provided patch in jkflow-3.5alpha.tgz

```
tar -zxvf flow-tools-0.67.tar.gz
patch -p0 < flow-tools-0.67_gcc3.4_patch
```

Problem:

I can't get the JKGrapher.pl CGI script to work in Fedora 3, RHEL4, CentOS4 (any SELinux distribution)

Solution:

When installing on a system using SELinux (like Fedora 2, 3, REHL4, CentOS4), you may have to enable CGI binaries in Apache. This can be configured using a boolean flag. If things don't work out, you can also disable SELinux with the command 'setenforce 0'.

Appendix B: (old!) JKFlow.xml example (version 1,2)

```
<config>
        <all localsubnets="10.0.0.0/8">
                        <application name="web">80/tcp,8080/tcp</application>
                <application name="secureweb">443/tcp</application>
                <application name="windows">
                        137-139/tcp,137-139/udp,42/tcp,135/tcp,445/tcp
                </application>
                <application name="dns">53/udp,53/tcp</application>
                <application name="snmp">161-162/udp</application>
                <services>22-23/tcp,25/tcp,102/tcp,110/tcp,119/tcp,143/tcp</services>
                <protocols>tcp,udp,icmp</protocols>
                <ftp/>
                <multicast/>
                <tos/>
                <total/>
        </all>
        <routers>
                <router                 routers="10.1.1.1"
                                name="routers1"
                                localsubnets="10.0.0.0/8">
                        <application name="web">80/tcp,8080/tcp</application>
                        <application name="secureweb">443/tcp</application>
                        <application name="mailreading">110/tcp,143/tcp</application>
                        <application name="windows">
                                137-139/tcp,137-139/udp,42/tcp,135/tcp,445/tcp
                        </application>
                        <application name="dns">53/udp,53/tcp</application>
                        <application name="snmp">161-162/udp</application>
                        <services>20-23/tcp,25/tcp,102/tcp,119/tcp</services>
                        <protocols>tcp,udp,icmp</protocols>
                        <multicast/>
                        <tos/>
                        <total/>
                </router>
                <router                 routers="10.2.2.2,10.3.3.3"
                                name="routers2"
                                write="no">
                        <application name="web">80/tcp,8080/tcp</application>
                        <application name="secureweb">443/tcp</application>
                        <application name="mailreading">110/tcp,143/tcp</application>
                        <application name="windows">
                                137-139/tcp,137-139/udp,42/tcp,135/tcp,445/tcp
                        </application>
                        <application name="dns">53/udp,53/tcp</application>
                        <application name="snmp">161-162/udp</application>
                        <services>20-23/tcp,25/tcp,102/tcp,119/tcp</services>
                        <protocols>tcp,udp,icmp</protocols>
                        <multicast/>
                        <tos/>
                        <total/>
                </router>
        <total_router/>
        </routers>
        <subnets>
                <subnet         subnets="10.129.0.0/16"
                                name="Belgium"
                                localsubnets="10.129.0.0/16"
                                write="yes">
                        <direction      name="Belgium-Internet"
                                        fromsubnets="10.129.0.0/16"
                                        tosubnets="0.0.0.0/0"
                                        notosubnets="10.129.0.0/16">
                                <application name="web">80/tcp,8080/tcp</application>
                                <application name="secureweb">443/tcp</application>
                                <application name="mailreading">
                                        110/tcp,143/tcp
                                </application>
                                <application name="windows">
                                        137-139/tcp,137- 139/udp,42/tcp,135/tcp,445/tcp
                                </application>
                                <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
                                <protocols>tcp,udp,icmp</protocols>
                                <ftp/>
                                <multicast/>
                                <tos/>
                                <total/>
                        </direction>
                        <direction      name="Transit"
                                        nofromsubnets="10.129.0.0/16"
                                        notosubnets="10.129.0.0/16">
                                <application name="web">80/tcp,8080/tcp</application>
                                <application name="secureweb">443/tcp</application>
                                <application name="mailreading">
                                        110/tcp,143/tcp
```

```
                                </application>
                                <application name="windows">
                                        137-139/tcp,137-139/udp,42/tcp,135/tcp,445/tcp
                                </application>
