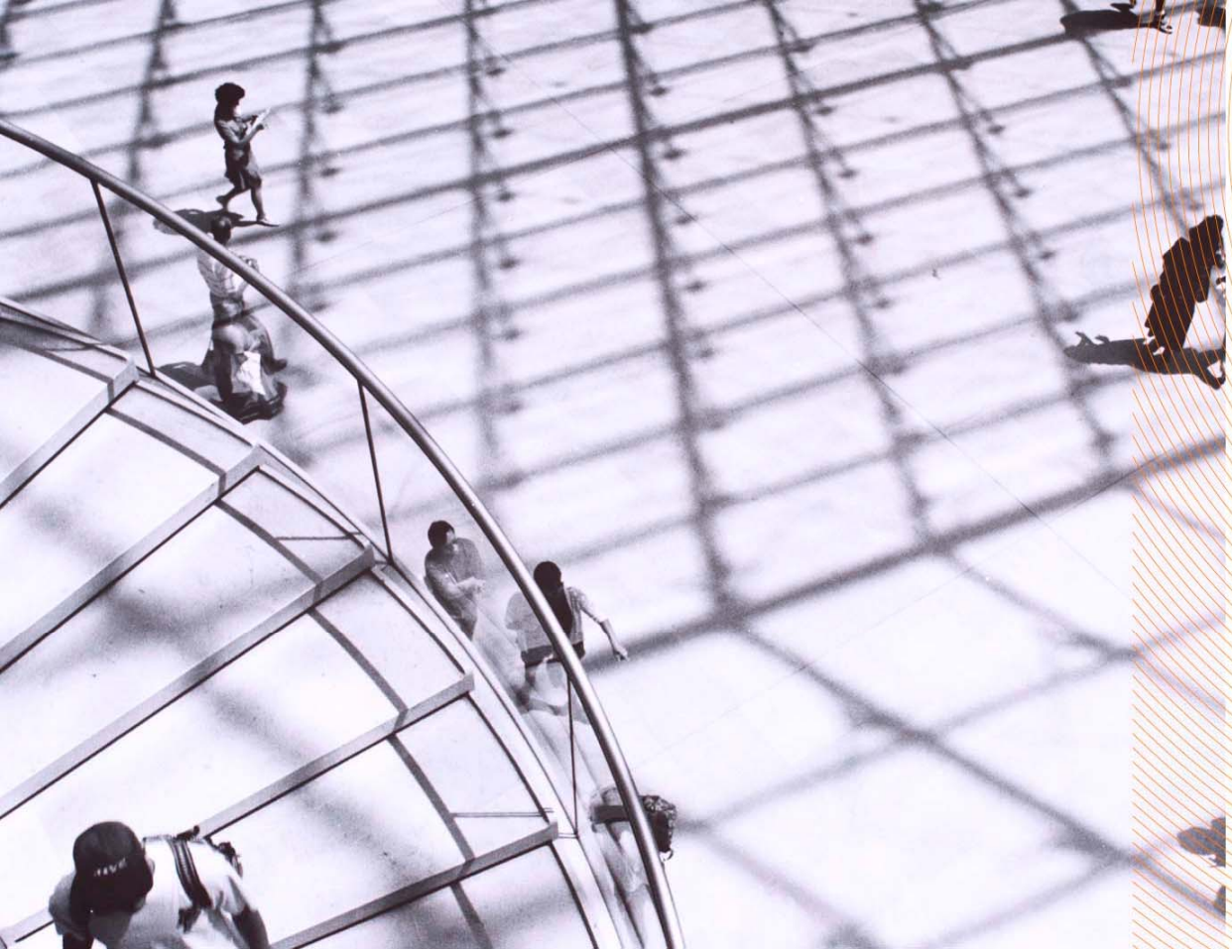


Automatic anomaly detection using NfSen

Wim Biemolt, SURFnet

Werner Schram, SURFnet

Automatic anomaly detection using NfSen

- SURFnet and netflow anomaly detection
  - NERD
  - NfSen
  - PeakFlow SP
- Currently used detection methods
  - DDos
  - Botnet
  - Holt-Winters aberrant behavior

SURFnet and netflow anomaly detection

- NERD v1
  - Developed by TNO
  - Based on cflowd
  - cflowd is no longer supported
- NERD v2
  - Initially developed by TNO
  - Has serious performance problems
  - NfSen can do the same but without the performance problems

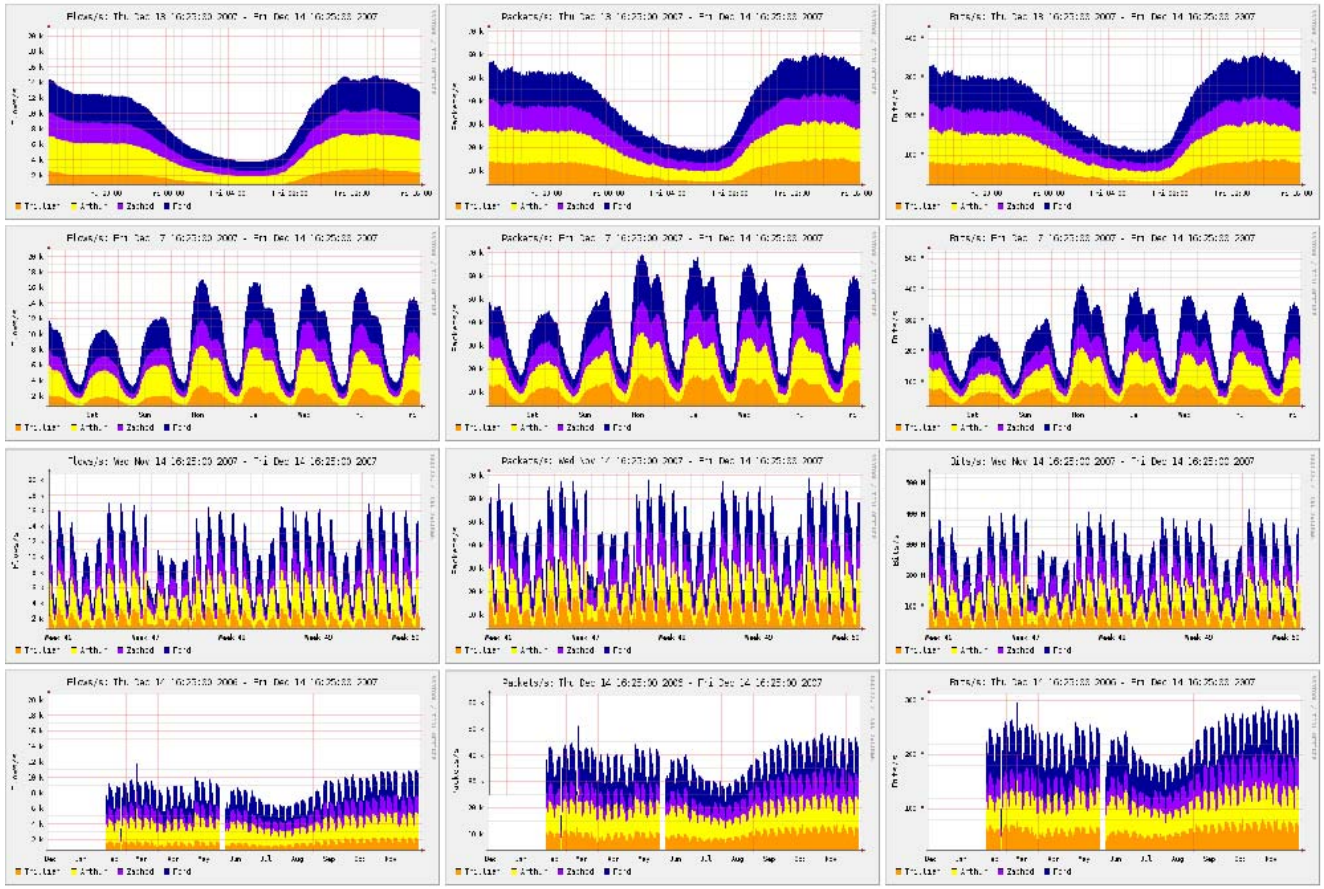
NfSen

- Netflow Sensor (NfSen) is a
  - network statistics tool
  - Developed by Peter Haag
  - Currently in active development
  - Alert plug-in system
  - Generic plug-in system
  - Some plug-ins already available

NfSen

Home | Graphs | Details | Alerts | Stats | Plugins | live | Bookmark URL | Profile: | live ▼

