



Hardware anonymization

Sven Ubik, Petr Zejdl, Vladimir Smotlacha

TNC-2006, Catania, 16.5.2006



Passive monitoring



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

Problem: access to sensitive information in payload and headers.

- corporation data
- network management
- personal data
 - private data (credit card numbers, accounts, passwords)
 - activities of person, used services
 - favorite URLs

Restriction in data processing

- administrative
- technical



Anonymization

Traditional anonymization rule (NLANR):

- source IP address is encrypted
- destination IP packet is encrypted
- payload is completely removed



Anonymization methods

Balance: privacy x useful information

- encryption
 - no information is lost, controlled access
- hashing
 - information is lost, can be compared
- randomization, clearing
 - data are lost



Software x hardware

Software approach

- full control
- easily configurable
- flexible (new methods)

Hardware approach

- real data only on card
 - **trustworthy processing**
 - no access for system operator
- saves CPU capacity



Perfect monitoring adapter

- 2 or more inputs, wirespeed
- classification of packets
 - static
 - dynamic
- on-board processing -> less data on system bus
 - packet truncation
 - packet sampling
 - pattern searching
 - calculation of statistics
- anonymization



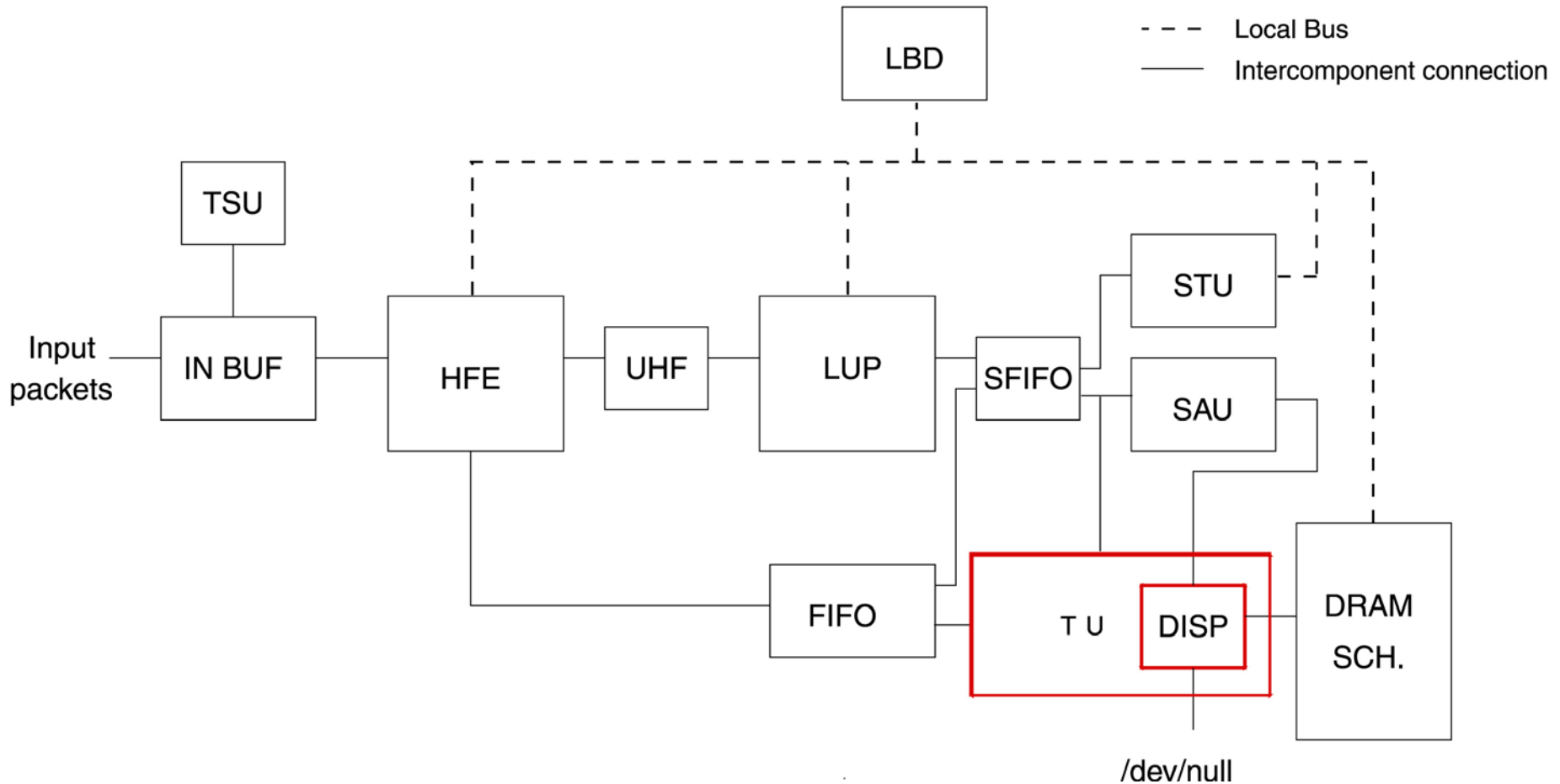
On-board anonymization

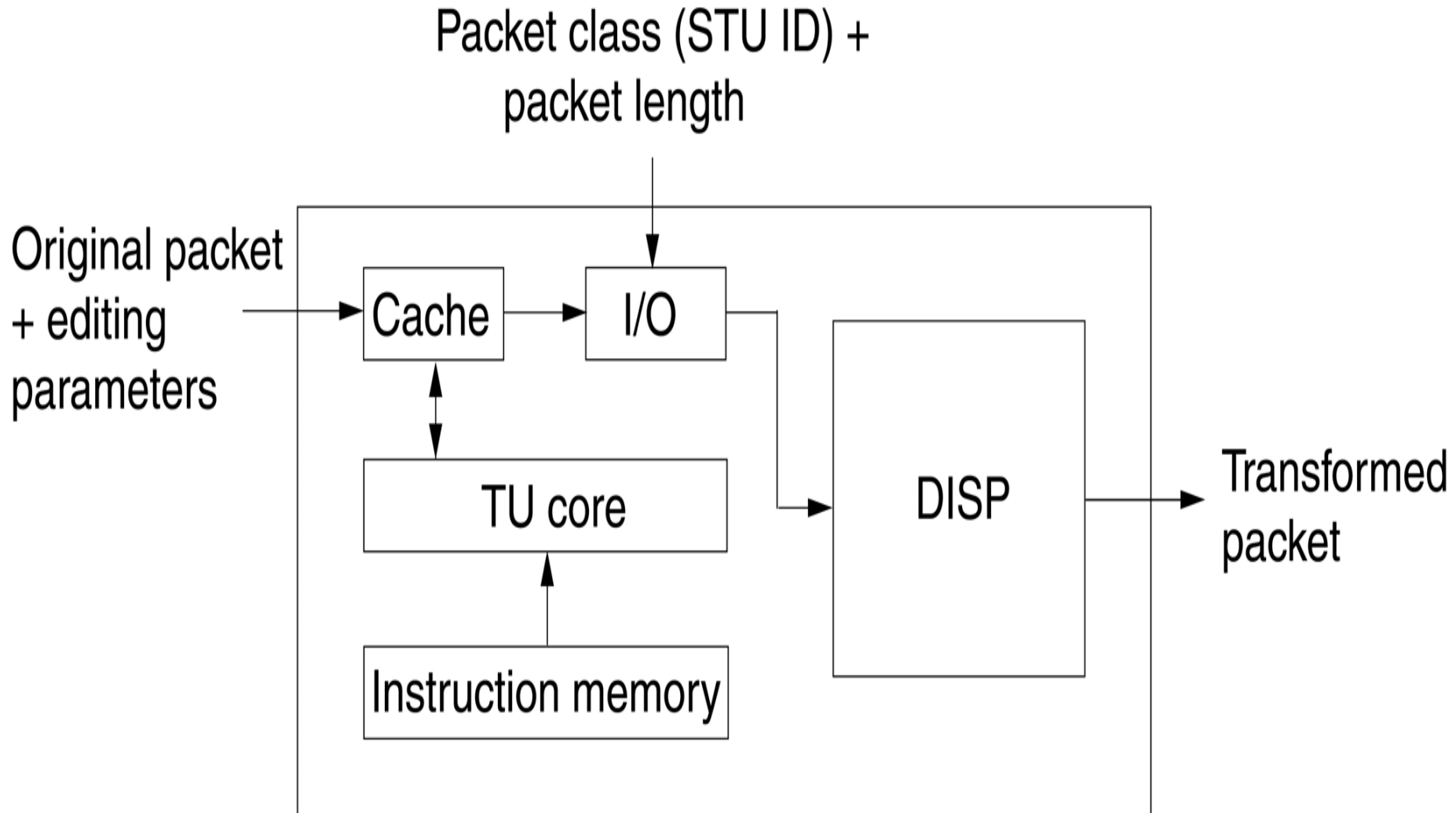
On-board anonymization in Lobster:

- COMBO6 card + Scampi monitoring firmware
- programmable Transformation Unit (TU)
- TU is implemented as „nanoprocessor“
- up 256 different classes of packets



Position of TU in firmware







Transformations

Transformations that can be applied to header fields:

- set to a specified constant
- xor with a specified constant
- set to a pseudorandom number
- hash using map table
- prefix-preserving IP address mapping

- any combination of the above (e.g., first half of IP address set to constant, second half randomized)



TU ports (I/O signals)

-- SFIFO interface

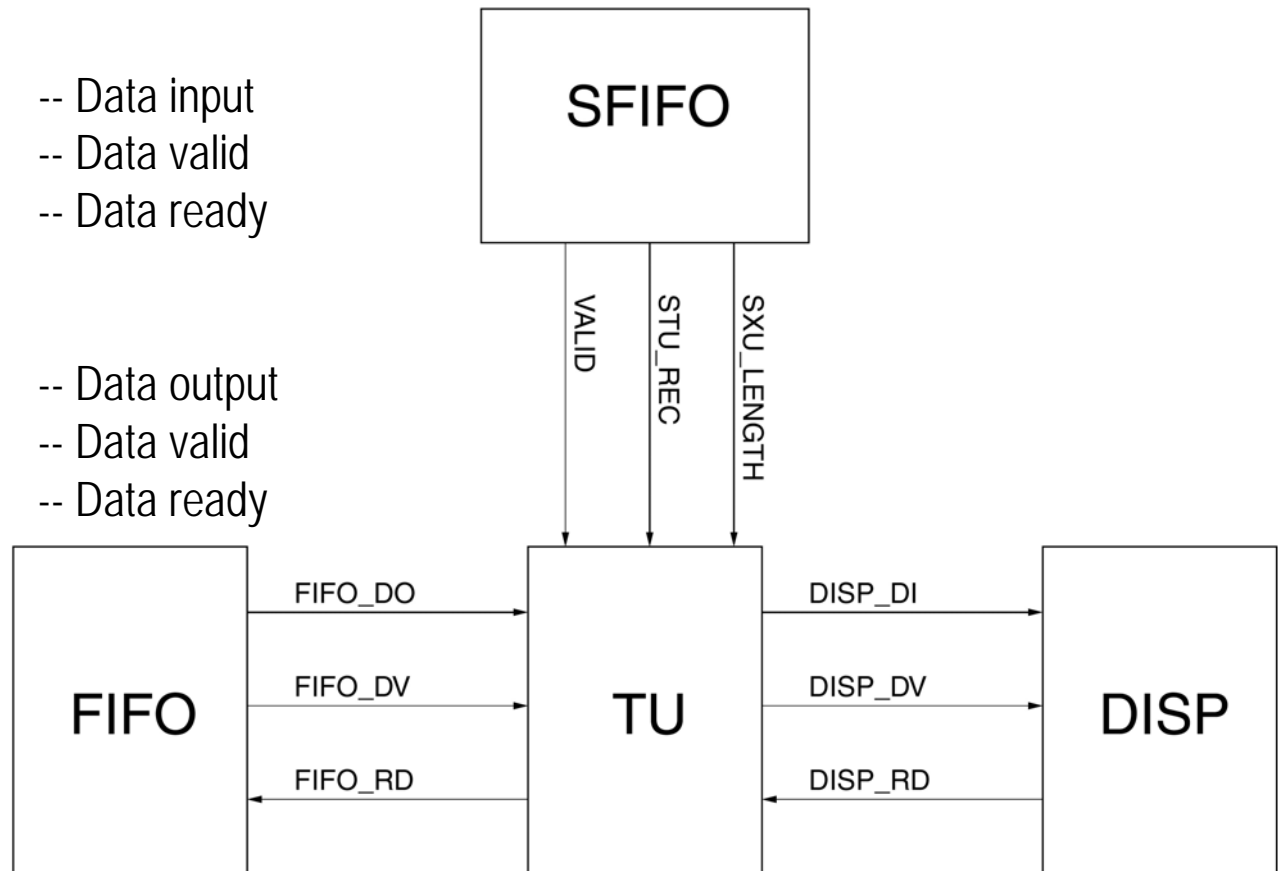
TU_STU_REC : in 8 bits -- ID STU
TU_SXU_LENGTH : in 14 bits -- Paket length
TU_STU_VALID : in -- Data valid

