

# INFORMATION SOCIETY TECHNOLOGIES (IST) PROGRAMME



## *Large Scale Monitoring of BroadBand Internet Infrastructure Contract No. 004336*

### **D 4.5b: “Second Period's Training Material”**

<b>Contractual Date of Delivery</b>	June 2006
<b>Actual Date of Delivery</b>	August 2006
<b>Deliverable Security Class</b>	Public
<b>Editor</b>	Anna Doxastaki
<b>Contributors</b>	Evangelos Markatos, Herbert Bos, Arne Oslebo

The LOBSTER Consortium consists of:

FORTH-ICS	Coordinator	Greece
VU	Partner	Netherlands
CESNET	Partner	Czech Republic
UNINETT	Partner	Norway
ENDACE	Partner	United Kingdom
Alcatel	Partner	France
FORTHnet	Partner	Greece
TNO	Partner	Netherlands
TERENA	Partner	Netherlands



## Table of Contents

<i>Overview</i> _____	<b>3</b>
<i>Tutorial Evaluation</i> _____	<b>4</b>
<i>List of Participants</i> _____	<b>6</b>
<i>Annex I - Questionnaire</i> _____	<b>8</b>
<i>Annex II – slides of presentations</i> _____	<b>9</b>



## Overview

The LOBSTER Tutorial took place on the 15<sup>th</sup> of May 2006, in conjunction with the TERENA Networking Conference (<http://www.terena.nl/events/tnc2006/>), in Catania, Italy. The speakers of the Tutorial came from some partners of the project, in particular:

- FORTH - Evangelos Markatos
- VU – Herbert Bos
- UNINETT – Arne Oslebo

The tutorial aimed to introduce to the attendants the principles and practice of passive network traffic monitoring, i.e. monitoring based on full packet capture and inspection. The tutorial also covered topics, such as: applications using MAPI, capturing traffic, and overview of traffic analysis.

The Tutorial comprised four talks:

- “Introduction to passive network monitoring and the packet capture (pcap) library”  
*Evangelos Markatos, (FORTH)*
- “Implementing passive monitoring applications using MAPI”  
*Arne Øslebø (UNINETT)*
- “Capturing packets with FFPF: Fairly Fast Packet Filters”  
*Herbert Bos (VU)*
- “An overview of traffic analysis using NetFlow”  
*Arne Øslebø (UNINETT)*

These presentations are available on line, through the link:  
<http://www.ist-lobster.org/events/tutorial-2006/>.

They also appear in Annex II of this report, in page 9.





## Tutorial Evaluation

A questionnaire (please refer to Annex I in page 8) was disseminated during the Tutorial to the participants, who were asked to provide their feedback in terms of the usefulness of the Tutorial. Figure 1 below shows that 43% of the participants thought that the Tutorial was good, while 32% thought it was very good.

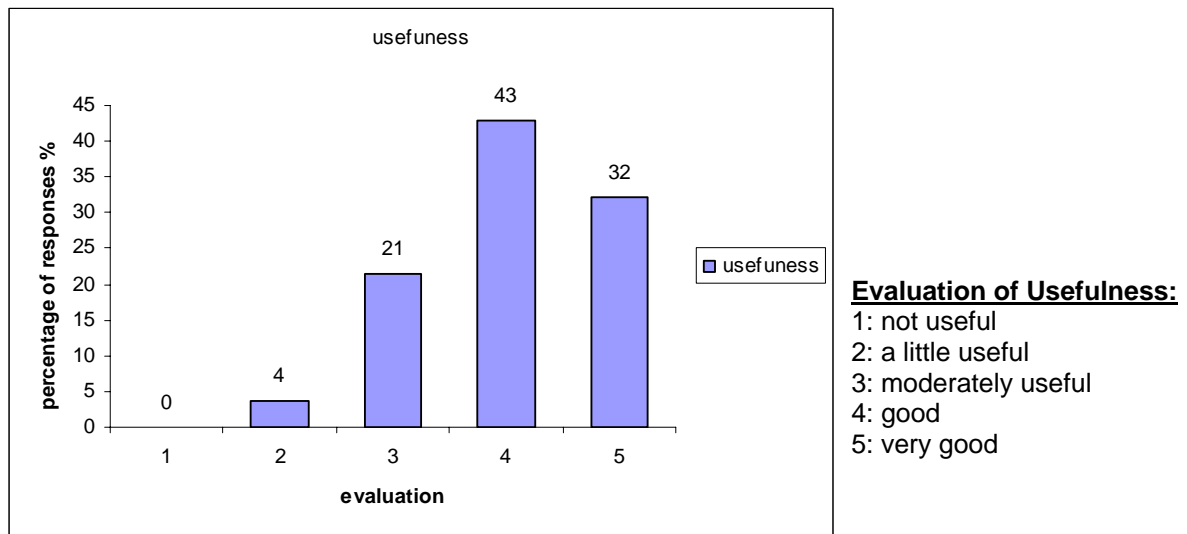


Figure 1: Perceived usefulness of the Tutorial by the participants (in percentages).

The participants were also asked to evaluate the Tutorial in terms of its organization. As it appears in Figure 2 below, about 46% of the participants thought that the organization of the Tutorial was good, while 43% thought that it was very good.

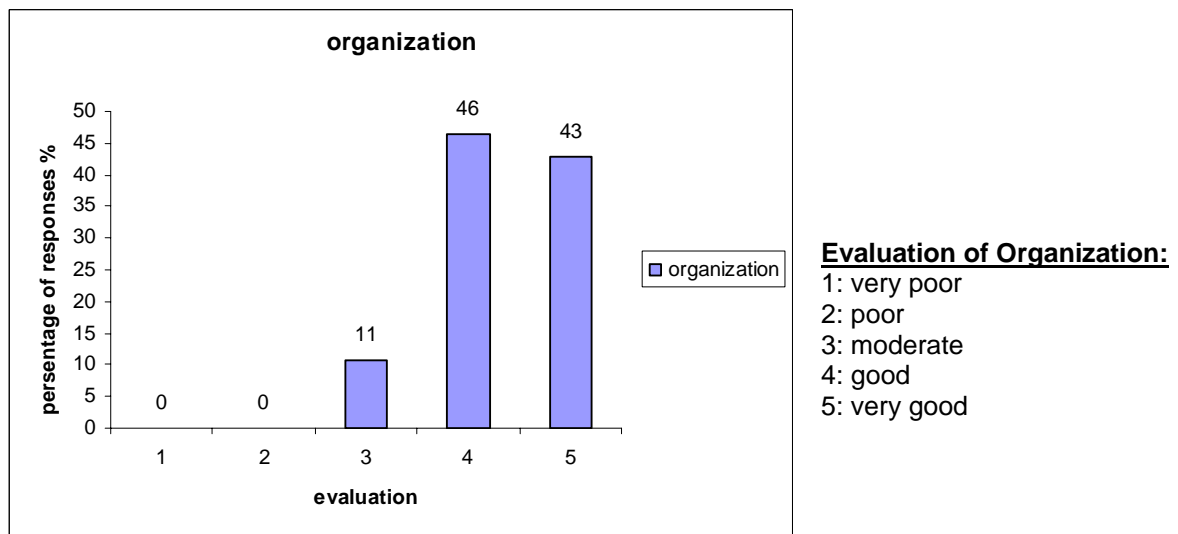


Figure 2: Evaluation of the Tutorial’s organization by the participants (in percentages).



## D 4.5b: “Second Period's Training Material” IST-004336

Finally, there was a question that involved the statement of any comments the participants could have. One of these comments was that the “developing of PCAP APIs” should be a separate topic from traffic analysis (NetFlow), since these generate interest on different groups of people.

Overall, the comments were in the lines of overall satisfaction.

Therefore, the Tutorial was successful since the majority of the participants considered it to be good in terms of usefulness. There was a lively discussion and questions about the Project and in general the attendants demonstrated their interest in the project.

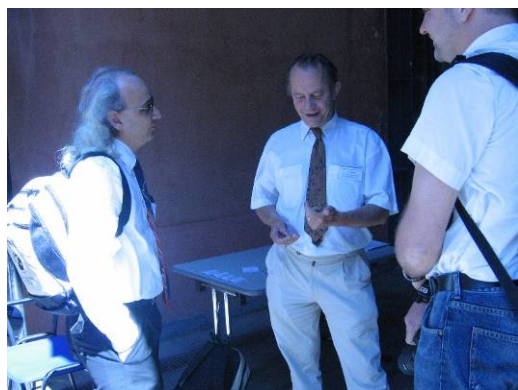
By taking into account the feedback of the participants, the work of the project will address the ideas expressed and the questions raised.



(a)



(b)



(c)



(d)

**Figure 3:** Photos from the Tutorial. From upper left to bottom right:

- a. Herbert Bos (VU) during his talk
- b. Audience at the tutorial
- c. Prof. Markatos talking with Mr. Wim Jansen, LOBSTER Project Officer
- d. Arne Øslebø (UNINETT) during his talk



## List of Participants

The number of registered participants exceeded 50, but the actual participants were about 60, since some of them attended at the last minute.