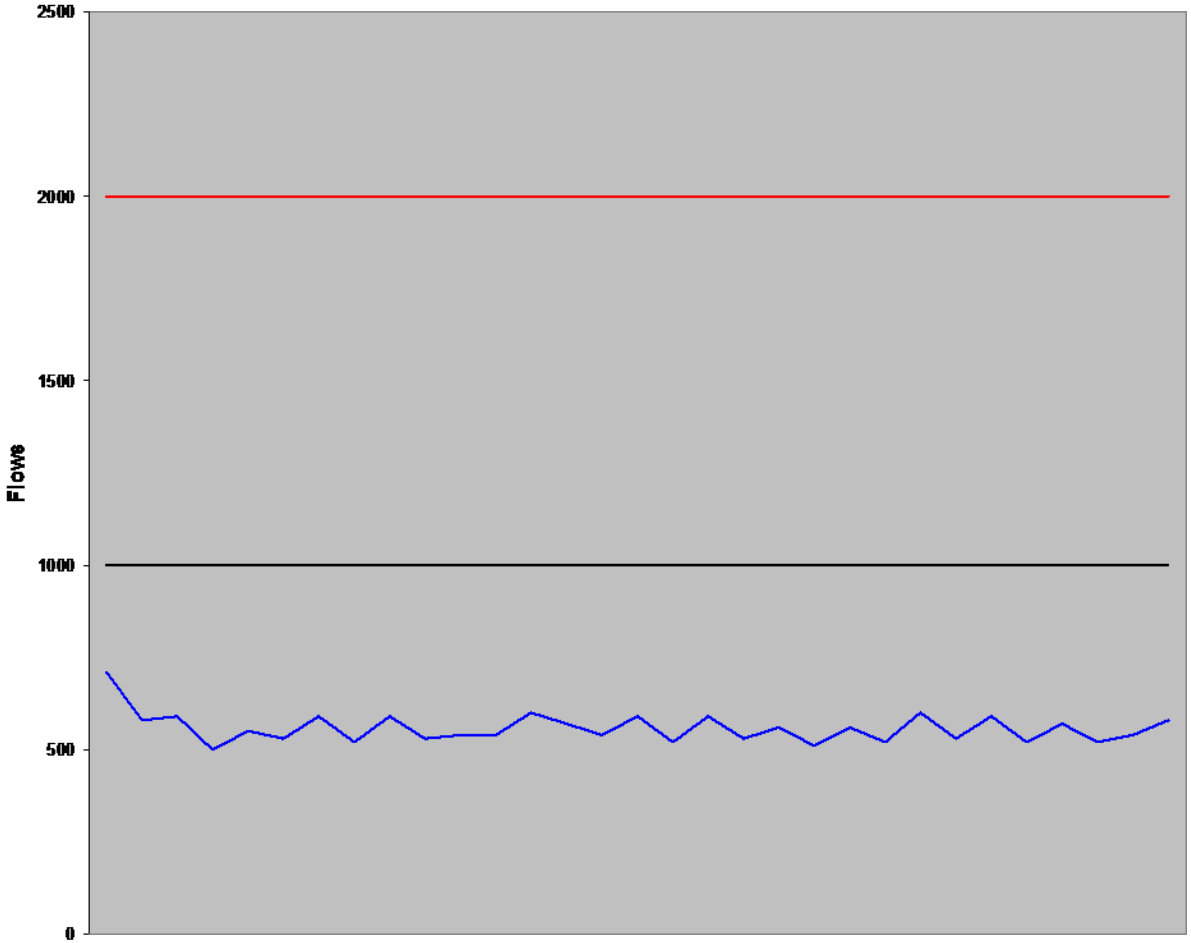
Overview Profile: live, Group: (nogroup)



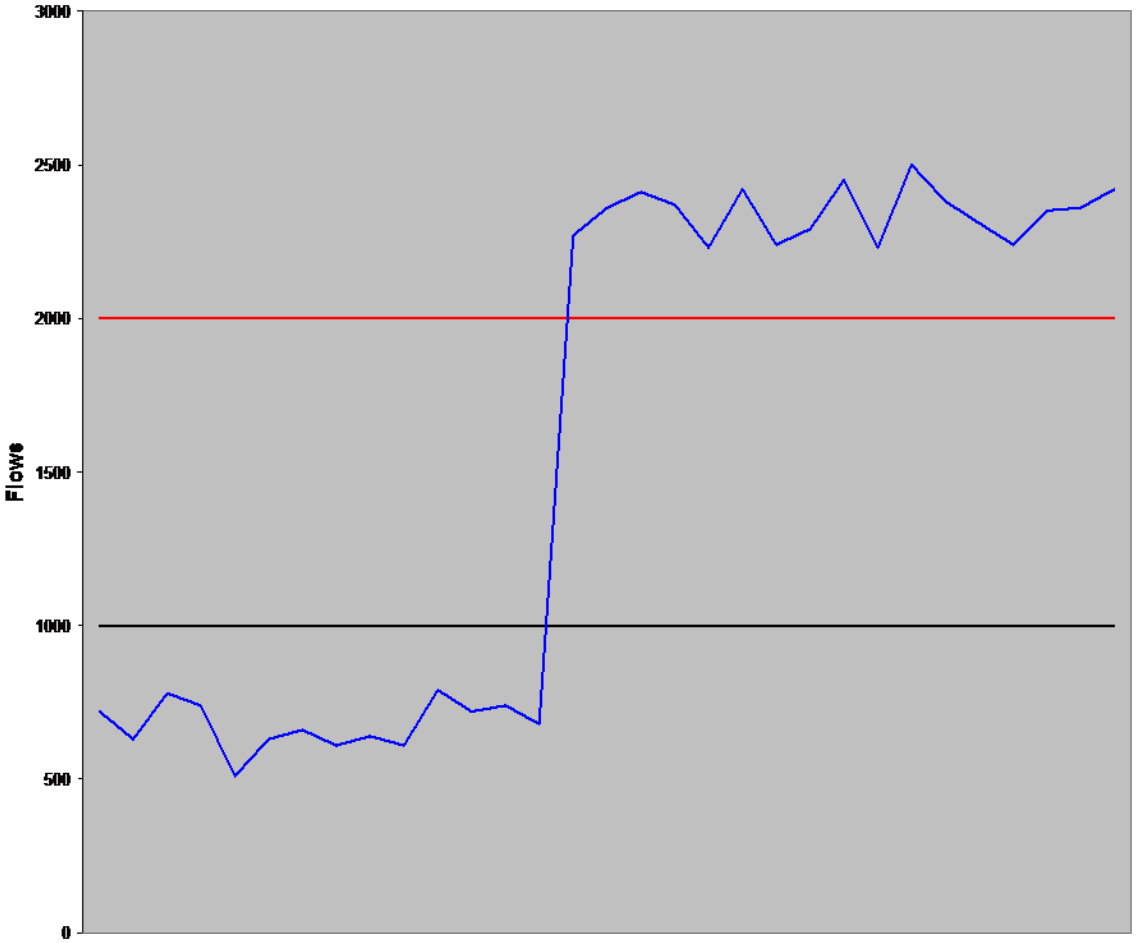
DDos detection

- Simple flow analysis
  - based on NERD v1 DDos detection
  - using a low threshold and a high threshold
  - Rules for traffic between those thresholds
  - Custom thresholds for high load services