                                <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
                                <protocols>tcp,udp,icmp</protocols>
                                <ftp/>
                                <multicast/>
                                <tos/>
                                <total/>
                        </direction>
                        <direction       name="Belgium-Netherlands"
                                         fromsubnets="10.129.0.0/16"
                                         tosubnets="10.240.0.0/16">
                                <application name="web">80/tcp,8080/tcp</application>
                                <application name="secureweb">443/tcp</application>
                                <application name="mailreading">
                                        110/tcp,143/tcp
                                </application>
                                <application name="windows">
                                        137-139/tcp,137-139/udp,42/tcp,135/tcp,445/tcp
                                </application>
                                <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
                                <protocols>tcp,udp,icmp</protocols>
                                <ftp/>
                                <multicast/>
                                <tos/>
                                <total/>
                        </direction>
                        <application name="web">80/tcp,8080/tcp</application>
                        <application name="secureweb">443/tcp</application>
                        <application name="mailreading">110/tcp,143/tcp</application>
                        <application name="windows">
                                137-139/tcp,137-139/udp,42/tcp,135/tcp,445/tcp
                        </application>
                        <application name="dns">53/udp,53/tcp</application>
                        <application name="snmp">161-162/udp</application>
                        <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
                        <protocols>tcp,udp,icmp</protocols>
                        <ftp/>
                        <multicast/>
                        <tos/>
                        <total/>
                </subnet>
                <subnet          subnets="10.130.0.0/16"
                                 name="UnitedKingdom"
                                 write="yes">
                        <application name="web">80/tcp,8080/tcp</application>
                        <application name="secureweb">443/tcp</application>
                        <application name="mailreading">110/tcp,143/tcp</application>
                        <application name="windows">
                                137-139/tcp,137-139/udp,42/tcp,135/tcp,445/tcp
                        </application>
                        <application name="dns">53/udp,53/tcp</application>
                        <application name="snmp">161-162/udp</application>
                        <services>20-23/tcp,25/tcp,102/tcp,119/tcp</services>
                        <protocols>tcp,udp,icmp</protocols>
                        <multicast/>
                        <tos/>
                        <total/>
                </subnet>
                <subnet          subnets="10.131.0.0/16"
                                 name="France"
                                 write="yes">
                        <application name="web">80/tcp,8080/tcp</application>
                        <application name="secureweb">443/tcp</application>
                        <application name="mailreading">110/tcp,143/tcp</application>
                        <application name="windows">
                                137-139/tcp,137-139/udp,42/tcp,135/tcp,445/tcp
                        </application>
                        <application name="dns">53/udp,53/tcp</application>
                        <application name="snmp">161-162/udp</application>
                        <services>20-23/tcp,25/tcp,102/tcp,119/tcp</services>
                        <protocols>tcp,udp,icmp</protocols>
                        <multicast/>
                        <tos/>
                        <total/>
                </subnet>
                <total_subnet/>
        </subnets>
        <networks>
                <network name="netwerk1">
                        <routers>routers1,routers2</routers>
                </network>
        </networks>
        <outputdir>/var/flows/reports/rrds</outputdir>
</config>
```

Appendix C: JKFlow.xml example (version 3.4)

JKFlow_example_verybasic.xml

```
<config>
        <all localsubnets="10.0.0.0/8">
                <application name="web">80/tcp,8080/tcp</application>
                <application name="secureweb">443/tcp</application>
                <application name="mailreading">110/tcp,143/tcp</application>
                <application name="windows">137-139/tcp,137-139/udp</application>
                <application name="dns">53/udp,53/tcp</application>
                <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
                <protocols>tcp,udp,icmp</protocols>
                <otherservices/>
                <otherprotocols/>
                <ftp/>
                <tos/>
                <total/>
        </all>
        <rrddir>/var/flows/reports/rrds</rrddir>
        <scoredir>/var/flows/score</scoredir>
        <sampletime>300</sampletime>
</config>
```