1. Maurizio Aiello	CNR - IEIIT
2. Lachlan Andrew	Caltech
3. Marco Bencivenni	INFN-CNAF
4. Spiros Bolis	University of Athens
5. Ronald Boontje	Universiteit van Amsterdam
6. Joe Burns	Queen's University Belfast
7. Stefano Catani	University of Trieste
8. Pavel CELEDA	CESNET z. s. p. o.
9. Massimiliano CICCARELLI	DATAMAT/ESA
10 Sandra Denasi	INRIM
11 Freek Dijkstra	Universiteit van Amsterdam
12 Anna Doxastaki	FORTH-ICS
13 Giacomo Fazio	INAF IASF
14 David Foster	CERN
15 Giorgio Giorgetti	University of Trieste
16 Lisa Golka	Univ. of Stuttgart
17 Paola Grosso	UvA
18 Mike Harron	University of Hertfordshire
19 Wim Jansen	European Commission DG INFSO
20 Peter Juul	Uni-C / Danish NREN
21 Dariusz Klimowicz	TASK
22 Markus Krieger	Universitaet Wuerzburg
23 Felix Kugler	SWITCH
24 Ladislav Lhotka	CESNET
25 Eric Lopez	RAICES - CLARA
26 Evangelos Markatos	FORTH-ICS
27 Jaime Leonardo Martínez Rodríguez	Red Nacional Academica de Tecnologia Avanzada
28 Maurizio Molina	DANTE
29 Herman Moons	University of Leuven
30 Walter Eduardo Munguia Martinez	RAAP-CLARA
31 Mscislaw Nakonieczny	TASK
32 Jiri Novotny	CESNET z.s.p.p
33 Brian O'Hora	University of Dublin, Trinity College
34 Aida Omerovic	UNINETT FAS
35 Chris Oornick	KLPD
36 Jukka Orajärvi	Oulu Polytech
37 Vasile Florian Pais	National Institute for Laser, Plasma & Radiation Physics
38 Michalis Polychronakis	FORTH-ICS
39 Juergen Rauschenbach	DFN-Verein



**D 4.5b: “Second Period's Training Material” IST-004336**

40 Genevieve Romier	CNRS
41 Matti Saarinen	University of Helsinki
42 Piotr Sasiedzki	Silesian University of Technology Computer Centre
43 Wolfgang Schrimm	University of Heidelberg
44 Vladimir Smotlacha	CESNET
45 rogieer spoor	SURFnet
46 Tomislav Stivojevic	CARNet
47 Fergal Suipeil	University College Dublin
48 Zbigniew S. Szewczak	Faculty of Math & Comp. Sc. N. Copernicus Univ.
49 Derek Taylor	University of Hertfordshire
50 Pedro Veiga	FCCN
51 Pavle Vuletic	University of Belgrade
52 Björn Wiberg	Uppsala University



## Annex I - Questionnaire

### LOBSTER Tutorial on Passive Network Monitoring

*Questionnaire:*

- Please rate the usefulness of the Tutorial using the scale:  
1: *not useful*, 2: *a little useful* 3: *moderately useful*, 4: *good*, 5: *very good*

1     2     3     4     5

- Which talk(s) did you find the most interesting? (Please write their numbers)

.....  
.....

- Which talk(s) did you find the least interesting? (Please write their numbers)

.....  
.....

- Which topics would you like to see presented in a future LOBSTER Tutorial?

.....  
.....

- Please rate the overall organization of the Tutorial using the scale:  
1: *very poor*, 2: *poor*, 3: *moderate*, 4: *good*, 5: *very good*

1     2     3     4     5

- Do you have any general comments for the Tutorial overall?

.....  
.....  
.....  
.....



## Annex II – slides of presentations

1. “Introduction to passive network monitoring and the packet capture (pcap) library”  
*Evangelos Markatos, (FORTH)*
2. “Implementing passive monitoring applications using MAPI”  
*Arne Øslebø (UNINETT)*
3. “Capturing packets with FFPF: Fairly Fast Packet Filters”  
*Herbert Bos (VU)*
4. “An overview of traffic analysis using NetFlow”  
*Arne Øslebø (UNINETT)*



Evangelos Markatos  
FORTH-ICS  
[markatos@ics.forth.gr](mailto:markatos@ics.forth.gr)

<http://www.ics.forth.gr/~markatos>  
Institute of Computer Science (ICS)  
Foundation for Research and Technology – Hellas (FORTH)

Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH

- Motivation
  - What is network traffic monitoring?
  - Why is it important?
- Two methodologies for monitoring
  - **Active** network monitoring
    - Examples
  - **Passive** network monitoring
    - Examples
- pcap
- Introduce the rest of this tutorial
- Summary and Conclusions



← Focus of this tutorial

Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH

- Committee on **Research Horizons in Networking** (formed in 2001)
  - **David Patterson, Chair**, University of California at Berkeley
    - RISC processors, RAID storage, NOWs – clusters of workstations
  - **David Clark**, MIT Laboratory for Computer Science
    - The “father” of the “end-to-end” argument, on top of which the Internet design is based
  - **Anna Karlin, Jim Kurose, Edward D. Lazowska, David Liddle, Derek McAuley, Vern Paxson, Stefan Savage, Ellen Zegura**
- The committee was assigned to
  - “**formulate a fresh look at networking research**”
- They prepared a report
  - “**Looking over the fence at networks: a Neighbor’s View of Networking Research**”
  - They identified three “**Grand Research Challenges**”



Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH

**The first GRAND Challenge in Computer Networking is to**

*“... develop and deploy the technology to make it possible to record a day in the life of the Internet...”*



Committee on Research Horizons in Networking  
Clark, Lazowska, Patterson, Paxson, Savage, Zegura, ....

2001



Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

- Why is it important?
  - “a data set with **typical days for the next 10 years of the Internet** might be a **treasure chest for networking researchers**”
  - **Measurement-based GRAND challenges**, such as the **human genome**, have served to
    - Crystallize research issues, and
    - Mobilize research efforts
  - Good network monitoring data are necessary for **operational needs**
    - Why is my network slow?
    - Which route do my packets follow?
    - Why is a particular flow missing lots of packets?
    - How much peer-to-peer traffic is there?
- **Next GRAND Challenge in Networking Research:**
  - **Monitor a day in the life of the Internet**



Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH

- So, do Network Traffic Monitoring
  - To get a better understanding on “what’s on the network”
- What can you do with monitoring?
  - Know the **state** of the Internet and your network
  - **Capacity** planning
  - **Traffic accounting**
    - Which application generates most traffic?
  - Understand the **performance** of individual applications
    - “why is my application so sloooooooooow”?
  - Detect Security threats
    - DoS Attacks, Worm outbreaks



Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

- Motivation
  - What is network traffic monitoring?
  - Why is it important?
- Two methodologies for monitoring
  - **Active** network monitoring
    - Examples
  - **Passive** network monitoring
    - Examples
- pcap
- Introduce the rest of this tutorial
- Summary and Conclusions

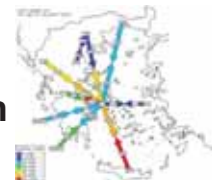


← Focus of  
this tutorial

Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH

- Inject packets into the network
- Measure their arrival time, loss rate, etc.
- What can you do with it?
  - Measure **delay** (one-way / two-way)
  - Measure **bottleneck link bandwidth**
  - Find **network topology**
  - **What's up?** (in my network)
    - Which nodes are up and running?



Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)



## Ping: is the receiver host alive?



<http://www.ist-lobster.org/>

```
%> ping www.ist-lobster.org
PING www.ist-lobster.org (192.87.30.11): 56 data bytes
64 bytes from 192.87.30.11: icmp_seq=0 ttl=49 time=308.7 ms
64 bytes from 192.87.30.11: icmp_seq=1 ttl=49 time=307.6 ms
64 bytes from 192.87.30.11: icmp_seq=2 ttl=49 time=244.4 ms

--- www.ist-lobster.org ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 244.4/286.9/308.7 ms
%>
```

Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH



## Traceroute: find all intermediate routers between a source and a destination computer

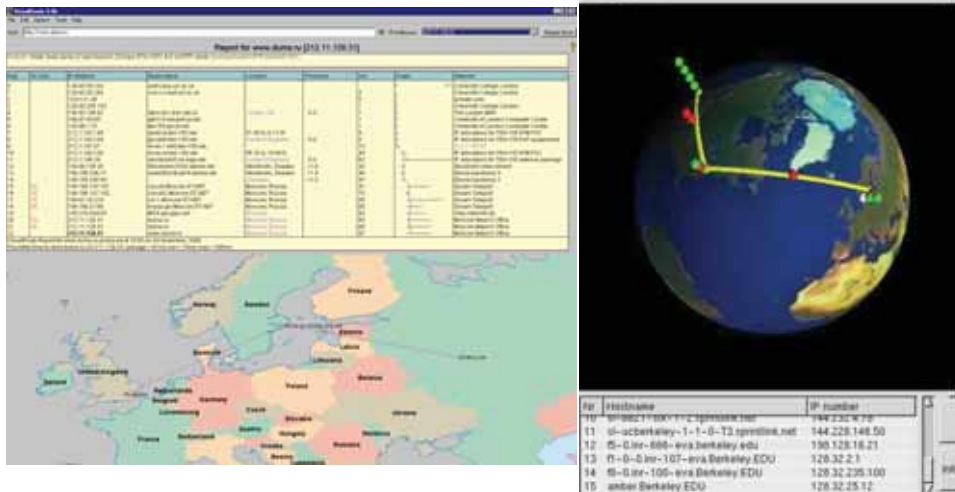


<http://www.ist-lobster.org/>

```
%> traceroute www.ist-lobster.org (from Crete)
traceroute to www.ist-lobster.org (192.87.30.11), 30 hops max, 40 byte packets
 1 147.52.17.1 (147.52.17.1) 1.050 ms 0.690 ms 0.592 ms
 2 olympos-e43.lanh.uoc.gr (147.52.12.1) 1.626 ms 1.033 ms 0.840 ms
 3 heraklio-uch-ATM.grnet.gr (194.177.209.141) 129.528 ms 112.644 ms 123.465 ms
 4 heraklio2-to-heraklio.backbone.grnet.gr (194.177.209.77) 124.791 ms 116.749 ms 119.965 ms
 5 Syros-to-Heraklio2.backbone.grnet.gr (195.251.27.81) 136.089 ms 104.469 ms 81.116 ms
 6 athens3-to-Syros.backbone.grnet.gr (195.251.27.10) 72.664 ms 62.814 ms 67.341 ms
 7 grnet.gr1.gr.geant.net (62.40.103.57) 81.392 ms 102.067 ms 79.488 ms
 8 gr.de2.de.geant.net (62.40.96.82) 129.641 ms 134.589 ms 144.765 ms
 9 de2-2.de1.de.geant.net (62.40.96.54) 139.478 ms 158.336 ms 146.815 ms
10 de.nl1.nl.geant.net (62.40.96.102) 180.696 ms 162.904 ms 173.813 ms
11 surfnet-gw.nl1.nl.geant.net (62.40.103.98) 184.078 ms 158.921 ms 160.933 ms
12 PO11-0.CR1.Amsterdam1.surf.net (145.145.166.33) 145.367 ms 150.166 ms 142.117 ms
13 PO0-0.AR5.Amsterdam1.surf.net (145.145.162.2) 163.605 ms 144.161 ms 177.526 ms
14 145.145.18.46 (145.145.18.46) 178.350 ms 175.365 ms 166.334 ms
15 ** 145.145.18.46 (145.145.18.46) 176.079 ms !X
16 * 145.145.18.46 (145.145.18.46) 171.861 ms !X *
17 145.145.18.46 (145.145.18.46) 192.753 ms !X * 180.104 ms !X
```

Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH



Evangelos Markatos info@ist-lobster.org

FORTH

- Motivation
  - What is network traffic monitoring?
  - Why is it important?
- Two methodologies for monitoring
  - **Active** network monitoring
    - Examples
  - **Passive** network monitoring
    - Examples
- pcap
- Introduce the rest of this tutorial
- Summary and Conclusions



← **Focus of this tutorial**

Evangelos Markatos info@ist-lobster.org

FORTH

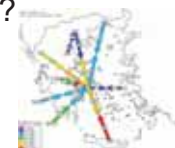
- What is it?
- Non-intrusive traffic monitoring
  - Much like a **telescope**
  - **Does not inject packets** in the network
- It **passively captures** information from passing packets such as
  - High-level network flows (CISCO Netflow)
  - Network packet headers (NLANR)
  - Entire network packets (incl. payload)
    - if allowed
    - maybe stripped/anonymized (to be shared with a broader audience)



Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

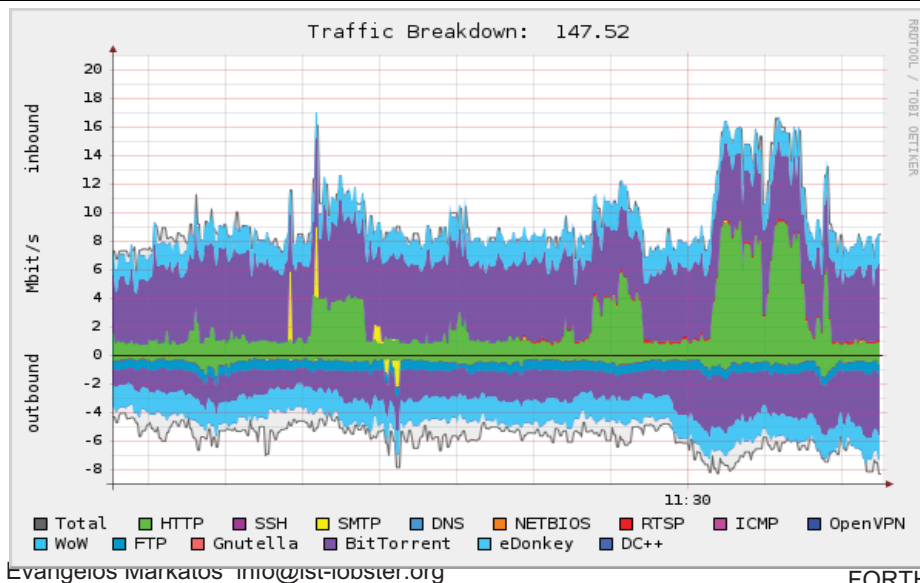
FORTH

- Traffic Categorization/Accounting:
  - What % of my traffic is due to email?
  - Which subnet generates most outgoing traffic?
- Bandwidth Estimation
  - What % of my bandwidth is available now?
  - What % of my bandwidth is being used?
- Study trends:
  - How does the application mix in the traffic changes with time?
    - ftp in the 80's, www in the 90's, p2p in the 00's
  - How does peer-to-peer traffic changes with time?
- Performance Debugging of individual applications
  - Why is **my** application so sloooow?



Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH



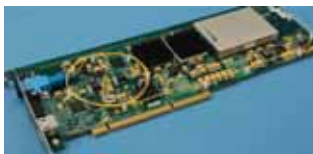
- **Intrusion Detection**
  - Are any of my computers compromised?
    - Do they participate in a botnet?
  - Is there any attacker trying to intrude into my network?
- **Large-scale Attack Detection** – Detection of Epidemics
  - DoS Attack detection
    - e.g. Detect sharp increases in TCP/SYN packets
  - Zero-day worm detection
    - e.g. Detect lots of identical packets, never seen before, from several sources to several destinations
    - e.g. Detect worm characteristics
      - such as NOP sleds: long sequences of executable code
- **Network Telescopes**
  - They monitor unused IP addresses (“dark matter”)
  - Ordinarily, unused IP addresses should not receive traffic
  - Observe victims of DoS attacks
    - “back-scatter” traffic, “background radiation”
  - Observe infected hosts
  - Port scans



- Equipment varies from
  - low-end (for low-speed networks) to
  - Sophisticated equipment (for high-speed networks)
- Low-end passive monitors (100Mbps – 1Gbps)
  - An ordinary PC
  - An ordinary network Interface (i.e. an Ethernet card)
    - put in promiscuous mode
  - Mirror all packets from a router to a port connected to the above PC

Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH



- High-end passive monitors (1Gbps – 10Gbps)
  - High-end computer
  - Specialized network interface
    - DAG Cards (Endace)
    - Combo cards (SCAMPI project)
    - Hardware-based filtering capabilities
      - Process packets at line speeds



Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH

- pcap: packet capture library from Berkeley
- MAPI: Monitoring API (Application Programming Interface)
  - <http://www.ist-lobster.org/downloads>
  - Developed within the IST SCAMPI project
    - co-funded by EU
- Net-flow-related tools
  - Graphical interfaces



Protocol	Source	Destination	Bytes	Packets	Errors
IP	192.168.1.1	192.168.1.2	10000	100	0
TCP	192.168.1.1	192.168.1.2	5000	50	0
UDP	192.168.1.1	192.168.1.2	5000	50	0
ICMP	192.168.1.1	192.168.1.2	5000	50	0
Other	192.168.1.1	192.168.1.2	5000	50	0

Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH

- Motivation
  - What is network traffic monitoring?
  - Why is it important?
- Two methodologies for monitoring
  - **Active** network monitoring
    - Examples
  - **Passive** network monitoring
    - Examples
- pcap
- Introduce the rest of this tutorial
- Summary and Conclusions



← Focus of this tutorial

Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH

- Packet capture Library
- Developed to capture packets
  - And dump them to disk
- Basic Steps:
  - Open a network interface
  - Get one packet at a time
  - Print it – dump it to the disk

Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH

- Example: Grab one packet:

```
main() {  
    /* open a network interface */  
    descr = pcap_open_live(dev, BUFSIZ, 0, 1, errbuf);  
    /* Get next packet */  
    packet = pcap_next(descr, &hdr);  
    /* print its length */  
    printf("Grabbed packet of length %d\n", hdr.len);  
}
```

Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH

- Example: Grab several packets:

```
main() {
    /* open a network interface */
    descr = pcap_open_live(dev,BUFSIZ,0, 1,errbuf);
    while (1) {
        /* Grab packets for ever */
        packet = pcap_next(descr,&hdr);
        /* print its length */
        printf("Grabbed packet of length %d\n", hdr.len);
    }
}
```

Evangelos Markatos info@ist-lobster.org

FORTH

- Example: Grab several packets:

```
main() {
    /* open a network interface */
    descr = pcap_open_live(dev,BUFSIZ,0, 1,errbuf);
    pcap_loop (descr,atoi(argv[1]),my_callback, NULL);
}

/*callback function */
void my_callback (u_char *u,const struct pcap_pkthdr* pkthdr,
    const u_char* packet)
{
    /* just count the number of packets */
    static int count = 1;
    printf("%d, \n",count);
    count++;
}

Grab 100 packets:
%>a.out 100
```

Evangelos Markatos info@ist-lobster.org

FORTH

- Example: Suppose that you want to capture only packets destined to your web servers (**destination port 80**):

```
main() {
descr = pcap_open_live(dev,BUFSIZ,0, 1,errbuf);
/* install a filter */
pcap_compile(descr,&fp,"dst port 80",0,netp)
pcap_setfilter(descr,&fp)

pcap_loop (descr,atoi(argv[1]),my_callback, NULL);
}

/*callback function */
void my_callback (u_char *u,const struct pcap_pkthdr* pkthdr, const
u_char* packet)
{
static int count = 1;
printf("%d, \n",count);
count++;
}
}
```

Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH

- Excellent for capturing and dumping packets by a single application
- Several applications
  - High overhead – copies all packets to all applications
- Limited functionality
  - No string searching, no packet counting, etc.

Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH

- <http://www.cet.nau.edu/~mc8/Socket/Tutorials/section1.html>
- <http://www.tcpdump.org/>
- <http://www.winpcap.org/>

Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH

- MAPI
  - Daemon-based packet monitoring
  - Rich functionality (string searching, counters, anonymization, etc.)
- FFPF (Fairly Fast Packet Filters)
- Flow-level passive monitoring
  - General introduction
    - NetFlow and IPFIX
  - NetFlow-based applications
    - Stager (UNINETT)
    - NERD (TNO)
- Ruler Anonymization Language

Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH



### *“Monitor a Day in the Life of the Internet”*

Committee on Res. Horizons in Networking



- Traffic Monitoring help us understand what's on the network
- Passive Network Traffic Monitoring applications for :
  - Performance
    - traffic accounting/categorization
    - Performance debugging
  - Security
    - DoS attack detection,
    - Internet epidemics
    - Intrusion Detection

Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH



Evangelos Markatos  
FORTH-ICS  
[markatos@ics.forth.gr](mailto:markatos@ics.forth.gr)

<http://www.ics.forth.gr/~markatos>  
Institute of Computer Science (ICS)  
Foundation for Research and Technology – Hellas (FORTH)

Evangelos Markatos [info@ist-lobster.org](mailto:info@ist-lobster.org)

FORTH



## Implementing passive monitoring applications using MAPI

**Arne Øslebø**  
UNINETT  
Arne.Oslebo@uninett.no



## Outline

- General overview
  - What is MAPI?
  - Example
- Design
- Using MAPI
  - MAPI calls
  - MAPI functions
- Using DiMAPI
- Extending MAPI



- Monitoring Application Programming Interface
- Design goals:
  - Make it quick and easy to implement new monitoring applications
  - Low overhead
  - Support for multiple concurrent users and applications
    - Optional support for strong authentication.
  - Global optimization
    - Optimize processing of packets based on all applications from all users.
  - Transparent support for different hardware adapters
    - NIC, DAG, COMBO6
  - Easy to extend