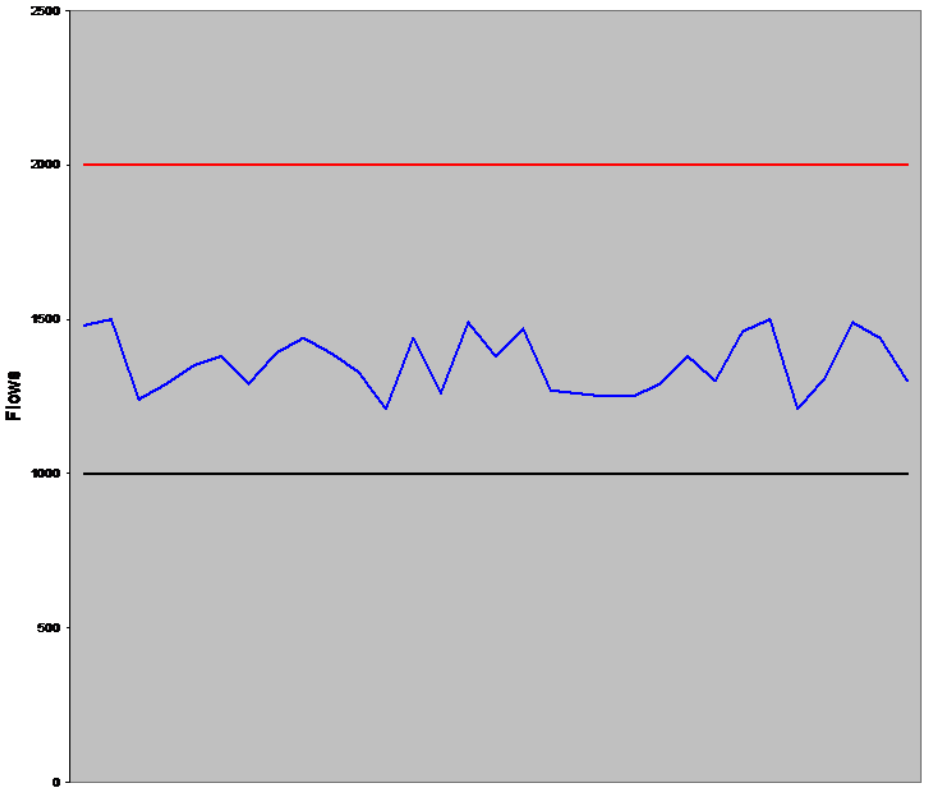
Expected traffic

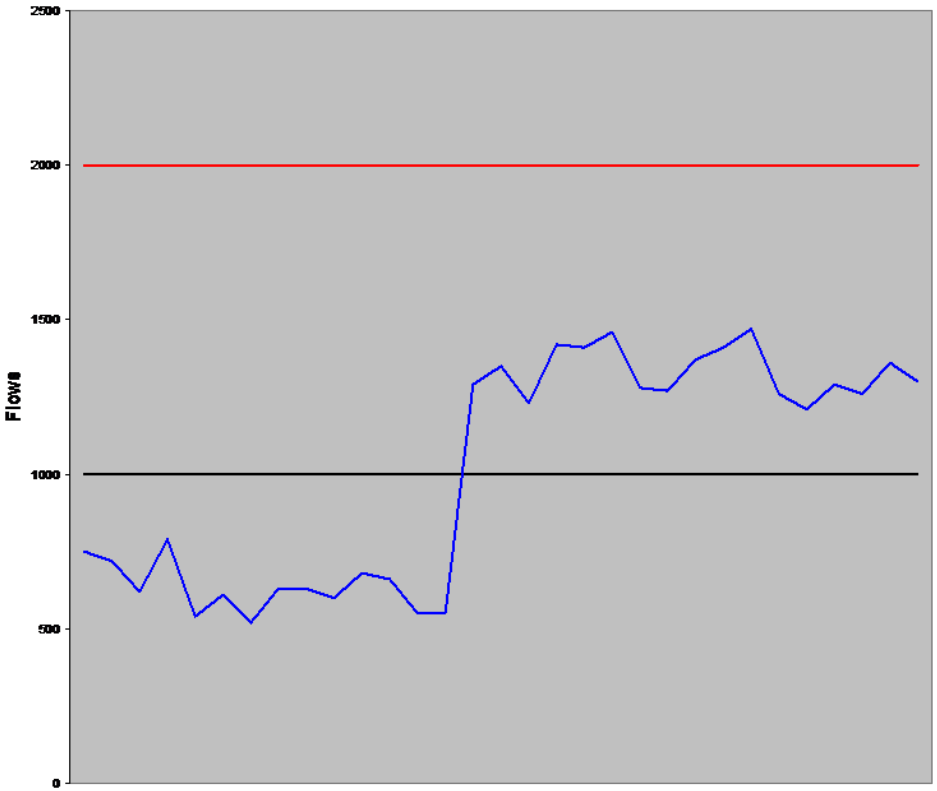


Definitively Conspicuous Traffic

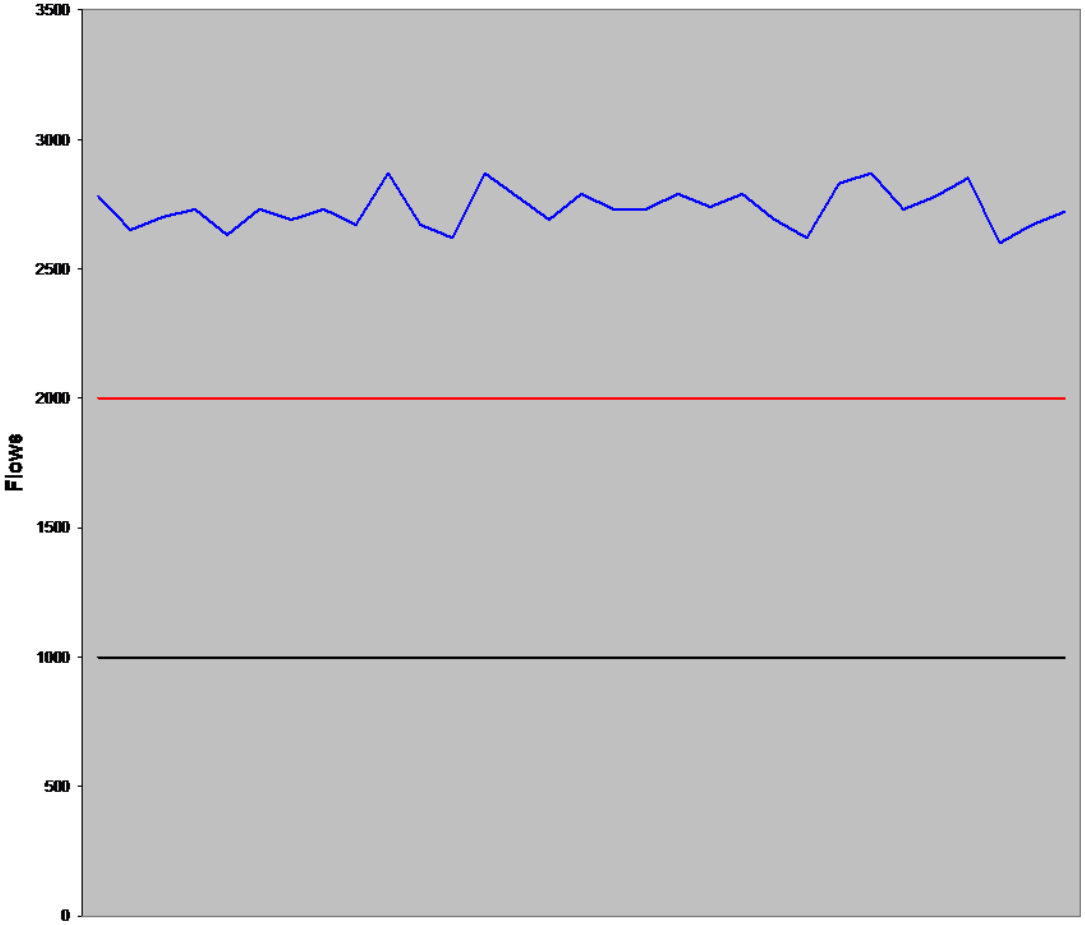


Border cases

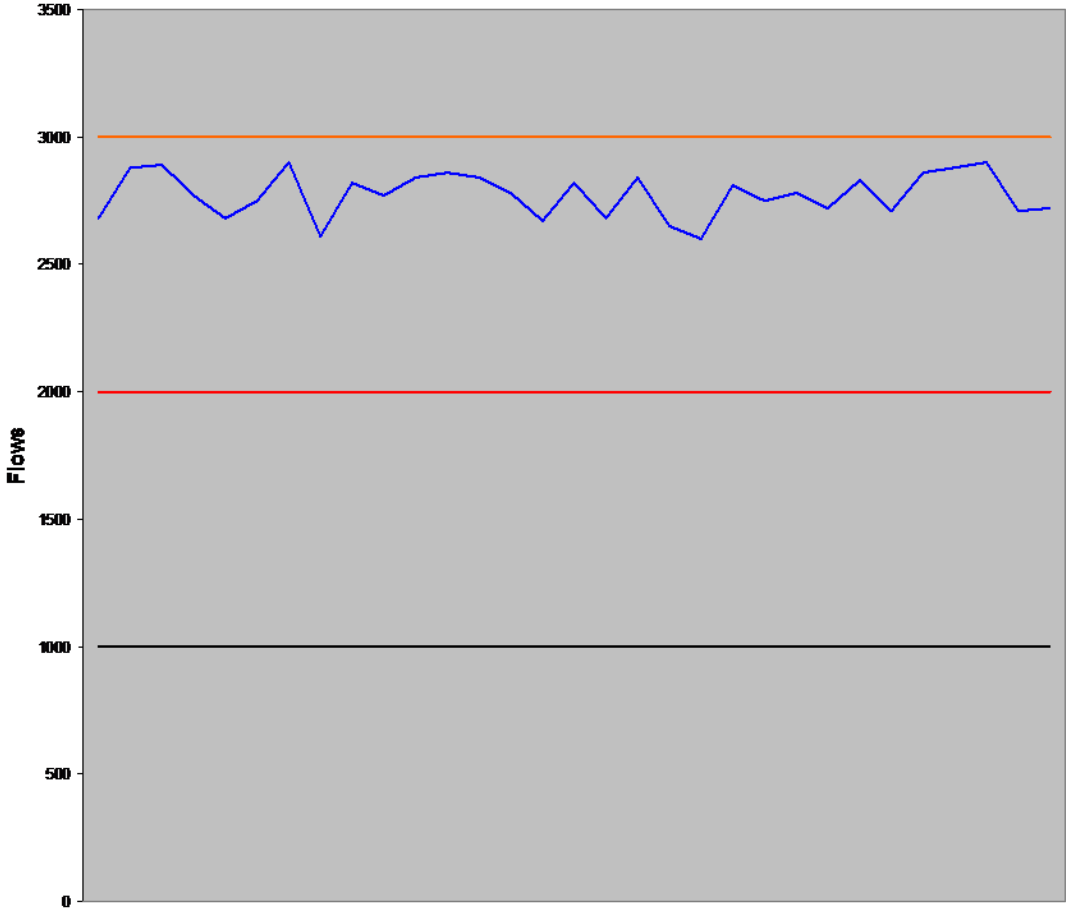




High load servers



Custom thresholds



DDos interface: report

Home | Graphs | Details | Alerts | Stats | Plugins | live | Bookmark URL | Profile: live ▼

alarm | Events

report | setup | thresholds | botnets

number of alarms to show: 10 (0 for all)

from 7 days ago

up to 0 days ago

alarms: ddos

Ok

The ddos alarms between **2007-12-07** and **2007-12-15**

ID	Destination	Flows per 5 minutes	Average packets/flow	Average bytes/flow	Starttime	Stoptime	Active	
#50598	[redacted]	7772	5054	4	2007-12-14 08:55:00	2007-12-14 16:32:50	1	Delete
#50596	[redacted]	10620	3859	4	2007-12-14 08:39:54	2007-12-14 16:32:50	1	Delete
#50594	[redacted]	9510	3147	3	2007-12-14 08:25:01	2007-12-14 16:32:50	1	Delete
#50593	[redacted]	12951	129	2	2007-12-14 08:24:58	2007-12-14 16:32:50	1	Delete
#50490	[redacted]	9517	73	1	2007-12-13 06:13:41	2007-12-14 16:32:50	1	Delete
#49820	[redacted]	281618	163	1	2007-12-04 14:47:47	2007-12-14 16:32:50	1	Delete