JKFlow_example_basicsites_nosets.xml

```
<config>
  <sites>
    <site name="Belgium" subnets="10.10.0.0/16"/>
    <site name="Holland" subnets="10.20.0.0/16"/>
    <site name="England" subnets="10.30.0.0/16"/>
    <site name="Internet" subnets="0.0.0.0/0"/>
  </sites>
  <directions>
    <direction name="Belgium-Internet" from="Belgium" to="Internet" noto="Belgium">
        <application name="web">80/tcp,8080/tcp</application>
        <application name="secureweb">443/tcp</application>
        <application name="mailreading">110/tcp,143/tcp</application>
        <application name="windows">137-139/tcp,137-139/udp</application>
        <application name="dns">53/udp,53/tcp</application>
        <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
        <otherservices/>
        <ftp/>
        <protocols>tcp,udp,icmp</protocols>
        <multicast/>
        <otherprotocols/>
        <tos/>
        <total/>
    </direction>
    <direction name="Holland-Internet" from="Holland" to="Internet" noto="Holland">
        <application name="web">80/tcp,8080/tcp</application>
        <application name="secureweb">443/tcp</application>
        <application name="mailreading">110/tcp,143/tcp</application>
        <application name="windows">137-139/tcp,137-139/udp</application>
        <application name="dns">53/udp,53/tcp</application>
        <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
        <otherservices/>
        <ftp/>
        <protocols>tcp,udp,icmp</protocols>
        <multicast/>
        <otherprotocols/>
        <tos/>
        <total/>
    </direction>
    <direction name="England-Internet" from="England" to="Internet" noto="England">
        <application name="web">80/tcp,8080/tcp</application>
        <application name="secureweb">443/tcp</application>
        <application name="mailreading">110/tcp,143/tcp</application>
        <application name="windows">137-139/tcp,137-139/udp</application>
        <application name="dns">53/udp,53/tcp</application>
        <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
        <otherservices/>
        <ftp/>
        <protocols>tcp,udp,icmp</protocols>
        <multicast/>
        <otherprotocols/>
        <tos/>
        <total/>
    </direction>
  </directions>
  <rrddir>/var/flows/reports/rrds</rrddir>
  <scoredir>/var/flows/score</scoredir>
```

```
  <sampletime>300</sampletime>
</config>
```

JKFlow_example_basicsites_withsets.xml

```
<config>
  <definesets>
    <defineset name="Common Services">
         <application name="web">80/tcp,8080/tcp</application>
         <application name="secureweb">443/tcp</application>
         <application name="mailreading">110/tcp,143/tcp</application>
         <application name="windows">137-139/tcp,137-139/udp</application>
         <application name="dns">53/udp,53/tcp</application>
         <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
         <otherservices/>
         <ftp/>
         <protocols>tcp,udp,icmp</protocols>
         <multicast/>
         <otherprotocols/>
         <tos/>
         <total/>
    </defineset>
  </definesets>
  <sites>
         <site name="Belgium" subnets="10.10.0.0/16"/>
         <site name="Holland" subnets="10.20.0.0/16"/>
         <site name="England" subnets="10.30.0.0/16"/>
         <site name="Internet" subnets="0.0.0.0/0"/>
  </sites>
  <directions>
    <direction name="Belgium-Internet" from="Belgium" to="Internet" noto="Belgium">
         <set name="Common Services"/>
    </direction>
    <direction name="Holland-Internet" from="Holland" to="Internet" noto="Holland">
         <set name="Common Services"/>
    </direction>
    <direction name="England-Internet" from="England" to="Internet" noto="England">
         <set name="Common Services"/>
    </direction>
  </directions>
  <rrddir>/var/flows/reports/rrds</rrddir>
  <scoredir>/var/flows/score</scoredir>
  <sampletime>300</sampletime>
</config>
```