### Worm detection:

```
1: fd=mapi_create_flow("/dev/dag0");
2: mapi_apply_function(fd,"BPF_FILTER","src port 1234");
3: ctr_id1=mapi_apply_function(fd,"PKT_COUNTER");
4: mapi_apply_function(fd,"STR_SEARCH","pattern",100,300);
5: ctr_id2=mapi_apply_function(fd,"PKT_COUNTER");
6: mapi_apply_function(fd,"TO_FILE",MFF_PCAP,"worm.trace",0);
7: mapi_connect(fd);
8:
9: while(1) {
10:     ctr_val1=mapi_read_results(fd,ctr_id1);
11:     ctr_val2=mapi_read_results(fd,ctr_id2);
12:
13:     printf("BPF match: %llu String match: %llu\n",
14:           *ctr_val1,*ctr_val2);
15:     sleep(10);
16: }
```





## MAPI function libraries



<http://www.ist-lobster.org>

- Collection of MAPI functions that can be applied to a MAPI flow
  - BPF, string search, packet counter, jitter, statistical function etc.
- Functions can:
  - filter packets
  - transform packets
  - return or process information from packets
  - read and process results from other functions
  - etc.
- Standard library: 24 functions

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

7



## BINOP



<http://www.ist-lobster.org>

- Adds or subtracts values from two other MAPI functions
- Arguments:
  - fd and fid for left function: int
  - fd and fid for right function: int
  - type:int (BINOP\_ADD,BINOP\_SUB)
- Return value:
  - unsigned long long
- Example:
  - `ts1=mapi_apply_function(fd,"PKT_INFO",PKT_TS);`
  - `ts2=mapi_apply_function(fd2,"PKT_INFO",PKT_TS);`
  - `binop=mapi_apply_function(fd,"BINOP",`  
`fd2,ts2,fd1,ts1,BINOP_SUB);`

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

8



## BPF\_FILTER



<http://www.ist-lobster.org>

- Filters the packets according to a BPF filter expression
- Arguments
  - bpf\_filter : char\*
    - BPF filter
- Return value:
- Usage:
  - `mapi_apply_function(fd, "BPF_FILTER", "port 80");`



## BUCKET



<http://www.ist-lobster.org>

- Returns results from other MAPI functions in precise periodical intervals
- Arguments
  - interval : char\*
  - fd : int
  - function : int
- Return value:
  - void \*
- Usage:
  - `count=mapi_apply_function(fd, "PKT_COUNTER");`
  - `mapi_apply_function(fd, "BUCKET", fd, count, "1s");`



## BYTE\_COUNTER



<http://www.ist-lobster.org>

- Counts number of bytes in a flow
- Arguments
- Return value:
  - unsigned long long
- Usage:
  - `mapi_apply_function(fd, "BYTE_COUNTER");`



## COOKING



<http://www.ist-lobster.org>

- Processes the packets of a flow by performing IP defragmentation and TCP stream reassembly
- Arguments
  - threshold : int
    - minimum stream size, default 32Kb
  - timeout : int
    - timeout value for further processing since the first arrival of a packet. Default 30 seconds
- Return value:
- Usage:
  - `mapi_apply_function(fd, "COOKING", -1, -1);`



## DIST



<http://www.ist-lobster.org>

- Returns an array that represents the distribution of results from another MAPI function.
- Arguments
  - fd : int
  - function : int
  - min : char\*
  - max : char\*
  - interval : char\*
- Return value:
  - unsigned long long []
- Usage:
  - gap=map\_i\_apply\_function(fd,"GAP");
  - map\_i\_apply\_function(fd,"DIST",fd,gap,  
"1ms","2ms","1us");

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

13



## GAP



<http://www.ist-lobster.org>

- Returns the time delay between two consecutive packets in a flow
- Arguments
- Return value:
  - unsigned long long
- Usage:
  - map\_i\_apply\_function(fd,"GAP");

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

14



## HASH



<http://www.ist-lobster.org>

- Computes the additive hash function over the packets of a network flow.
- Arguments
  - layer : int
    - link, network, IP or application layer
- Return value:
  - unsigned int
- Usage:
  - `mapi_apply_function(fd, "HASH");`



## HASHSAMP



<http://www.ist-lobster.org>

- Hashing based sampling
- Only processes TCP and UDP packets
- Calculates hash on src/dst IP and src/dst port
- Arguments
  - range : int
  - keep : int
- Return value:
  - struct sample {
    - unsigned int source\_ip;
    - unsigned int dest\_ip;
    - ..
- Usage:
  - `mapi_apply_function(fd, "HASHSAMP", 100, 1);`



## PKT\_COUNTER



<http://www.ist-lobster.org>

- Counts number of packets in a flow
- Arguments
- Return value:
  - unsigned long long
- Example:
  - `mapi_apply_function(fd, "PKT_COUNTER");`



## PKT\_INFO



<http://www.ist-lobster.org>

- Returns information about a packet
- Arguments
  - info : int
    - PKT\_TS – packet timestamp
    - PKT\_SIZE – packet size
- Return value:
  - unsigned long long
- Usage:
  - `mapi_apply_function(fd, "PKT_INFO", PKT_SIZE);`



## PROTINFO



<http://www.ist-lobster.org>

- Returns a specific protocol field
- Arguments
  - info : int
    - PI\_TCPSEQ – TCP Sequence number
    - PI\_TCPACK – TCP Ack number
    - Will be extended to support all fields in IP, TCP and UDP
- Return value:
  - unsigned long long
- Usage:
  - `mapi_apply_function(fd, "PROTINFO", PI_TCPSEQ);`

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

19



## REGEXP



<http://www.ist-lobster.org>

- Regular expression pattern matching
- Arguments
  - str : char\*
- Return value:
- Example:
  - `mapi_apply_function(fd, "REGEXP", "[Aa]");`

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

20



## RES2FILE



<http://www.ist-lobster.org>

- Stores results from other MAPI function to a file
- Arguments
  - functions : char\*
    - list of functions to read results from
    - “<fid>@<fd>, <fid2>@<fd2>”
  - format : char\*
    - R2F\_RAW, R2F\_ULLSTR, R2F\_ULLSEC, R2F\_STATS
  - header : char\*
    - string to store in the beginning of the file
  - filename : char\*
  - interval : char\*
    - “-1” for always, “0” for once when flow closes, “1s”, “1.2ms” etc.

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

21



## RES2FILE(2)



<http://www.ist-lobster.org>

- Usage:
  - `count=mapi_apply_function(fd, "PKT_COUNT");`
  - `gap=mapi_apply_function(fd, "GAP");`
  - `sprintf(fids, "%d@d, %d@d", count, fd, gap, fd);`
  - `sprintf(format, "%d, %d", R2F_ULL, R2F_ULLSEC);`
  - `mapi_apply_function(fd, "RES2FILE", fids, format, "Count Gap", "test.res", "-1")`

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

22



## SAMPLE



<http://www.ist-lobster.org>

- Performs sampling
- Arguments
  - value : int
    - for DETERMINISTIC it specifies a sampling interval of 1 out of *value* packets. For probabilistic it specifies the probability in %.
  - mode : int
    - DETERMINISTIC or PROBABILISTIC
- Return value:
  - unsigned long long
- Usage:
  - `mapi_apply_function(fd, "SAMPLE", 100, DETERMINISTIC);`

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

23



## STARTSTOP



<http://www.ist-lobster.org>

- Starts and/or stops measurements at a specific time
- Arguments
  - start : char\*
  - stop : char\*
    - So far only relative time from the first captured packet is supported. Will be extended to support absolute time.
    - "0", "1s", "10ms"
- Return value:
- Example:
  - `mapi_apply_function(fd, "STARTSTOP", "10s", "0");`

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

24



## STAT



<http://www.ist-lobster.org>

- Returns statistical information about results from other MAPI functions
- Arguments
  - fd : int, fid : int
  - skip : int - number of packets to skip before reading first result
- Return value:
  - struct stats {  
    unsigned long long count;  
    long double sum;  
    long double sum2;  
    double max;  
    double min}
- Usage:
  - gap=mapi\_apply\_function(fd, "GAP");
  - mapi\_apply\_function(fd, "STAT", fd, gap, 1);

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

25



## STR\_SEARCH



<http://www.ist-lobster.org>

- Search the payload of a packet for a specific pattern
- Arguments
  - pattern : char\*
    - pattern to search for. Can contain non-printable characters or binary data using Snort syntax.
  - offset : int
    - offset into packet for starting the search
  - depth : int
    - maximum depth of the search
- Return value:
- Usage:
  - mapi\_apply\_function(fd, "STR\_SEARCH",  
    "ab|63 64|", 0, 1500);

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

26



## THRESHOLD



<http://www.ist-lobster.org>

- Signals when a certain threshold is reached.
- Arguments
  - type : int
    - CHAR, INT, ULL
  - fd : int
  - fid : in
  - boundary : int
    - EQ, GT, LT, EQ\_D, GT\_D, LT\_D
  - threshold : unsigned long long
  - timeout : int
  - divider : int
  - count : int
- Return value:



## TOP



<http://www.ist-lobster.org>

- Returns top X values of a field
- Arguments
  - x : int
  - protocol : int
  - field : int
- Return value:
  - unsigned int []
- Usage:
  - `mapi_apply_function(fd, "TOP", 5, TOPX_TCP, TOPX_TCP_DSTPORT);`



## TO\_BUFFER



<http://www.ist-lobster.org>

- Stores packets of a flow to a buffer so that they can be read using `mapi_get_next_packet`

- Arguments

- Return value:

```
- struct mapipkt {
    unsigned long long ts;
    unsigned short ifindex;
    unsigned caplen;
    unsigned wlen;
    unsigned char pkt;
}
```

- Usage:

```
- mapi_apply_function(fd, "TO_BUFFER");
```

*info@ist-lobster.org*

*Arne Øslebø, UNINETT*

29



## TO\_FILE



<http://www.ist-lobster.org>

- Stores packets to a file
- Arguments
  - format : int
    - MFF\_PCAP, MFF\_ERF, MFF\_RAW
  - file\_name : char\*
  - count : unsigned long long
    - number of packets to store. 0 means all packets will be stored.