#49191	[redacted]	327975	125	1	2007-11-27 13:19:14	2007-12-14 16:32:50	1	Delete
#49074	[redacted]	22047	171	2	2007-11-26 13:32:20	2007-12-14 16:32:50	1	Delete
#50656	[redacted]	5222	2550	3	2007-12-14 16:20:07	2007-12-14 16:29:56	1	Delete
#50635	[redacted]	6031	1155	7	2007-12-14 11:44:53	2007-12-14 16:22:51	1	Delete

DDos interface: Details

Home | Graphs | Details | Alerts | Stats | Plugins | live | Bookmark URL | Profile: live ▼

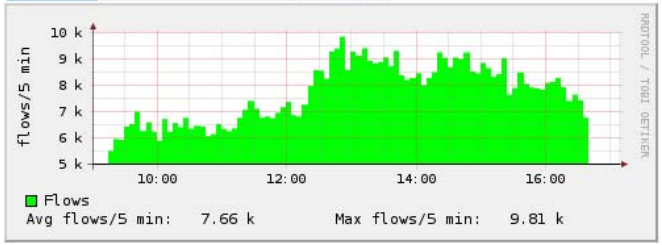
alarm | Events

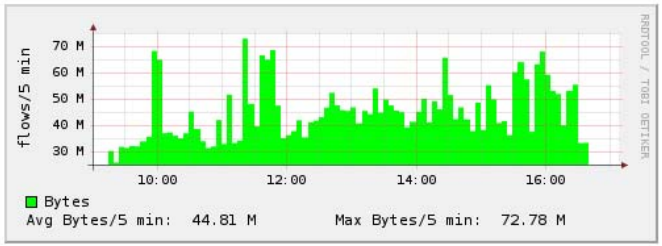
report | setup | thresholds | botnets | details: 50598

Remove tab

analyse

ip	[redacted]
hostname	[redacted]
first packet	2007-12-14 08:55
last packet	2007-12-14 16:37
threshold	Change





Top 10 flows per 5 minutes at 2007-12-14 16:37:40:

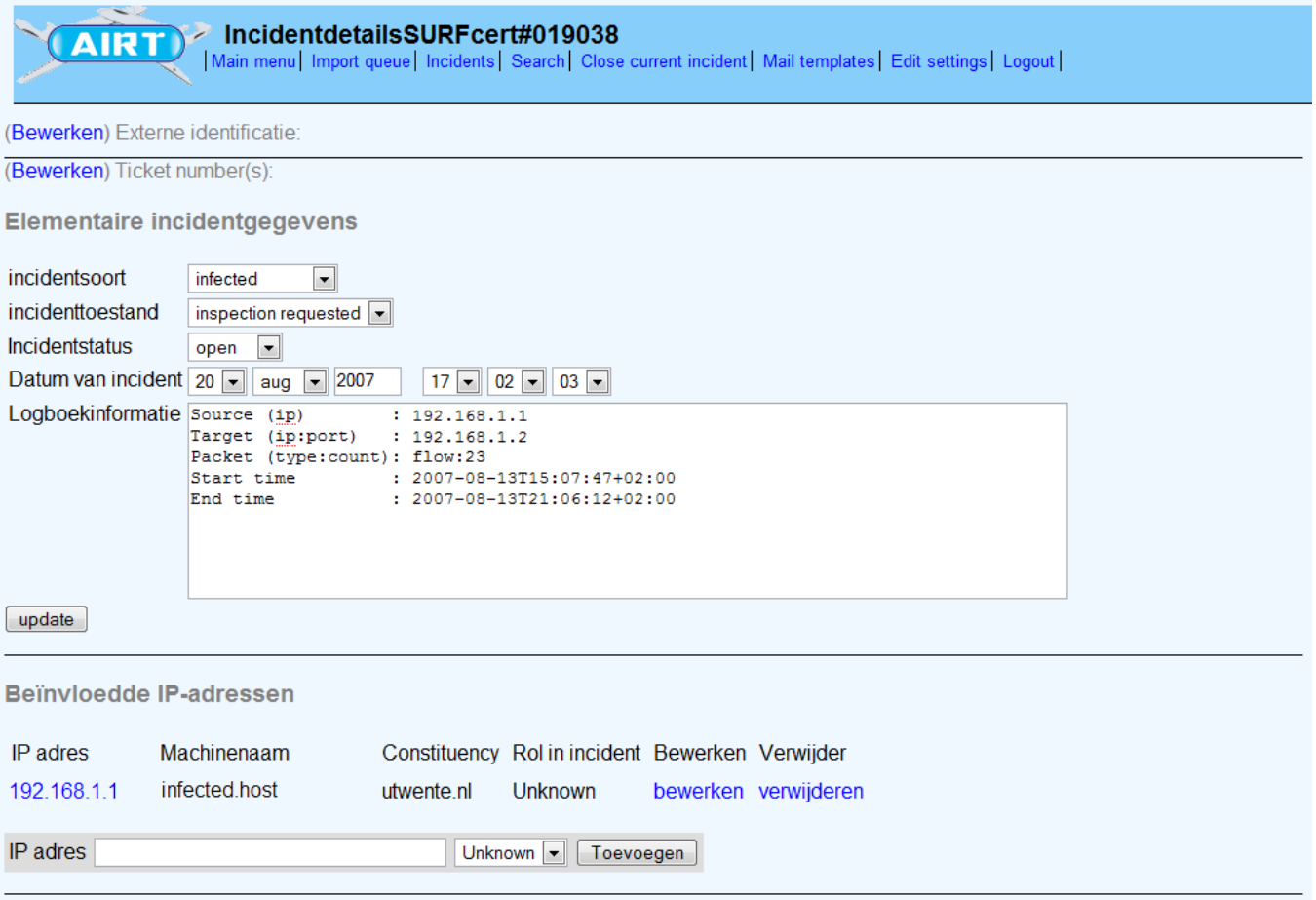
address	flows	bytes	port usage	last seen	actions
[redacted]	1379	2947950	min: 1046, max: 65508	2007-12-14 12:37:51	Report port scan / analyse
[redacted]	1353	2897466	min: 1038, max: 65509	2007-12-14 12:53:00	Report port scan / analyse
[redacted]	1342	2963856	min: 1071, max: 65502	2007-12-14 13:03:01	Report port scan / analyse
[redacted]	1341	2997262	min: 16971, max: 56329	2007-12-14 13:17:59	Report port scan / analyse