JKFlow_example_as.xml

```
<config>
  <definesets>
    <defineset name="Common Services">
         <application name="web">80/tcp,8080/tcp</application>
         <application name="secureweb">443/tcp</application>
         <application name="mailreading">110/tcp,143/tcp</application>
         <application name="windows">137-139/tcp,137-139/udp</application>
         <application name="dns">53/udp,53/tcp</application>
         <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
         <otherservices/>
         <ftp/>
         <total/>
    </defineset>
    <defineset name="Extra Services">
         <protocols>tcp,udp,icmp</protocols>
         <multicast/>
         <otherprotocols/>
         <tos/>
    </defineset>
    <defineset name="Scoreboarding">
         <scoreboard>
                 <report count="12" hostsbase="agghostshour" portsbase="aggportshour"/>
                 <report count="72" hostsbase="agghosts6hour" portsbase="aggports6hour"/>
         </scoreboard>
    </defineset>
  </definesets>
  <sites>
         <site name="Belgium" subnets="10.10.0.0/16"/>
         <site name="Holland" subnets="10.20.0.0/16"/>
         <site name="England" subnets="10.30.0.0/16"/>
         <site name="Internet" subnets="0.0.0.0/0" nosubnets="10.0.0.0/8"/>
  </sites>
  <directions>
    <direction name="AS404142-AS202122" fromas="40,41,42" toas="20,21,22">
         <set name="Common Services"/>
         <set name="Extra Services"/>
         <set name="Scoreboarding"/>
    </direction>
```

```
    <direction name="Belgium-Holland-AS4041-AS2021" from="Belgium" to="Holland"
fromas="40,41" toas="20,21">
        <set name="Common Services"/>
        <set name="Scoreboarding"/>
    </direction>
  </directions>
  <rrddir>/var/flows/reports/rrds</rrddir>
  <scoredir>/var/flows/score</scoredir>
  <sampletime>300</sampletime>
</config>
```