- Return value:

- Usage:

```
- mapi_apply_function(fd, "TO_FILE", MFF_ERF,
                    1000);
```

*info@ist-lobster.org*

*Arne Øslebø, UNINETT*

30



## Other libraries



<http://www.ist-lobster.org>

- DAG library
  - Store packets to file using DAG ERF format
  - Filter on interface
- Anonymization library
  - Versatile packet anonymization functions
- IPFIX library
  - Export Netflow v5, v9 or IPFIX flow records
- Protocol tracker library
  - Detect and track protocols that uses dynamic ports
    - P2P, FTP etc.

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

31



## Compile MAPI



<http://www.ist-lobster.org>

- <http://www.ist-lobster.org/downloads/>
- Makefile.in
  - DEBUG – turns on debug messages
  - WITH\_DAG – adds support for DAG cards
  - WITH\_COMBO6 – adds support for COMBO6 cards
  - WITH\_ADMISSION\_CONTROL – add support for admission control
  - WITH\_FUNCTION\_STATS – turns on counters for all functions to show how many packets where processed by each function
  - WITH\_PRIORITIES – turns on support for flow priorities
  - WITH\_MODIFY\_PKTS – turns on support for functions that modifies packets.
  - WITH\_ANONYMIZATION – compile anonymization library
  - WITH\_TRACKING – compile tracking library
  - WITH\_IPFIX – compile IPFIX library

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

32



## Edit mapi.conf and test



<http://www.ist-lobster.org>

```
libpath=.  
libs=mapidstdflib.so:dagflib.so  
drvpath=.
```

Start mapid: ./mapid

```
[driver]  
device=eth0  
driver=mapinicdrv.so
```

Run a test application:  
cd tests  
./test\_off\_pkt\_counter tracefile

```
[driver]  
device=/dev/dag0  
driver=mapidagdrv.so  
description=DAG 4.3GE  
alias=trd-ntnu
```

```
[format]  
format=MFF_DAG_ERF  
driver=mapidagdrv.so
```

```
[format]  
format=MFF_PCAP  
driver=mapinicdrv.so
```

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

33



## Example MAPI application



<http://www.ist-lobster.org>

- Implement an application that can measure some qualities of a TCP stream
- We want to look at the following attributes:
  - Packets per second
  - Bit per second
  - Jitter
  - TCP congestion window
- Results should be written to a file for further study.
- Syntax:
  - tcpanalyze <filename> <src\_ip> <src\_port>  
<dst\_ip> <dst\_port>

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

34



## Relevant MAPI functions



<http://www.ist-lobster.org>

- PKT\_COUNTER
  - To calculate packets per second
- BYTE\_COUNTER
  - To calculate bit per second
- GAP
  - To measure jitter
- PROTINFO
  - TCP sequence and ack numbers
- BINOP
  - To calculate TCP congestion window
- STAT
  - Calculate statistical values
- RES2FILE
  - To store results to file

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

35



## tcpanalyze.c



<http://www.ist-lobster.org>

- Create a new subdirectory
  - mkdir tcpanalyze
  - cd tcpanalyze
- Create file tcpanalyze.c
- Include necessary headers:

```
#include <stdlib.h>
#include <stdio.h>
#include "mapi.h"
#include "stdlib/pktinfo.h"
#include "stdlib/res2file.h"
#include "stdlib/protinfo.h"
```

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

36



## Create flow and apply BPF filter



<http://www.ist-lobster.org>

```
int main(int argc, char *argv[]) {
    char *file=argv[0];

    int snd=mapi_create_flow("eth0");
    if(snd==-1) {
        printf("Could not create flow\n");
        exit(-1);
    }
    int rec=mapi_create_flow("eth0");

    char filter[1024];
    snprintf(filter,1024,"tcp and src host %s and src port %s and
        dst host %s and dst port %s",
        argv[1],argv[2],argv[3],argv[4]);
    mapi_apply_function(snd,"BPF_FILTER",filter);

    snprintf(filter,1024,"tcp and src host %s and src port %s and
        dst host %s and dst port %s",
        argv[3],argv[4],argv[1],argv[2]);
    mapi_apply_function(rec,"BPF_FILTER",filter);
}
```

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

37



## Get statistical data and start flow



<http://www.ist-lobster.org>

```
int pkts=mapi_apply_function(snd,"PKT_COUNTER");
int bytes=mapi_apply_function(snd,"BYTE_COUNTER");
int gap=mapi_apply_function(snd,"GAP");
int gapstat=mapi_apply_function(snd,"STAT",snd,gap,1);
int seq=mapi_apply_function(snd,"PROTINFO",PI_TCPSEQ);
int ack=mapi_apply_function(rec,"PROTINFO",PI_TCPACK);
int cwnd=mapi_apply_function(snd,"BINOP",BINOP_SUB,
    snd,seq,rec,ack);
int cwndstat=mapi_apply_function(snd,"STAT",snd,gap,1);

char fids[64],types[64];
snprintf(fids,64,"%d@d,%d@d,%d@d,%d@d",
    pkts,snd,bytes,snd,gapstat,snd,cwndstat,snd);
snprintf(types,64,"%d,%d,%d,%d",R2F_ULLSTR,R2F_ULLSTR,
    R2F_STAT,R2F_STAT);
mapi_apply_function(fd,"RES2FILE",fids,types,file,"1s");

mapi_connect(snd);
mapi_connect(rec);
```

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

38



## Makefile



<http://www.ist-lobster.org>

```
SOURCES=$(wildcard *.c)
TARGETS=$(SOURCES:.c=)

CFLAGS=-g -O2 $(C_WARNINGS) -DDEBUG -I..
LDFLAGS= ../mapi.so -lpcap -lpthread

all: $(TARGETS)

%:%.c ../mapi.so ../mapi.h
$(CC) $(CFLAGS) -o $@ $< $(LDFLAGS)

clean:
rm -rf $(TARGETS)
```

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

39



## DiMAPI



<http://www.ist-lobster.org>

- What do you do if you have multiple probes and want to measure the TCP quality at all of them?
- Answer: use DiMAPI!
- DiMAPI is the distributed version of MAPI
- The SCOPE abstraction:
  - SCOPE is a set of lines (interfaces) to monitor
  - **mapi\_create\_flow("host1:eth2, host2:/dev/dag0, host3:eth1");**
  - mapi\_read\_result returns an array of results
- Fully compatible with MAPI
  - All previous functions work as usual

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

40



## Implementing new MAPI functions



<http://www.ist-lobster.org>

- All code in one single source file
- Header file only needed if function returns complex data.
- Script automatically creates source file for libraries
- MAPI function interfaces:
  - instance
  - init
  - process
  - get\_result
  - reset
  - cleanup
  - client\_init
  - client\_get\_result
  - client\_cleanup

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

41



## MAPI function definition



<http://www.ist-lobster.org>

```
typedef struct mapidflib_function_def {
    char* libname;
    char* name;
    char* descr;
    char* argdescr;
    char* devtype;
    mapi_result_method_t restype;
    int shm_size;
    short modifies_pkts;
    mapidflib_optimize_t optimize;
    int (*instance)(....);
    int (*init)(....);
    int (*process)(....);
    int (*get_result)(....);
    int (*change_args)(....);
    int (*reset)(....);
    int (*cleanup)(....);
    int (*client_init)(....);
    int (*client_read_result)(....);
    int (*client_cleanup)(....);
} mapidflib_function_def t;
```

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

42



## PKT\_COUNTER example



<http://www.ist-lobster.org>

```
static mapidflib_function_def_t finfo={
    "", //libname
    "PKT_COUNTER", //name
    "Counts number of packets\n\tReturn value: unsigned long
long", //descr
    "", //argdescr
    MAPI_DEVICE_ALL, //devtype
    MAPIRES_SHM, //Use shared memory to return results
    sizeof(unsigned long long), //shm size
    0, //modifies_pkts
    MAPIOPT_NONE, //global optimization
    NULL, //instance
    NULL, //init
    pktc_process,
    NULL, //get_result,
    pktc_reset,
    NULL, //cleanup
    NULL, //client_init
    NULL, //client_read_result
    NULL //client_cleanup
};
```

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

43



## PKT\_COUNTER example



<http://www.ist-lobster.org>

```
static int pktc_process(
    mapidflib_function_instance_t *instance,
    const unsigned char* dev_pkt,
    const unsigned char* link_pkt,
    mapid_pkthdr_t* pkt_head)
{
    (*(unsigned long long*)instance->result.data)++;
    return 1;
}

static int pktc_reset(
    mapidflib_function_instance_t *instance)
{
    (*(unsigned long long*)instance->result.data)=0;
    return 0;
}
```

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

44



## Current status of MAPI



<http://www.ist-lobster.org>

- Still under development
- Stable if used as intended
- Can segfault if wrong arguments are passed
- Working hard on making it more robust
- Available from:
  - <http://www.ist-lobster.org/downloads/>



## More information?



<http://www.ist-lobster.org>

<http://www.ist-lobster.org>





## Goals

- Major:
  - faster
  - more flexibletraffic processing
- Minor:
  - new networking subsystem
  - use new hardware



H.Bos

Vrije Universiteit Amsterdam



## Difficulties

- hardware:
  - widening memory gap
  - fast photons, slow silicon
- software/OS:
  - copies
  - context switching
  - memory allocation
- demand:
  - need for expensive processing (like IDS)



H.Bos

Vrije Universiteit Amsterdam

## Goals (derived)

- cater to
  - individual packets
  - streams
- domain
  - monitoring
  - transmission
- exploit hardware
- retain what is useful