Botnet detection

- Hosts infected by viruses connect to hosts known as botnet controllers
- List of botnet controllers are available, for example:
  http://www.bleedingthreats.net/rules/bleeding-botcc.rules
- Our plug-in logs all hosts that connect to known botnet controllers
- Automatically reports to incident report system using IODEF

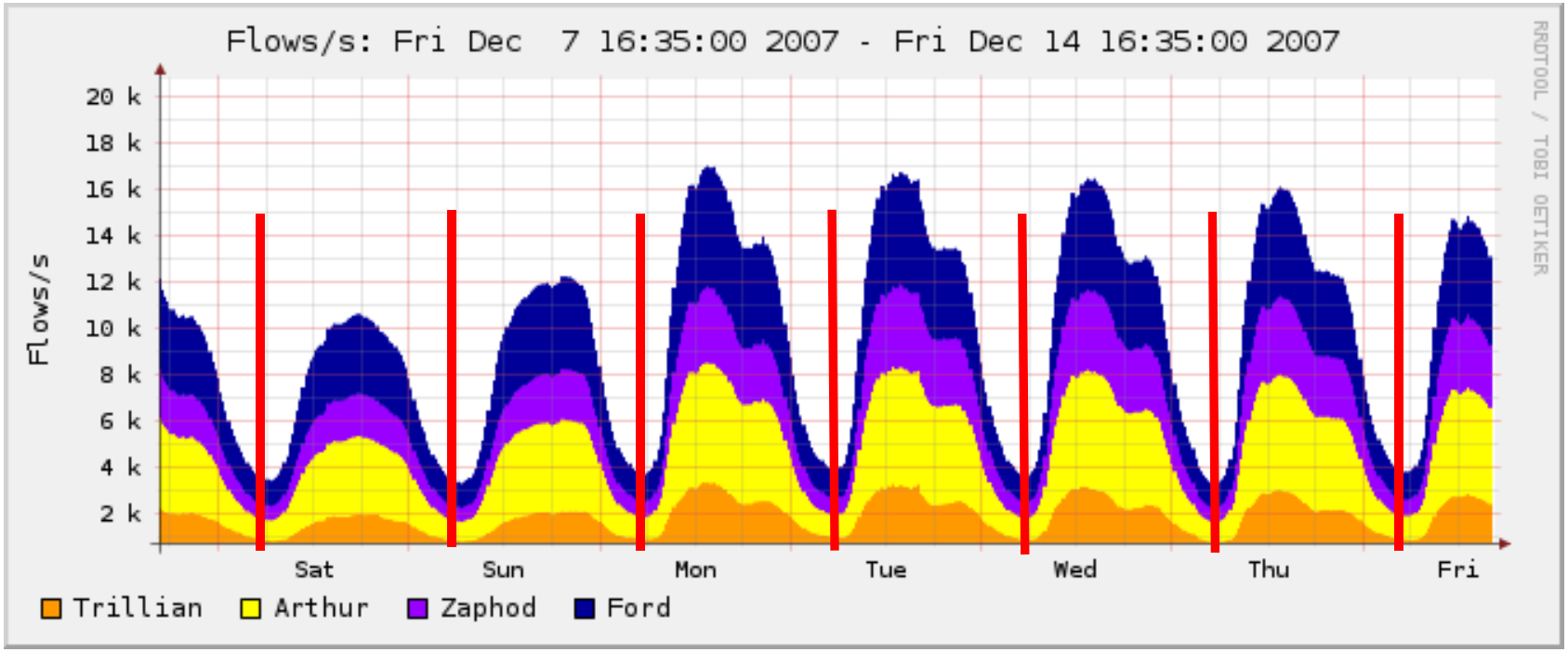
Botnet IODEF reports

```
<?xml version="1.0" encoding="iso-8859-1"?>
<io:IODEF-Document xmlns:io="urn:ietf:params:xml:ns:iodef-1.0” lang="en">
  <io:Incident purpo
    <io:IncidentID
    <io:StartTime>2
    <io:EndTime>200
    <io:ReportTime>
    <io:Assessment>
      <io:Impact ty
    </io:Assessment
    <io:Contact>
      <io:ContactNa
    </io:Contact>
    <io:EventData>
      <io:Method>
        <io:Referen
          <io:Refer
        </io:Refere
      </io:Method>
      <io:Flow>
        <io:System
          <io:Node>
            <io:Add
            <io:Cou
          </io:Node
        </io:System
        <io:System
          <io:Node>
            <io:Add
          </io:Node
          <io:Servi
            <io:Por
          </io:Serv
        </io:System
      </io:Flow>
    </io:EventData>
    <io:AdditionalD
NFSen</io:Additiona
  </io:Incident>
</io:IODEF-Document>
```

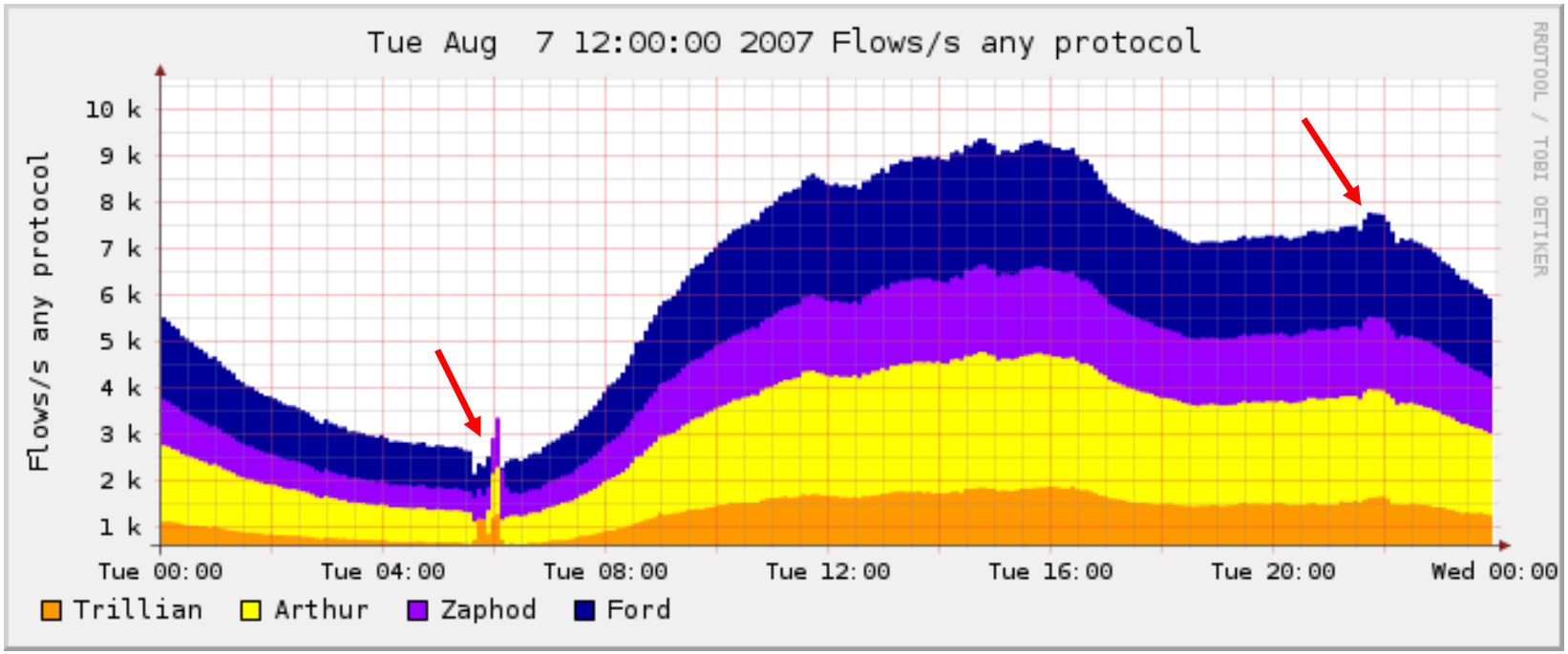


Holt-Winters aberrant behavior detection

- Uses information about periodic data to predict aberrant behavior.