JKFlow_example_routers.xml

```
<config>
  <definesets>
    <defineset name="Common Services">
        <application name="web">80/tcp,8080/tcp</application>
        <application name="secureweb">443/tcp</application>
        <application name="mailreading">110/tcp,143/tcp</application>
        <application name="windows">137-139/tcp,137-139/udp</application>
        <application name="dns">53/udp,53/tcp</application>
        <application name="netflow">2055/tcp,2055/udp</application>
        <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
        <protocols>tcp,udp,icmp</protocols>
        <otherprotocols/>
        <otherservices/>
        <ftp/>
        <multicast/>
        <tos/>
        <total/>
    </defineset>
  </definesets>
  <routergroups>
    <routergroup name="ROUTERS1">
        <router exporter="10.1.1.1" interface="10"/>
        <router exporter="10.1.1.1" interface="11"/>
        <router exporter="10.1.1.2" interface="10"/>
        <router exporter="10.1.1.2" interface="11"/>
    </routergroup>
    <routergroup name="ROUTERS1-LINE1">
        <router exporter="10.1.1.1" interface="10"/>
        <router exporter="10.1.1.2" interface="10"/>
    </routergroup>
    <routergroup name="ROUTERS1-LINE2">
        <router exporter="10.1.1.1" interface="11"/>
        <router exporter="10.1.1.2" interface="11"/>
    </routergroup>
    <routergroup name="ROUTERS2">
        <router exporter="10.2.2.2" interface="9"/>
        <router exporter="10.2.2.3" interface="9"/>
        <router exporter="10.2.2.4" interface="9"/>
        <router exporter="10.2.2.5" interface="9"/>
        <router exporter="10.2.2.2" interface="10"/>
        <router exporter="10.2.2.3" interface="10"/>
        <router exporter="10.2.2.4" interface="10"/>
        <router exporter="10.2.2.5" interface="10"/>
        <router exporter="10.2.2.2" interface="13"/>
        <router exporter="10.2.2.3" interface="13"/>
        <router exporter="10.2.2.4" interface="13"/>
        <router exporter="10.2.2.5" interface="13"/>
    </routergroup>
    <routergroup name="ROUTERS2-LINE1">
        <router exporter="10.2.2.2" interface="9"/>
        <router exporter="10.2.2.3" interface="9"/>
        <router exporter="10.2.2.4" interface="9"/>
        <router exporter="10.2.2.5" interface="9"/>
    </routergroup>
    <routergroup name="ROUTERS2-LINE2">
        <router exporter="10.2.2.2" interface="10"/>
        <router exporter="10.2.2.3" interface="10"/>
        <router exporter="10.2.2.4" interface="10"/>
        <router exporter="10.2.2.5" interface="10"/>
    </routergroup>
    <routergroup name="ROUTERS2-LINE3">
        <router exporter="10.2.2.2" interface="13"/>
        <router exporter="10.2.2.3" interface="13"/>
        <router exporter="10.2.2.4" interface="13"/>
        <router exporter="10.2.2.5" interface="13"/>
    </routergroup>
    </routergroups>
  <directions>
    <direction name="ROUTERS1" routergroup="ROUTERS1">
        <set name="Common Services"/>
        <scoreboard>
                <report count="12" hostsbase="agghostshour" portsbase="aggportshour"/>
```

```
                <report count="72" hostsbase="agghosts6hour" portsbase="aggports6hour"/>
            </scoreboard>
            <scoreboardother>
                <report count="12" hostsbase="agghostshour" portsbase="aggportshour"/>
                <report count="72" hostsbase="agghosts6hour" portsbase="aggports6hour"/>
            </scoreboardother>
        </direction>
        <direction name="ROUTERS1-LINE1" routergroup="ROUTERS1-LINE1">
            <set name="Common Services"/>
            <scoreboard>
                <report count="12" hostsbase="agghostshour" portsbase="aggportshour"/>
                <report count="72" hostsbase="agghosts6hour" portsbase="aggports6hour"/>
            </scoreboard>
            <scoreboardother>
                <report count="12" hostsbase="agghostshour" portsbase="aggportshour"/>
                <report count="72" hostsbase="agghosts6hour" portsbase="aggports6hour"/>
            </scoreboardother>
        </direction>
        <direction name="ROUTERS1-LINE2" routergroup="ROUTERS1-LINE2">
            <set name="Common Services"/>
        </direction>
        <direction name="ROUTERS2" routergroup="ROUTERS2">
            <set name="Common Services"/>
        </direction>
        <direction name="ROUTERS2-LINE1" routergroup="ROUTERS2-LINE1">
            <set name="Common Services"/>
        </direction>
        <direction name="ROUTERS2-LINE2" routergroup="ROUTERS2-LINE2">
            <set name="Common Services"/>
             <scoreboard>
                <report count="12" hostsbase="agghostshour" portsbase="aggportshour"/>
                <report count="72" hostsbase="agghosts6hour" portsbase="aggports6hour"/>
            </scoreboard>
        </direction>
        <direction name="ROUTERS2-LINE3" routergroup="ROUTERS2-LINE3">
            <set name="Common Services"/>
        </direction>
        <direction name="other">
            <set name="Common Services"/>
            <scoreboard hosts="1" ports="1" >
                <report count="12" hostsbase="agghostshour" portsbase="aggportshour"/>
                <report count="72" hostsbase="agghosts6hour" portsbase="aggports6hour"/>
            </scoreboard>
            <scoreboardother>
                <report count="12" hostsbase="agghostshour" portsbase="aggportshour"/>
                <report count="72" hostsbase="agghosts6hour" portsbase="aggports6hour"/>
            </scoreboardother>
        </direction>
    </directions>
    <rrddir>/var/flows/report/rrds</rrddir>
    <scoredir>/var/flows/score</scoredir>
    <sampletime>300</sampletime>
</config>
```