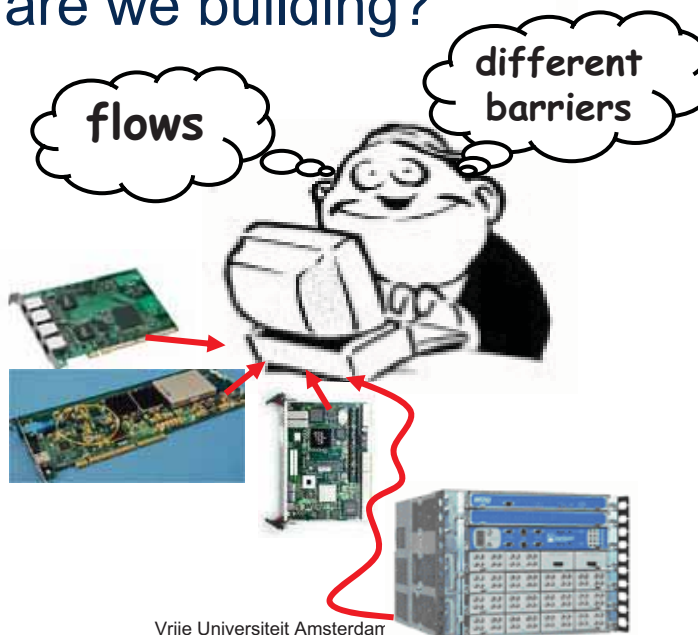


H.Bos

Vrije Universiteit Amsterdam



## What are we building?

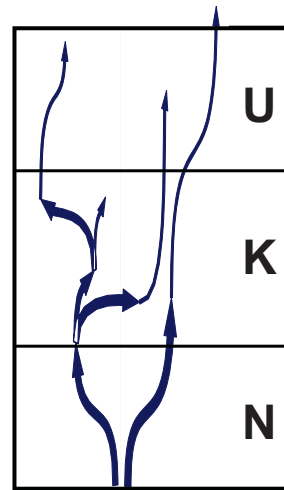


H.Bos

Vrije Universiteit Amsterdam

## Streamline design

- single framework for pkt processing that
  - uses **all levels** in the processing hierarchy
  - is **language neutral**
  - offers advanced **processing in NIC**
  - supports **stateless** and **stateful** filters
  - is backward compatible with **pcap** (while also supporting much more **powerful packet languages**)
  - helps users to build complex monitoring applications by '**clicking components together**'

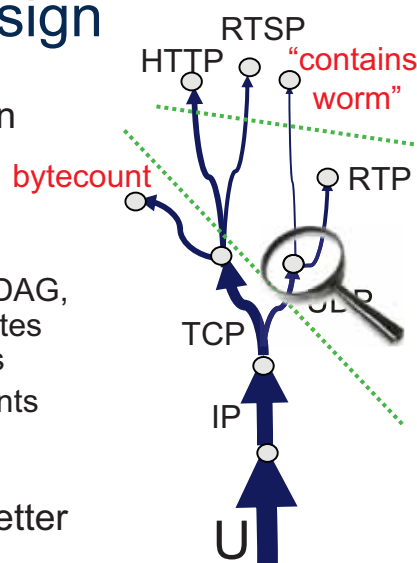


H.Bos

Vrije Universiteit Amsterdam

## Streamline design

- specify processing in terms of a DAG
- Streamline handles instantiation
  - for each function in DAG, Streamline investigates potential placements
  - collective requirements determine definite placement
- Heuristic: lower is better



H.Bos

Vrije Universiteit Amsterdam



# Buffers

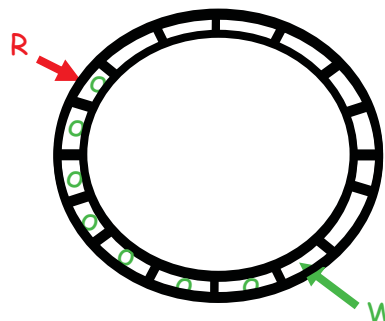
H.Bos

Vrije Universiteit Amsterdam



# Buffers

- PacketBuf
  - circular buffer with N slots
  - e.g., large enough to hold packet
- IndexBuf
  - circular buffer with N slots
  - pointers to packets in PacketBuf

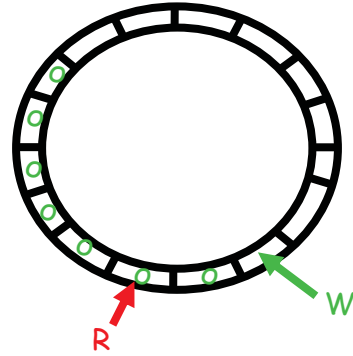


H.Bos

Vrije Universiteit Amsterdam

# Buffers

- PacketBuf
  - circular buffer with N slots
  - e.g., large enough to hold packet
- IndexBuf
  - circular buffer with N slots
  - pointers to packets in PacketBuf

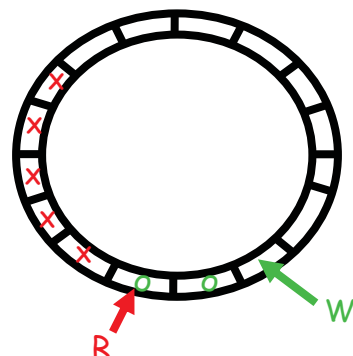


H.Bos

Vrije Universiteit Amsterdam

# Buffers

- PacketBuf
  - circular buffer with N slots
  - e.g., large enough to hold packet
- IndexBuf
  - circular buffer with N slots
  - pointers to packets in PacketBuf



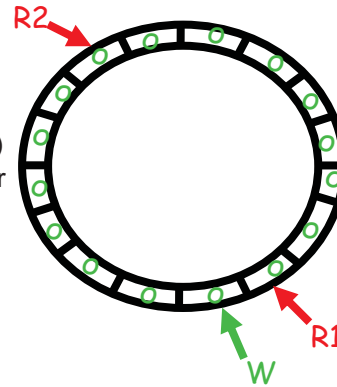
H.Bos

Vrije Universiteit Amsterdam

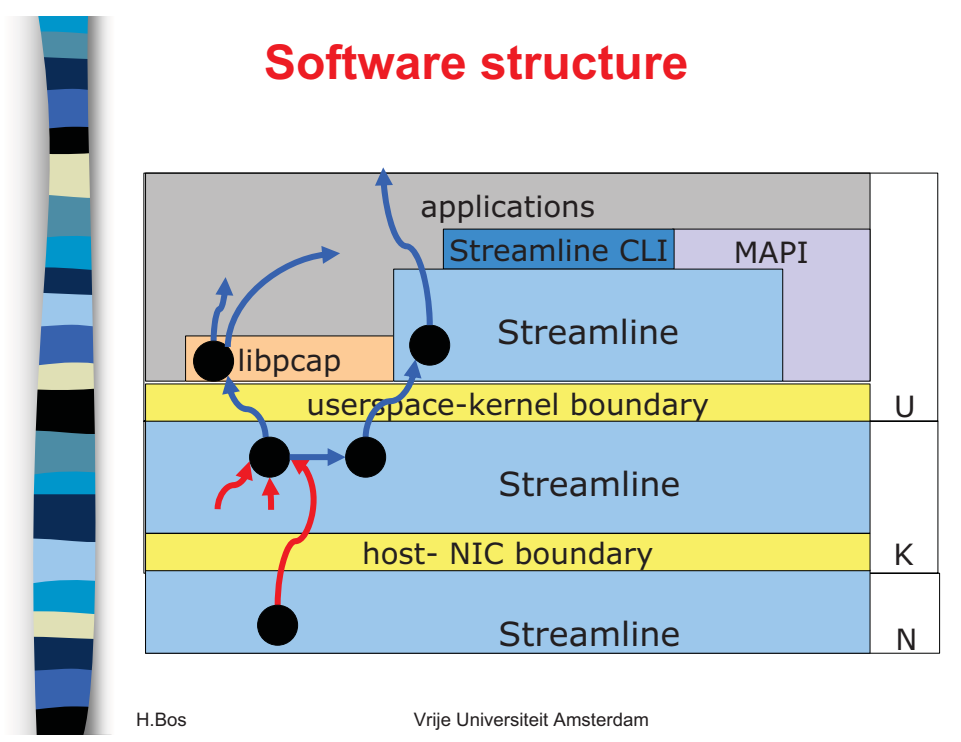
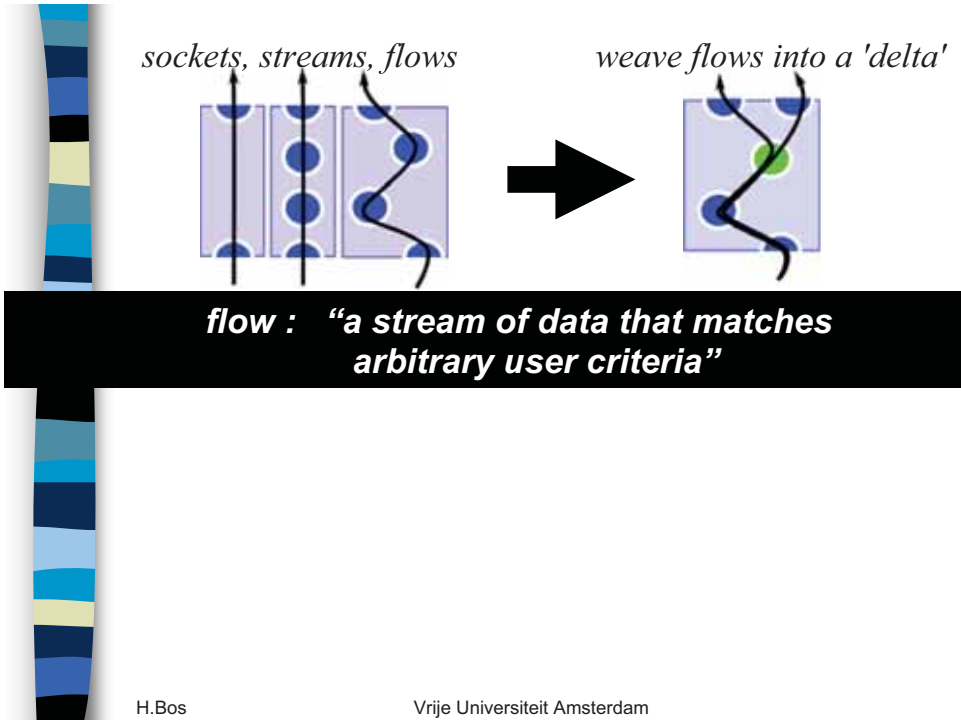
# Buffer management

→ what to do if writer catches up with slowest reader?