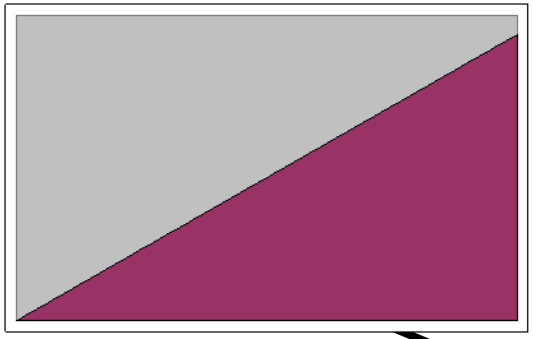


Holt-Winters: Example

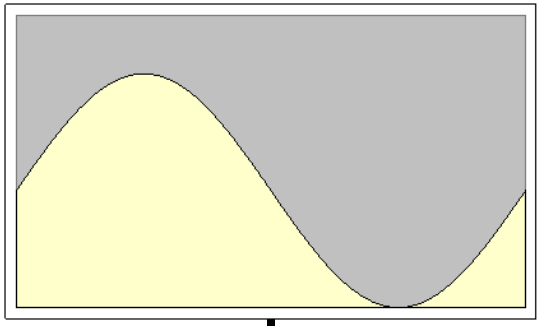


Holt-Winters: Original implementation

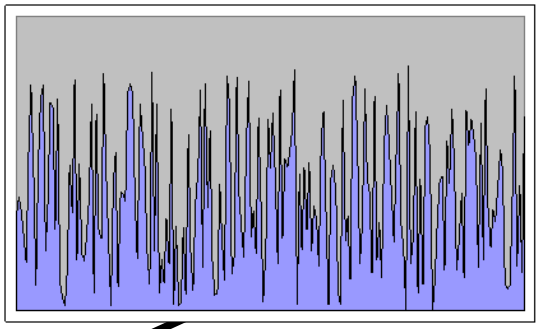
Trend

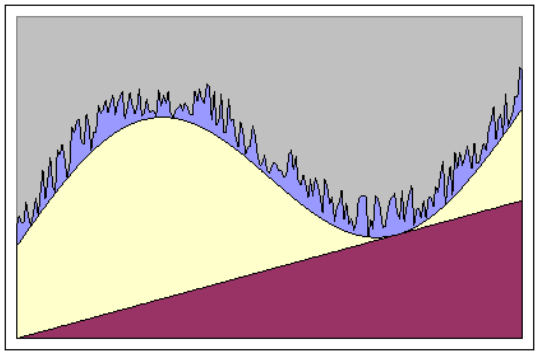


Periodic information



Noise





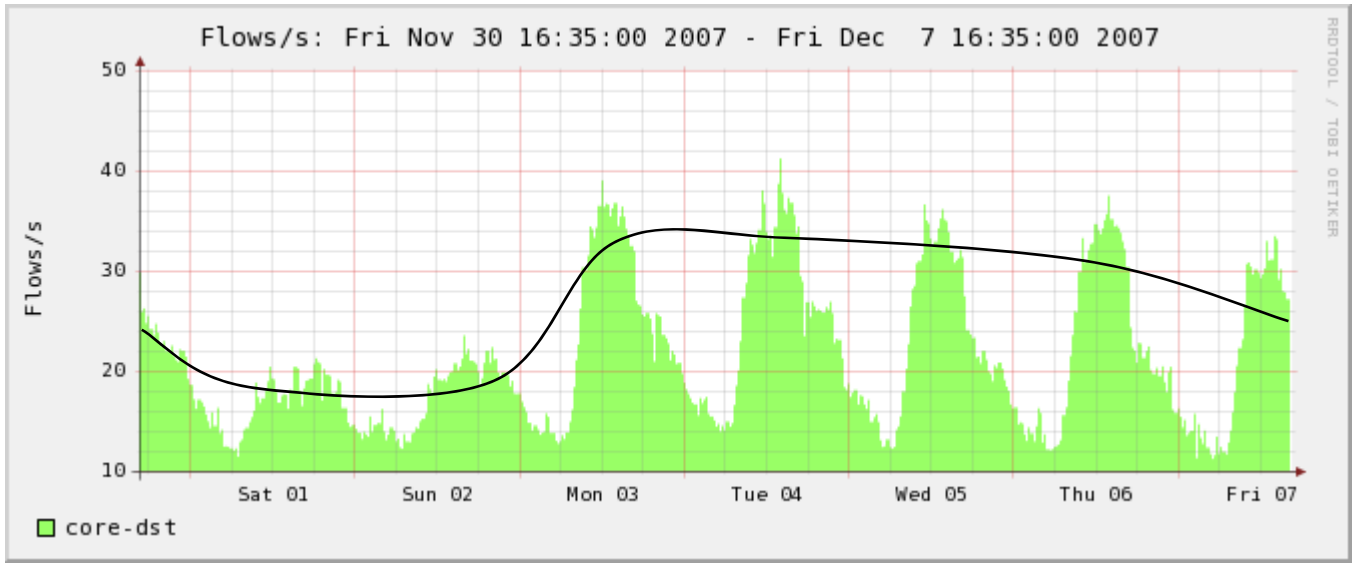
Prediction

Limitations of the original implementation

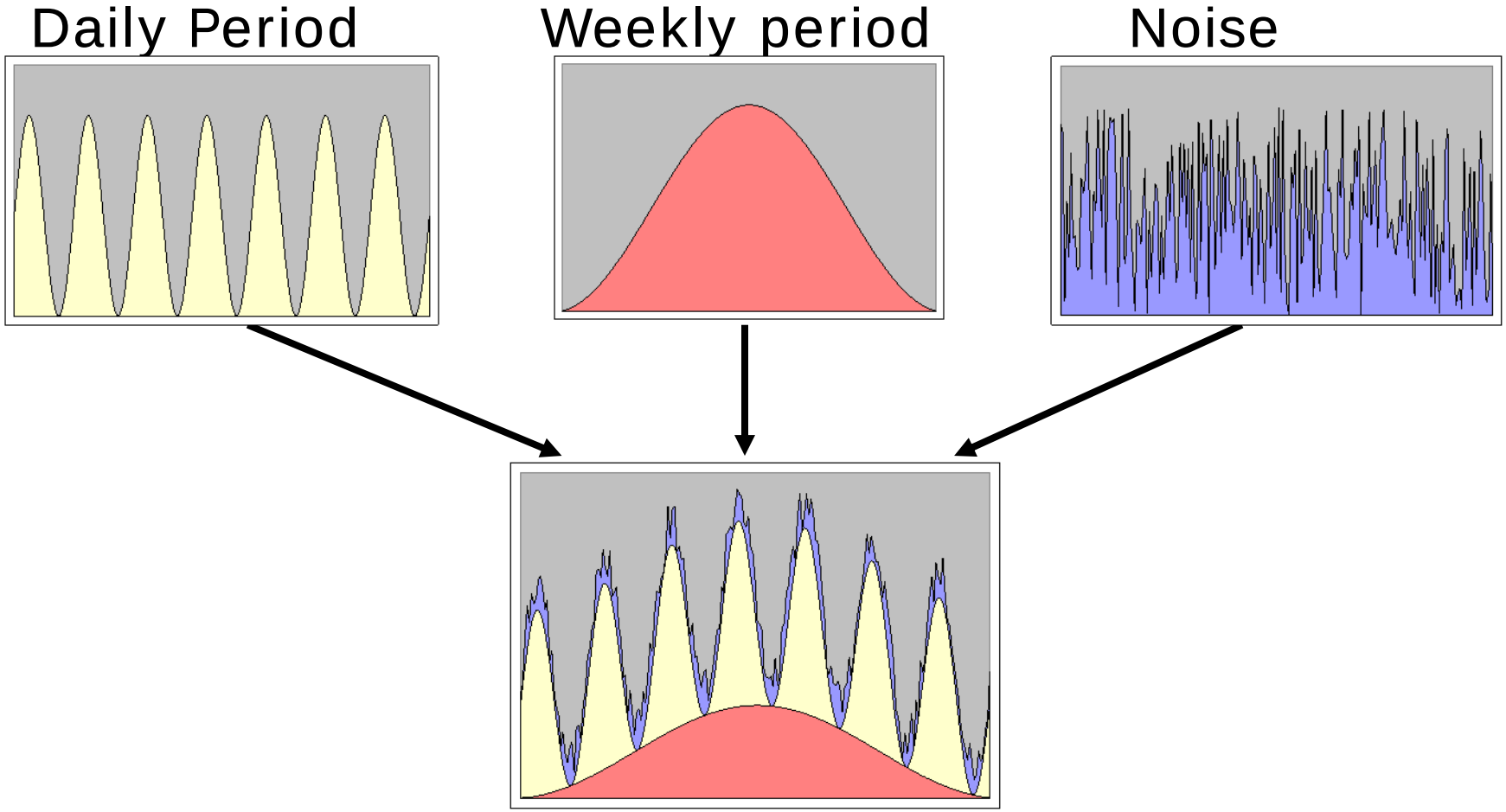
- The original algorithm has three parameters which define:
  - the weight of historical data
  - the weight of the trend
  - the amount of expected noise
- The original algorithm has a constant learning rate
  - If a low learning rate is used, the selection of the initial values is critical. This will introduce false positives for a long time.
  - With a high learning rate, the model will likely be overfitted. This will introduce false negatives
- The trend parameter has no significant influence with the resolution we are using