JKFlow_example_sites.xml

```
<config>
  <definesets>
    <defineset name="Common Services">
      <application name="web">80/tcp,8080/tcp</application>
      <application name="secureweb">443/tcp</application>
      <application name="mailreading">110/tcp,143/tcp</application>
      <application name="windows">137-139/tcp,137-139/udp</application>
      <application name="dns">53/udp,53/tcp</application>
      <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
      <ftp/>
      <total/>
    </defineset>
    <defineset name="Extra Services">
      <protocols>tcp,udp,icmp</protocols>
      <multicast/>
      <tos/>
    </defineset>
  </definesets>
  <sites>
    <site name="Belgium" subnets="10.10.0.0/16"/>
    <site name="Holland" subnets="10.20.0.0/16"/>
    <site name="England" subnets="10.30.0.0/16"/>
    <site name="Germany" subnets="10.40.0.0/16"/>
    <site name="France" subnets="10.50.0.0/16"/>
    <site name="Internet" subnets="0.0.0.0/0"/>
  </sites>
  <routergroups>
    <routergroup name="routers1">
      <router exporter="10.1.2.2" interface="3"/>
      <router exporter="10.1.2.3" interface="4"/>
```

```
  </routergroup>
  <routergroup name="routers2">
    <router exporter="10.1.2.3" localsubnets="10.0.0.0/8"/>
  </routergroup>
  <routergroup name="routers3">
    <router exporter="10.1.2.3" interface="1"/>
  </routergroup>
  <routergroup name="routers4">
    <router exporter="10.1.2.3" interface="2"/>
  </routergroup>
</routergroups>
<all localsubnets="10.0.0.0/8" samplerate="1">
  <set name="Common Services"/>
  <set name="Extra Services"/>
</all>
<directions>
  <direction name="Belgium-Internet" from="Belgium" to="Internet" noto="Belgium">
    <set name="Common Services"/>
    <set name="Extra Services"/>
    <scoreboard hosts="1" ports="1"/>
  </direction>
  <direction name="Holland-Internet" from="Holland" to="Internet" noto="Holland">
    <set name="Common Services"/>
  </direction>
  <direction name="Holland-Internet interface 1" from="Holland" to="Internet"
  noto="Holland" routergroup="routers3">
    <set name="Common Services"/>
  </direction>
  <direction name="Holland-Internet interface 2" from="Holland" to="Internet"
  noto="Holland" routergroup="routers4">
    <set name="Common Services"/>
  </direction>
  <direction name="Holland-Internet subnet" from="Holland" to="Internet"
  noto="Holland" routergroup="routers2">
    <set name="Common Services"/>
  </direction>
  <direction name="Engeland-Germany" from="England" to="Germany">
    <set name="Common Services"/>
  </direction>
  <direction name="Engeland-France" from="England" to="France">
    <set name="Common Services"/>
  </direction>
  <direction name="Engeland-Internet" from="England" to="Internet" noto="England">
    <set name="Common Services"/>
  </direction>
  <direction name="Duitsland-Internet" from="Germany" to="Internet" noto="Germany">
    <set name="Common Services"/>
  </direction>
  <direction name="France-Internet" from="France" to="Internet" noto="France">
    <set name="Common Services"/>
    <set name="Extra Services"/>
    <scoreboard hosts="1" ports="1"/>
  </direction>
  <direction name="France-Germany" from="France" to="Germany">
    <set name="Common Services"/>
    <set name="Extra Services"/>
    <scoreboard hosts="1" ports="1"/>
  </direction>
  <direction name="Holland-Germany" from="Holland" to="Germany">
    <set name="Common Services"/>
    <set name="Extra Services"/>
  </direction>
  <direction name="Holland-France" from="Holland" to="France">
    <set name="Common Services"/>
    <set name="Extra Services"/>
  </direction>
  <direction name="Belgium-Germany" from="Belgium" to="Germany">
    <set name="Common Services"/>
    <set name="Extra Services"/>
  </direction>
  <direction name="NotFromHolland-Belgium" nofrom="Holland" to="Belgium">
    <application name="web">80/tcp,8080/tcp</application>
    <application name="secureweb">443/tcp</application>
    <application name="mailreading">110/tcp,143/tcp</application>
    <application name="windows">137-139/tcp,137-139/udp</application>
    <application name="dns">53/udp,53/tcp</application>
    <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
    <protocols>tcp,udp,icmp</protocols>
    <multicast/>
    <tos/>
    <ftp/>
    <total/>
    <scoreboard hosts="1"/>
  </direction>
  <direction name="Routers1" routergroup="routers1">
    <set name="Common Services"/>
    <set name="Extra Services"/>
```

```
      <scoreboard ports="1"/>
    </direction>
    <direction name="Routers2" routergroup="routers2">
      <set name="Common Services"/>
      <set name="Extra Services"/>
      <scoreboard hosts="1" ports="1"/>
    </direction>
  </directions>
  <rrddir>/var/flows/reports/rrds</rrddir>
  <scoredir>/var/flows/score</scoredir>
</config>
```