-- FIFO interface

TU_FIFO_DO : in 16 bits -- Data input
TU_FIFO_DV : in -- Data valid
TU_FIFO_RD : out -- Data ready

-- DISP interface

TU_DISP_DI : out 16 bits -- Data output
TU_DISP_DV : out -- Data valid
TU_DISP_RD : in -- Data ready



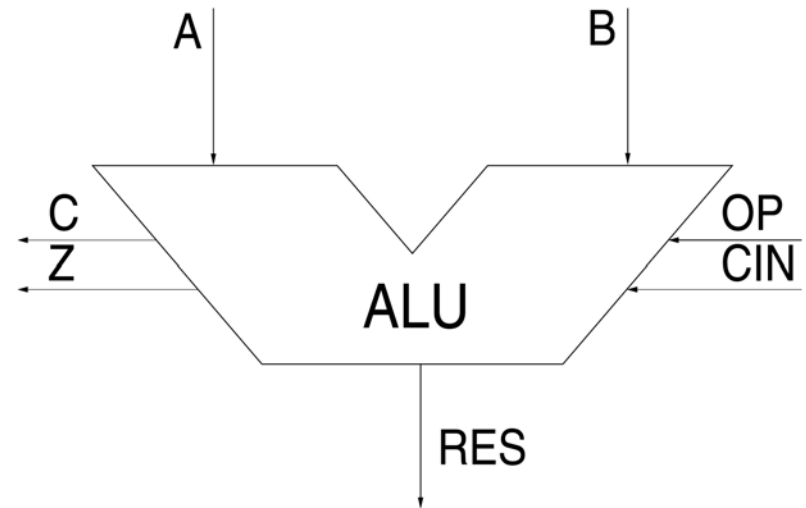


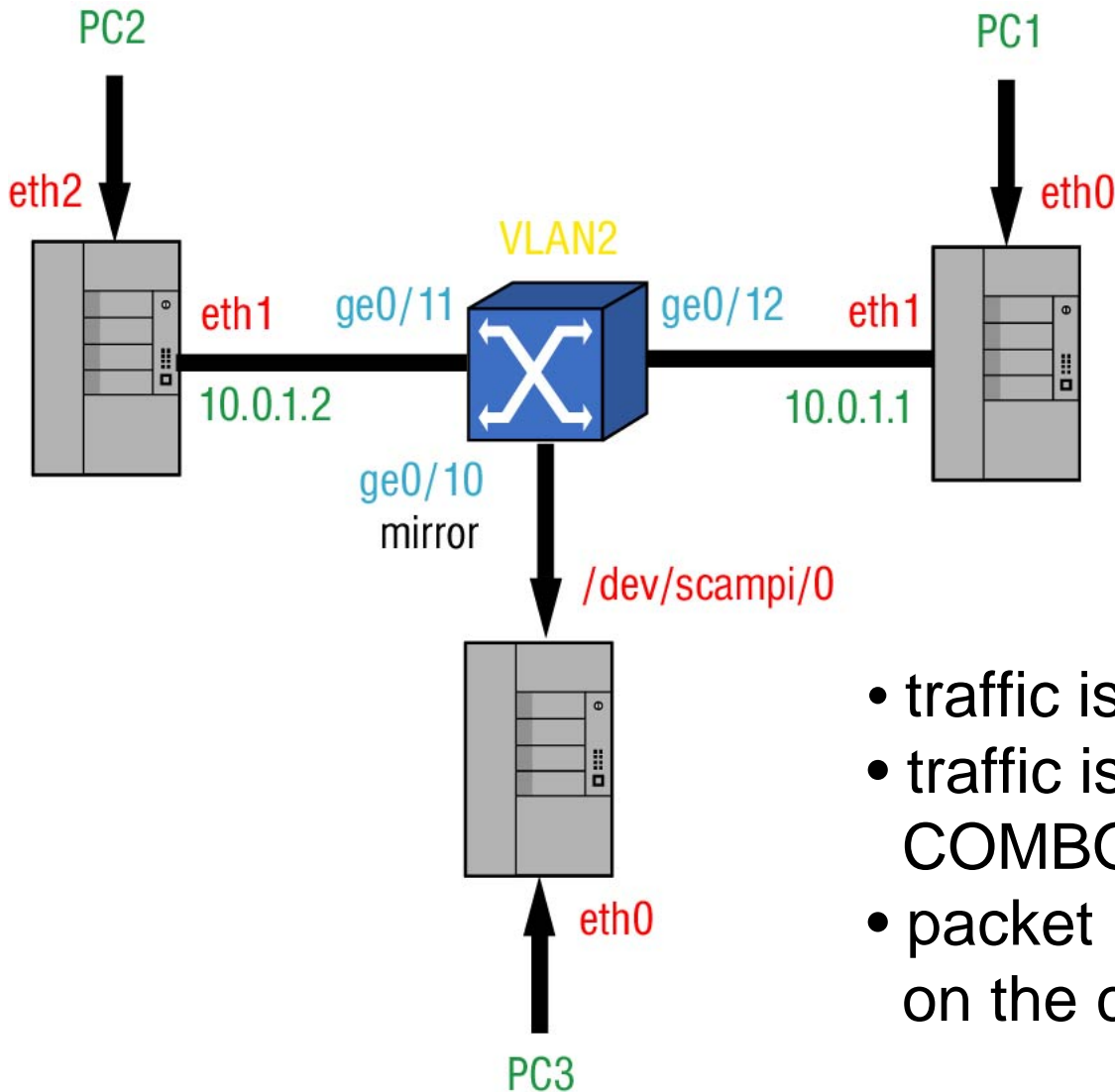
ALU signals and operation

A : in 16 bits -- operand A
B : in 16 bits -- operand B
OP : in -- operation code
CIN : in -- carry flag input
RES : out 16 bits -- result
Z : out -- zero flag output
C : out -- carry flag output

Operation code:

opADC -- addition with carry
opSBB -- subtraction with carry
opAND -- logical and
opOR -- logical or
opXOR -- logical xor
opSHL -- logical shift to left
opSHR -- logical shift to right
opSWP -- low/high 8-bit swap





- traffic is sent from PC1 to PC2
- traffic is mirrored to PC3 with COMBO card
- packet headers are anonymized on the card

- Packets pass through TU in 16-bit chunks per clock cycle (50 MHz clock)

1500-byte packets	65703 packets/s	788 Mb/s
64-byte packets	1162790 packets/s	595 Mb/s

- Current bottleneck is the classification unit
-> wire speed classification in Combo6X



A part of new libfilter library:

```
scampi_reset_classification(int mode);    /* DIRECT or COMPLEX */
scampi_compile_filter(int filter_id, char *filter, u_int16 sau_mask,
    u_int8 stu_id, u_int8 pck_mask);
int scampi_set_sau(int sau_id, sau_modes_t mode, u_int32_t init,
    u_int32_t threshold, int enable)
scampi_compile_pck(char *searched_string, int cam_line, u_int_8 pck_mask);
scampi_compile_transformation (int filter_id, int protocol,
    int field, int function, ...);
scampi_print_filters();
scampi_set_filters();
```



Thank you for your attention