Holt-Winters: Multiple trends

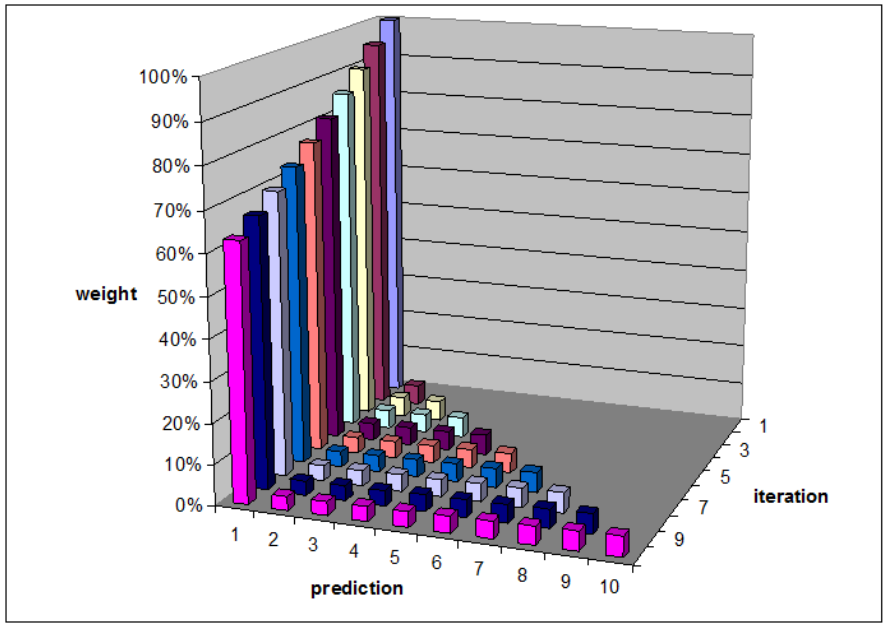
Network traffic time series often show multiple recurring patterns, for example a weekly trend:



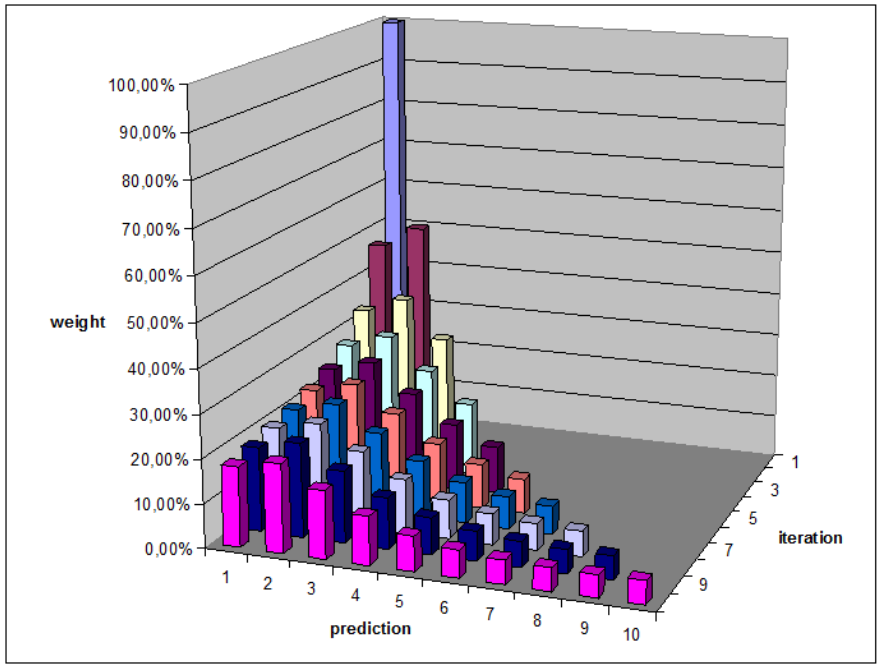
Holt-Winters: Multiple periods



Learning rate

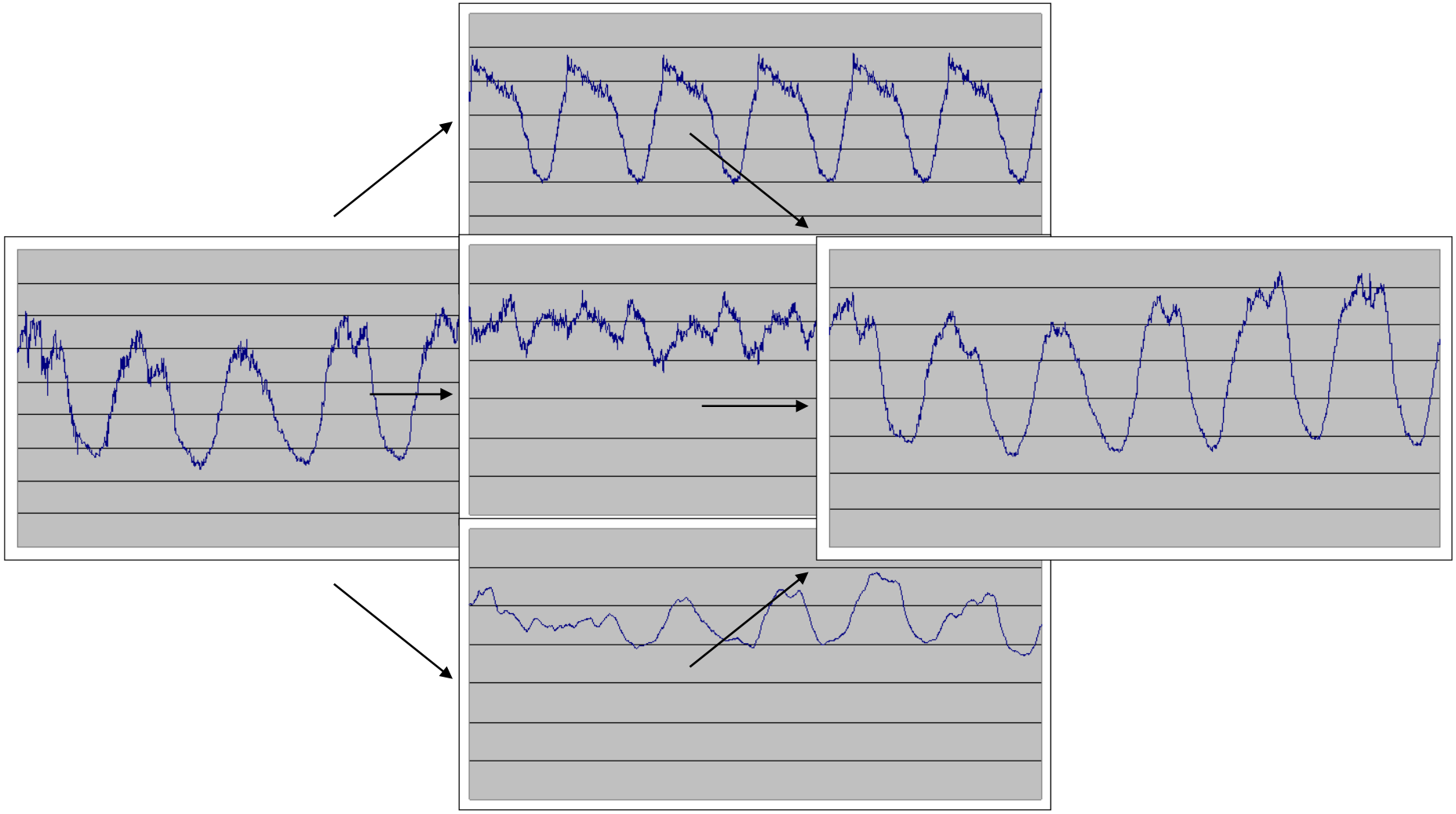


Fixed learning rate:
The first pattern is overweighted

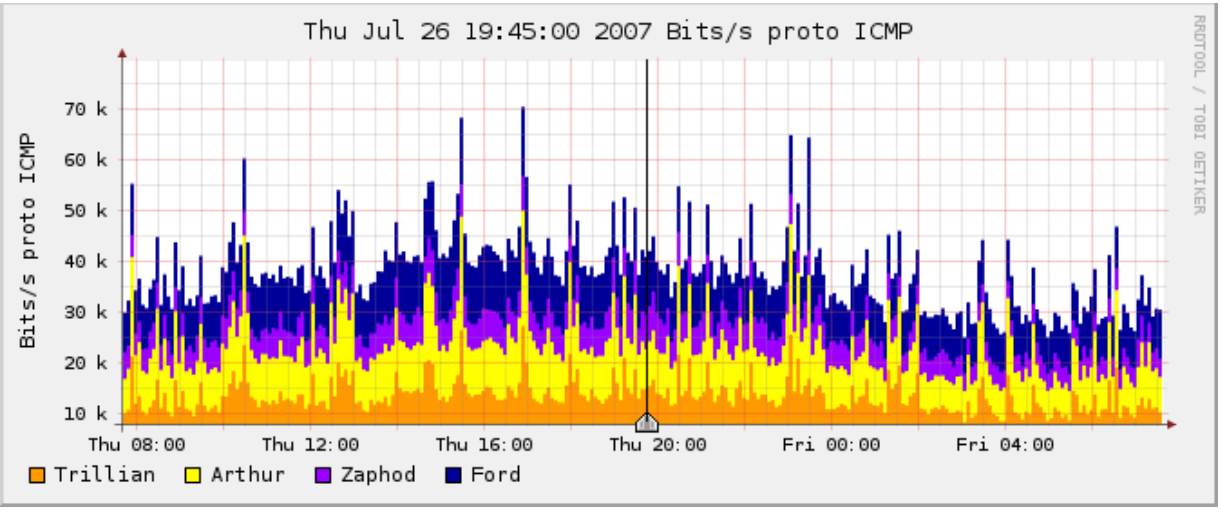


Adaptive learning rate:
The weight of the first pattern is relative to the rest

Real data example

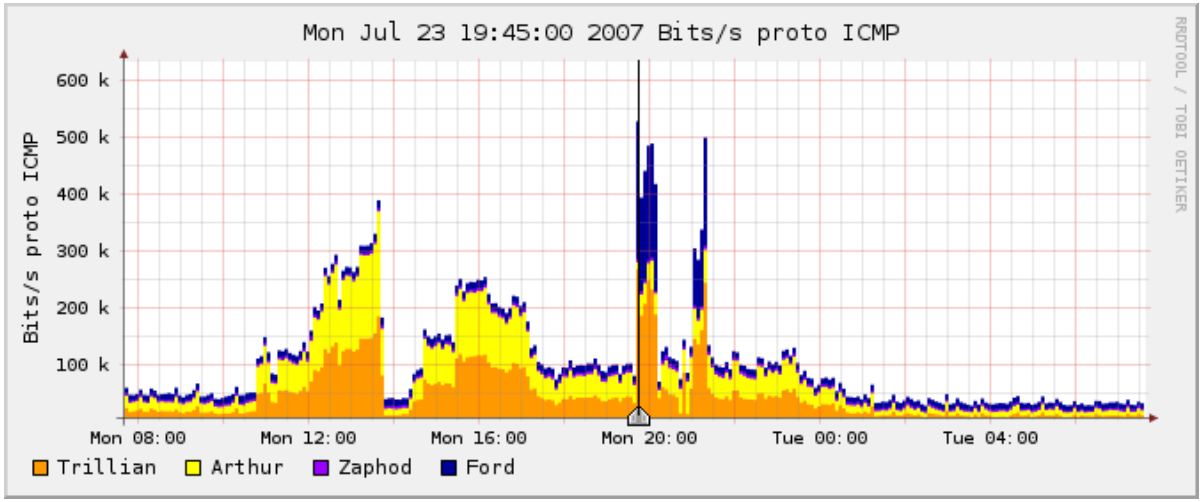


Holt Winters: Usage Example

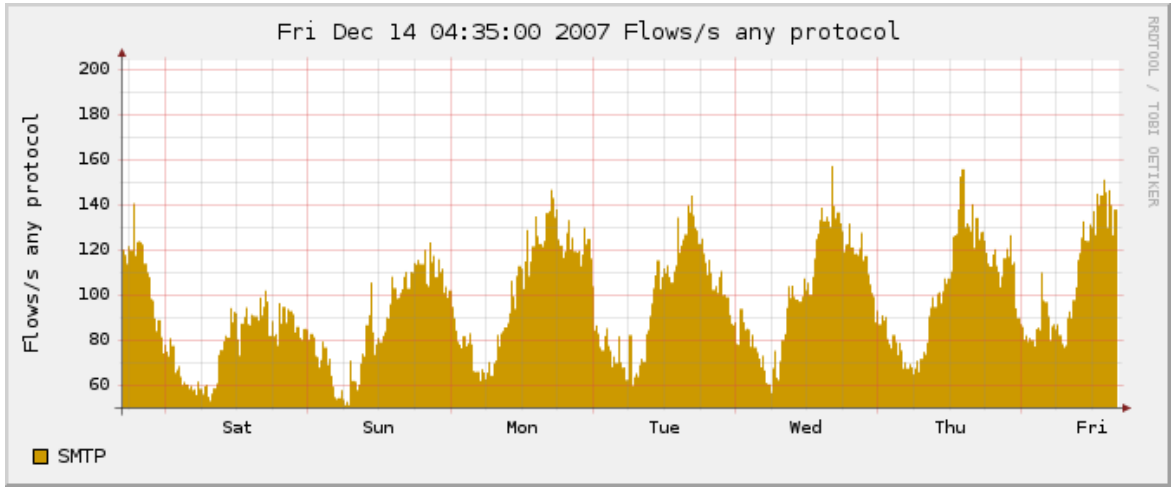


Normal ICMP Traffic

Aberrant ICMP Traffic:
Caused by DDos attack
by Stormworm
botnet

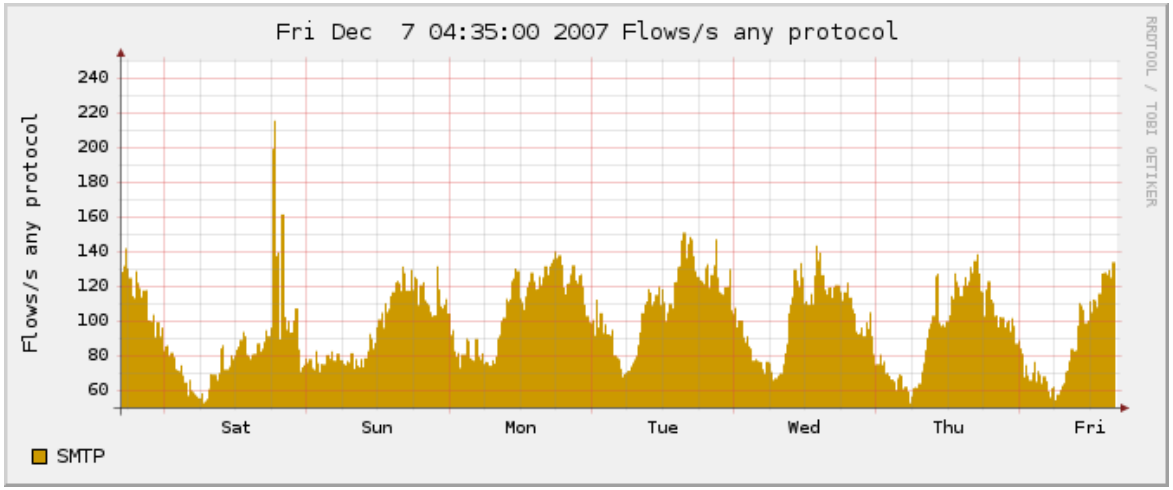


Holt Winters: Other possible uses



Common SMTP Traffic

Last week SMTP Traffic



Wim Biemolt
Wim.Biemolt@surfnet.nl
www.surfnet.nl

Werner Schram
Werner.Schram@surfnet.nl
www.surfnet.nl