- **slow reader preference**
  - drop new packets (traditional way of dealing with this)
  - overall speed set by slowest reader
- **fast reader preference**
  - overwrite existing packets
  - application responsible for keeping up
    - check whether packets have been overwritten
    - different drop rates for different apps



# Basic abstraction: flow





## How to use it?

- we build complex applications out of simple components
- different types of use
  - administrators
    - CLI/GUI: click applications together
    - program filters/functions in high-level languages
  - programmers
    - write code against the API
    - write new functions to be used by Streamline

H.Bos

Vrije Universiteit Amsterdam



## Part I: the administrator

H.Bos

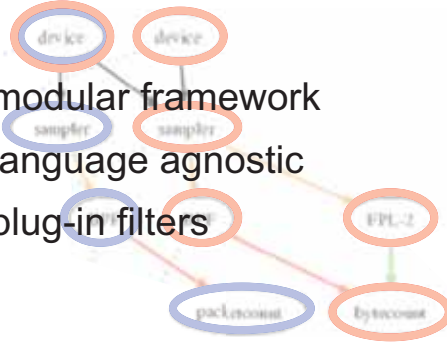
Vrije Universiteit Amsterdam

# Extensible

```
(device,expr=eth0) > (sampler,expr=2) > (BPF,expr="..") > (packetcount)
```

```
[(device,expr=eth0) | (device,expr=eth1)] > (sampler,expr=2) > [(FPL-2,expr="..") | (BPF,expr="..")] > (bytecount)
```

- ✓ modular framework
- ✓ language agnostic
- ✓ plug-in filters



## CLI language

- flowgraph language
  - boolean constructs
    - **a & b**
    - **a|b**
    - **!a**
  - stream:
    - **a -> b**
    - **a -80-> b**
  - grouping:
    - **a -> (b|c)**

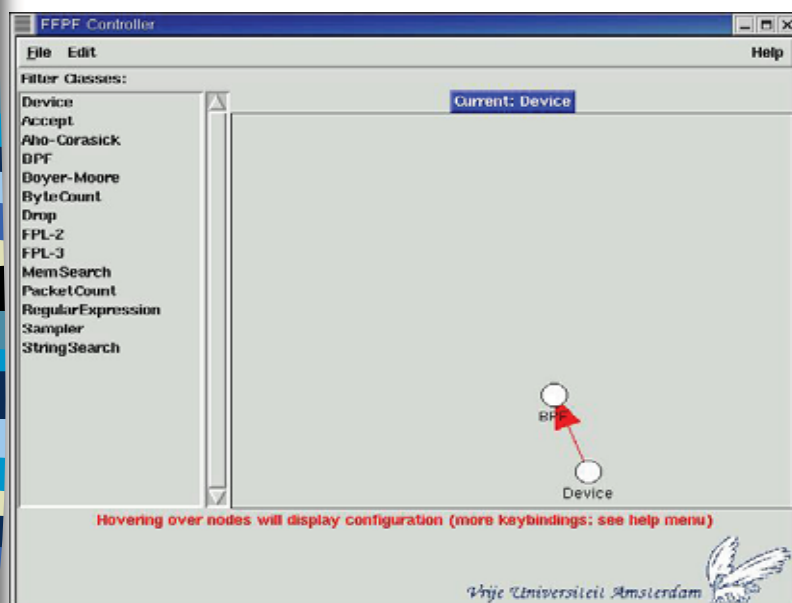
# Example expressions

- **trivial:**  
(generator,expression="1:0") > (accept,export,name=mybuffer)  
  
(pcapin,expression=infectedhttp.pcap) > (packetcount) >  
(sampler,expression=50) > (packetcount)
- **cleaning streams:**  
(pcapin,expression="infectedhttp.pcap") > (tcp) >  
(tcpreassemble) > (ips\_prospector) >  
(outfiles,expression=cleantcp)
- **classification:**  
(pcapin,expression="infectedhttp.pcap") > (sourceport)  
-80-> (tcp) > (tcpreassemble) > (inspect)
- **transcoding:**  
(ipv4\_receive,expression="t 127.0.0.1 5050") > (rot13) >  
(ipv4\_transmit,expression="t 127.0.0.1 5051")

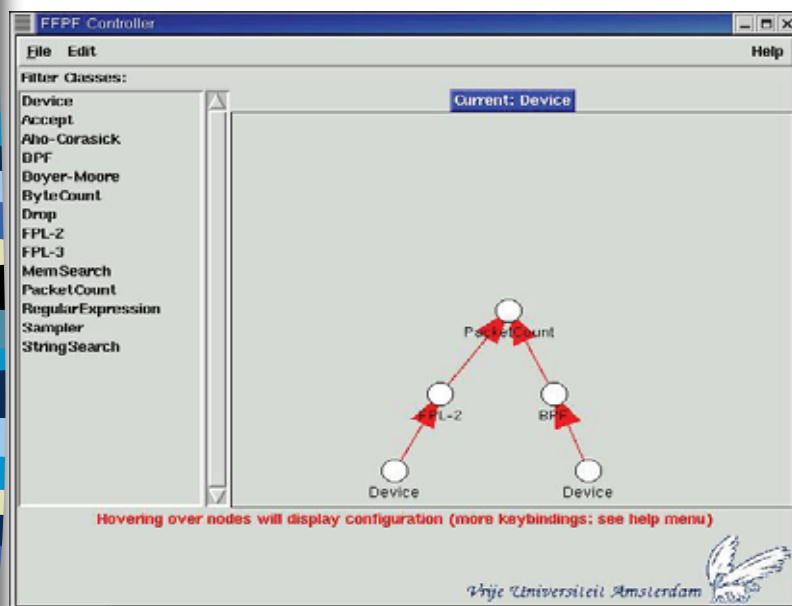
H.Bos

Vrije Universiteit Amsterdam

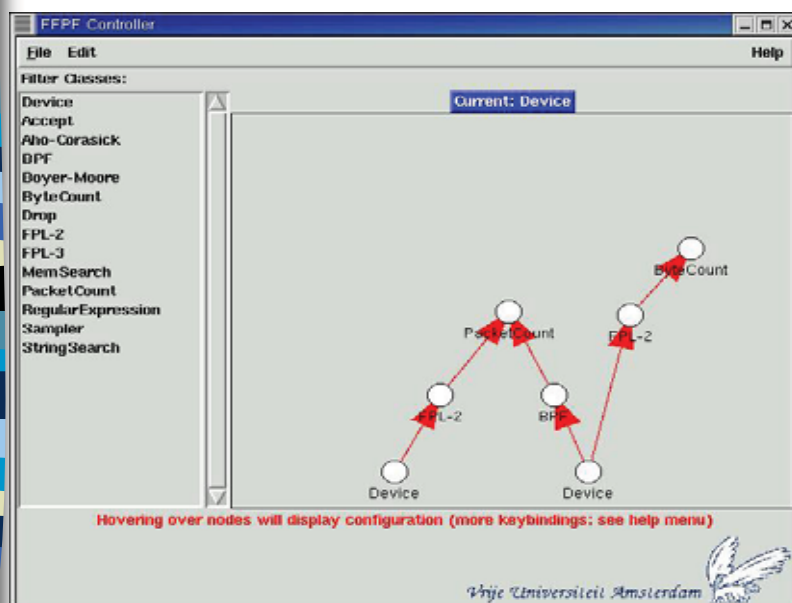
## Example 1: GUI for creating flowgraphs



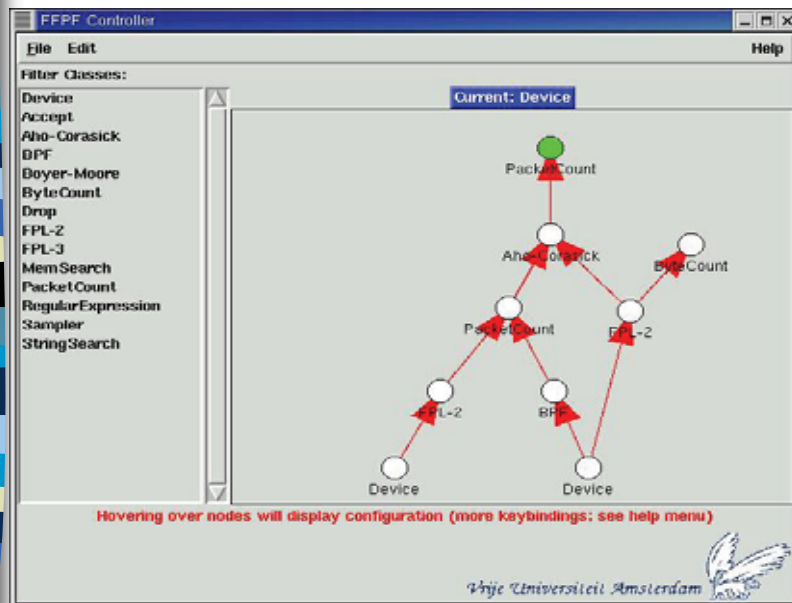
## Example 1: GUI for creating flowgraphs



## Example 1: GUI for creating flowgraphs



## Example 1: GUI for creating flowgraphs



## mapping on hardware

Streamline is responsible for mapping the flowgraph on the underlying hardware



## Copying

### Three flavours of packet processing

- **Regular**
  - copy only when needed
  - may be slow depending on access pattern
- **Zero copy**
  - *never* copy
  - may be slow depending on access pattern
- **Copy once**
  - copy always

H.Bos

Vrije Universiteit Amsterdam



## Part II: the savvy administrator

H.Bos

Vrije Universiteit Amsterdam

## Example 2: FPL-3

- new pkt processing language: FPL-2

- for IXP, kernel and userspace
- simple, efficient and flexible
- simple example: filter all webtraffic

```
IF ( (PKT.IP_PROTO == PROTO_TCP)
      && (PKT.TCP_PORT == 80) ) THEN RETURN 1;
```

- more complex example: count pkts in all TCP flows

```
IF (PKT.IP_PROTO == PROTO_TCP) THEN
  R[0] = Hash[ 14, 12, 1024];
  M[ R[0] ]++;
FI
```

## FPL-3

- all common arithmetic and bitwise operations
- all common logical ops
- all common integer types
  - for packet
  - for buffer (← useful for keeping state!)
- statements
  - Hash
  - External functions
    - to call hardware implementations
    - to call fast C implementations
  - If ... then ... else
  - For ... break; ... Rof
  - Return