JKFlow_example_scoreboard_tuples.xml (JKFlow v3.5)

```
<config>
  <definesets>
    <defineset name="Common Services">
      <application name="web">80/tcp,8080/tcp</application>
      <application name="secureweb">443/tcp</application>
      <application name="mailreading">110/tcp,143/tcp</application>
      <application name="windows">137-139/tcp,137-139/udp</application>
      <application name="dns">53/udp,53/tcp</application>
      <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
      <ftp/>
      <total/>
    </defineset>
    <defineset name="Extra Services">
      <protocols>tcp,udp,icmp</protocols>
      <multicast/>
      <dscp/>
    </defineset>
    <defineset name=”Scoreboard”>
        <scoreboard>
            <tuples>
                 <tuple>srcip,dstip</tuple>
                 <tuple>srcip,dstport</tuple>
            </tuples>
            <report count="12" base="agghour"/>
            <report count="72" base="agg6hour"/>
        </scoreboard>
    </defineset>
    <defineset name=”Scoreboard-Other”>
        </scoreboardother>
            <tuples>
                 <tuple>srcip,dstip</tuple>
                 <tuple>srcip,dstport</tuple>
            </tuples>
            <report count="12" base="agghour"/>
            <report count="72" base="agg6hour"/>
        </scoreboardother>
    </defineset>
  </definesets>
  <sites>
    <site name="Belgium" subnets="10.10.0.0/16"/>
    <site name="Holland" subnets="10.20.0.0/16"/>
    <site name="England" subnets="10.30.0.0/16"/>
    <site name="Germany" subnets="10.40.0.0/16"/>
    <site name="France" subnets="10.50.0.0/16"/>
    <site name="Internet" subnets="0.0.0.0/0"/>
  </sites>
  <all localsubnets="10.0.0.0/8" samplerate="1">
    <set name="Common Services"/>
    <set name="Extra Services"/>
    <set name="Scoreboard”/>
    <set name="Scoreboard-Other”/>
  </all>
  <directions>
    <direction name="Belgium-Internet" from="Belgium" to="Internet" noto="Belgium">
      <set name="Common Services"/>
      <set name="Extra Services"/>
      <set name="Scoreboard”/>
    </direction>
    <direction name="Holland-Internet" from="Holland" to="Internet" noto="Holland">
      <set name="Common Services"/>
      <set name="Scoreboard”/>
    </direction>
    <direction name="Engeland-Germany" from="England" to="Germany">
      <set name="Common Services"/>
      <set name="Scoreboard”/>
    </direction>
    <direction name="Engeland-France" from="England" to="France">
      <set name="Common Services"/>
    </direction>
    <direction name="Engeland-Internet" from="England" to="Internet" noto="England">
      <set name="Common Services"/>
    </direction>
```

```
    <direction name="Duitsland-Internet" from="Germany" to="Internet" noto="Germany">
      <set name="Common Services"/>
    </direction>
    <direction name="France-Internet" from="France" to="Internet" noto="France">
      <set name="Common Services"/>
      <set name="Extra Services"/>
    </direction>
    <direction name="France-Germany" from="France" to="Germany">
      <set name="Common Services"/>
      <set name="Extra Services"/>
      <set name="Scoreboard”/>
    </direction>
    <direction name="Holland-Germany" from="Holland" to="Germany">
      <set name="Common Services"/>
      <set name="Extra Services"/>
      <set name="Scoreboard”/>
    </direction>
    <direction name="Holland-France" from="Holland" to="France">
      <set name="Common Services"/>
      <set name="Extra Services"/>
      <set name="Scoreboard”/>
    </direction>
    <direction name="Belgium-Germany" from="Belgium" to="Germany">
      <set name="Common Services"/>
      <set name="Extra Services"/>
      <set name="Scoreboard”/>
      <set name="Scoreboard-Other”/>
    </direction>
    <direction name="NotFromHolland-Belgium" nofrom="Holland" to="Belgium">
      <application name="web">80/tcp,8080/tcp</application>
      <application name="secureweb">443/tcp</application>
      <application name="mailreading">110/tcp,143/tcp</application>
      <application name="windows">137-139/tcp,137-139/udp</application>
      <application name="dns">53/udp,53/tcp</application>
      <services>22-23/tcp,25/tcp,102/tcp,119/tcp</services>
      <protocols>tcp,udp,icmp</protocols>
      <multicast/>
      <tos/>
      <ftp/>
      <total/>
      <scoreboard>
          <tuples>
               <tuple>srcip,dstip</tuple>
               <tuple>srcip,dstport</tuple>
               <tuple>srcip,srcport,dstip,dstport</tuple>
          </tuples>
          <report count="12" base="agghour"/>
          <report count="72" base="agg6hour"/>
      </scoreboard>
    </direction>
    <direction name="Routers1" routergroup="routers1">
      <set name="Common Services"/>
      <set name="Extra Services"/>
        <scoreboard>
            <tuples>
                 <tuple>srcip,dstip</tuple>
                 <tuple>srcip,dstport</tuple>
                 <tuple>srcip,srcport,dstip,dstport</tuple>
            </tuples>
            <report count="12" base="agghour"/>
            <report count="72" base="agg6hour"/>
        </scoreboard>
    </direction>
    <direction name="Routers2" routergroup="routers2">
      <set name="Common Services"/>
      <set name="Extra Services"/>
        <scoreboard>
            <tuples>
                 <tuple>srcip,dstip</tuple>
                 <tuple>srcip,dstport</tuple>
                 <tuple>srcip,srcport,dstip,dstport</tuple>
            </tuples>
            <report count="12" base="agghour"/>
            <report count="72" base="agg6hour"/>
        </scoreboard>
    </direction>
  </directions>
  <rrddir>/var/flows/reports/rrds</rrddir>
  <scoredir>/var/flows/score</scoredir>
</config>
```

Appendix D: Resources

Books

"The Tao of Network Security Monitoring, Beyond Intrusion Detection", Richard Bejtlich, Addison Wesley
"Programming Perl", Larry Wall, Tom Christiansen & Jon Orwant, O'Reilly
"Advanced Perl Programming", Sriram Srinivasan, O'Reilly

URL's

JKFlow:
http://jkflow.sourceforge.net
http://users.telenet.be/jurgen.kobierczynski/jkflow/JKFlow.html
http://users.telenet.be/jurgen.kobierczynski/jkflow/eindwerk.pdf

NetFlowGuide:
http://www.dynamicnetworks.us/netflow/
http://netflowguide.com/?page=guide

Software List:
http://www.switch.ch/tf-tant/floma/software.html