## Example application: dynamic ports

```
1. // R[0] stores no. of dynports found (initially 0)
2. IF (PKT.IP_PROTO==PROTO_TCP) THEN
3.   IF (PKT.TCP_DPORT==554) THEN
4.     M[R[0]]=EXTERN("GetDynTCPDPortFromRTSP",0,0);
5.     R[0]++;
6.   ELSE // compare pkt's dst port to all ports in array – if match, return pkt
7.     FOR (R[1]=0; R[1] < R[0]; R[1]++)
8.       IF (PKT.TCP_DPORT == M[R[1]]) THEN
9.         RETURN TRUE;
10.      FI
11.     ROF
12.   FI
12. RETURN FALSE;
```

H.Bos

Vrije Universiteit Amsterdam



## Other example: Ruler

- regular expression matching and rewriting language
  - ➔ see the Lobster workshop tomorrow

H.Bos

Vrije Universiteit Amsterdam



## Part III: the programmer

H.Bos

Vrije Universiteit Amsterdam

### trivial example

```
1. int main (int argc, char** argv){
2.     int fd, bd, len;
3.     char *pkt;
4.
5.     // open FFPF and start processing
6.     streamline_init();
7.     fd=streamline_request_insert("(generator,expression=\"1:0\") > (packetcount, export, name=mybuf)");
8.
9.     // access the data
10.    bd = open_stream("mybuf", O_RDONLY | OF_INDIRECT); // extended, but posix compatible API
11.    while (1) {
12.        sleep(1);
13.        len = read_stream(bd, &pkt, 1500); // direct data access
14.        printf("read a packet of %d bytes\n", len);
15.    }
16.    close_stream(bd);
17.
18.    // close FFPF
19.    streamline_request_remove(fd);
20.    streamline_exit();
21.    return 0;
22. }
```

## Concluding remarks

- achieved: fast flexible packet processing
- minimising copying/context switching
- cater to different users
- exploit advanced hardware
- new languages

H.Bos

Vrije Universiteit Amsterdam

## more information

<http://ffpf.sourceforge.net/>





## An overview of traffic analysis using NetFlow

**Arne Øslebø**  
UNINETT  
[Arne.Oslebo@uninett.no](mailto:Arne.Oslebo@uninett.no)



## Outline

- What is Netflow?
- Available tools
- Collecting
- Processing
  - Detailed analysis
    - security incidents
  - Long term statistics
    - network trends
    - accounting
- Presentation
  - Stager



## What is NetFlow?



<http://www.ist-lobster.org>

- Cisco technology
  - 1996
- IPFIX definition:
  - A set of IP packets passing an observation point in a network during a certain time interval. All packets belonging to a particular flow have a set of common properties.
- Flow Key
  - Each of the properties that are used for defining a flow
- Flow record
  - Measured properties of a flow

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

3



## NetFlow v5



<http://www.ist-lobster.org>

- |                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"><li>• Flow key:<ul style="list-style-type: none"><li>– Source IP address</li><li>– Destination IP address</li><li>– Source port</li><li>– Destination port</li><li>– Layer 3 protocol type</li><li>– TOS</li><li>– Input interface</li></ul></li></ul> | <ul style="list-style-type: none"><li>• Flow record<ul style="list-style-type: none"><li>– Source and destination IP address</li><li>– Next hop router's IP address</li><li>– Input and output interface index</li><li>– Packets and bytes in the flow</li><li>– sysUptime at start and end of flow</li><li>– TCP/UDP source and destination port number</li><li>– Type of service</li><li>– TCP flags</li><li>– IP protocol</li><li>– Source and destination AS number</li><li>– Source and destination address prefix mask bits</li></ul></li></ul> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

4

- flow-tools
  - Stager
  - FlowViewer
  - FlowScan
  - CUFlow
- nfdump
  - nfsen
  - *Stager*
- flowd
- flamingo
- JKFlow
- <http://www.switch.ch/tf-tant/floma/software.html>



- For example: flow-capture or nfcapd
- Files rotated every  $n$  minutes
- Script can be started when file is rotated
- flow-capture supports gzip
  - 2:1 compression ratio
- UNINETT collects:
  - 27 routers, 207 interfaces
  - sampled netflow – 1:100 sampling rate
  - >30GB every day



## Process NetFlow – general statistics



<http://www.ist-lobster.org>

- flow-tools
  - flow-cat, flow-filter, flow-nfilter and flow-stat
- nfdump
- src/dst IP, src/dst AS, src/dst port, protocol etc.

```
flow-cat ft-v05.2006-05-12.08* | flow-filter -i1 |
flow-stat -f12 -S2
#
# protocol  flows                octets                packets
#
6           378694                5668763771           8060656
17          541505                352973483            1745931
50          1578                  172325976            424695
47          641                   55589150             165236
1           14176                 5970023              69637
2           30                    2124                  59
103         28                    1064                  28
41          4                     464                   4
info@ist-lobster.org           Arne Øslebø, UNINETT           7
```



## Process NetFlow – detailed analysis



<http://www.ist-lobster.org>

- Find traffic to/from specific IP address
- Example is anonymized

```
nfdump -r 2006-05-12.0800.nfdump -o 'fmt:%td
%sa %da %byt' 'ip 166.81.189.132'
```

Duration	Src IP Addr	Dst IP Addr	Bytes
0.952	212.187.175.58	166.81.189.132	552
0.952	166.81.189.132	212.187.175.58	6563
0.180	166.81.189.132	190.88.151.169	468
0.184	190.88.151.169	166.81.189.132	1446
0.524	68.129.203.157	166.81.189.132	536
0.520	166.81.189.132	68.129.203.157	762
1.048	192.127.132.177	166.81.189.132	2569
1.044	166.81.189.132	192.127.132.177	102321



- Who sendt the most traffic to this IP address?

```
nfdump -r 2006-05-12.0800.nfdump -n5 -ssrcip/bytes 'dst ip 166.81.189.132'
```

Top 5 Src IP Addr ordered by bytes:

Duration	Proto	Src IP Addr	Flows	Packets	Bytes
910.083	any	190.88.151.169	152	976	180764
904.953	any	166.81.191.137	14	655	117439
902.915	any	139.65.84.134	172	1413	106792
848.974	any	192.127.132.177	35	1253	92450
876.597	any	166.81.98.117	36	364	56876



- Long term statistics
- Easy to use web frontend
  - Text based reports and graphs
- Support for different types of network statistics
  - Netflow
  - *SNMP*
  - *Mping*
- Easy to add new reports
  - Templates and plugins
- Access control
  - Observation points and reports

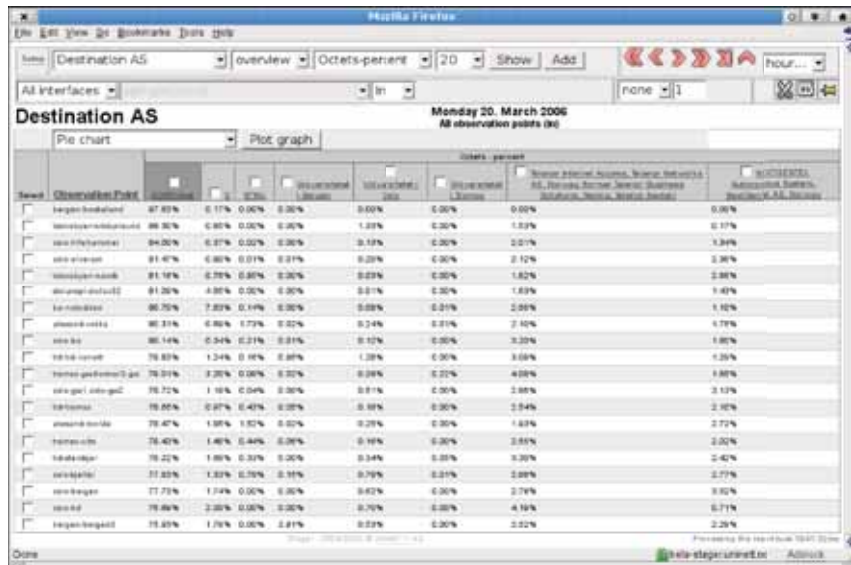
<http://software.uninett.no>



## Overview reports



<http://www.ist-lobster.org>



info@ist-lobster.org

Arne Øslebø, UNINETT

11



## Backend robustness



<http://www.ist-lobster.org>

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

info@ist-lobster.org

Arne Øslebø, UNINETT

12

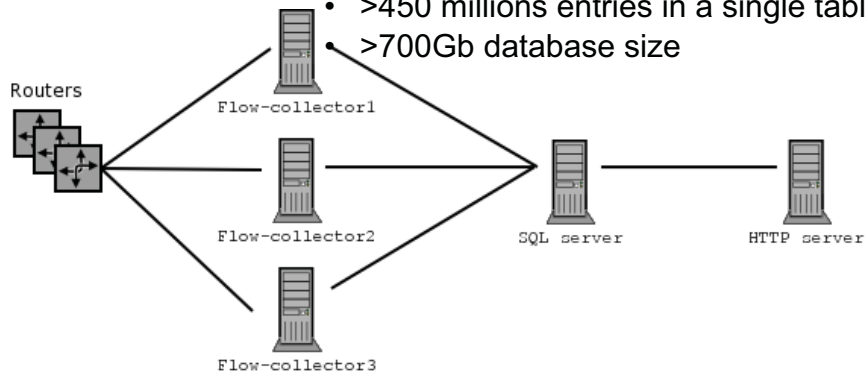


## Our NetFlow setup



<http://www.ist-lobster.org>

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



[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

13



## Performance



<http://www.ist-lobster.org>

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

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

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

[info@ist-lobster.org](mailto:info@ist-lobster.org)

Arne Øslebø, UNINETT

14



## Some links



<http://www.ist-lobster.org>

- <http://www.switch.ch/tf-tant/floma/software.html>
- <http://software.uninett.no/>
- <http://www.splintered.net/sw/flow-tools/>
- <http://nfdump.sourceforge.net/>
- <http://www.ietf.org/html.charters/ipfix-charter.html>
- [http://www.cisco.com/en/US/products/ps6601/products\\_ios\\_protocol\\_group\\_home.html](http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html)