traffiController^{4SpW} Technical Presentation

14th SpW Working Group Meeting,
February 23rd, 2010.

Stéphane Davy, Matthieu Salanave
spacewire@skylab-corporate.com

Dr Slim Ben Saoud
www.asic-international.com

SKYLAB Industries

Outline

1. Introduction
2. TraffiController^{4SpW} DVS
3. TraffiController^{4SpW} API
4. TraffiController^{4SpW} GUI
5. Conclusion and questions

SKYLAB Industries

Outline

1 Introduction

2 TraffiController^{4SpW} DVS

3 TraffiController^{4SpW} API

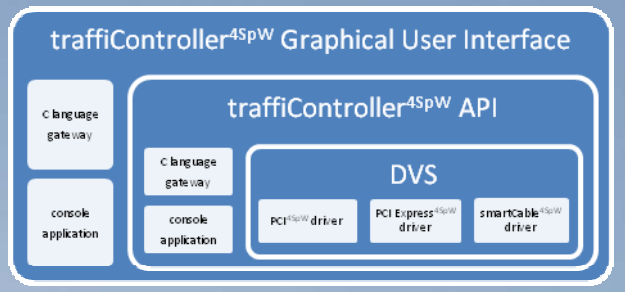
4 TraffiController^{4SpW} GUI

5 Conclusion and questions

SKYLAB Industries

3 Java software layers, with a gateway to C language

- Devices Virtualization Service
- Application Programming Interface
- Graphical User Interface



Multiple functions in one single software package

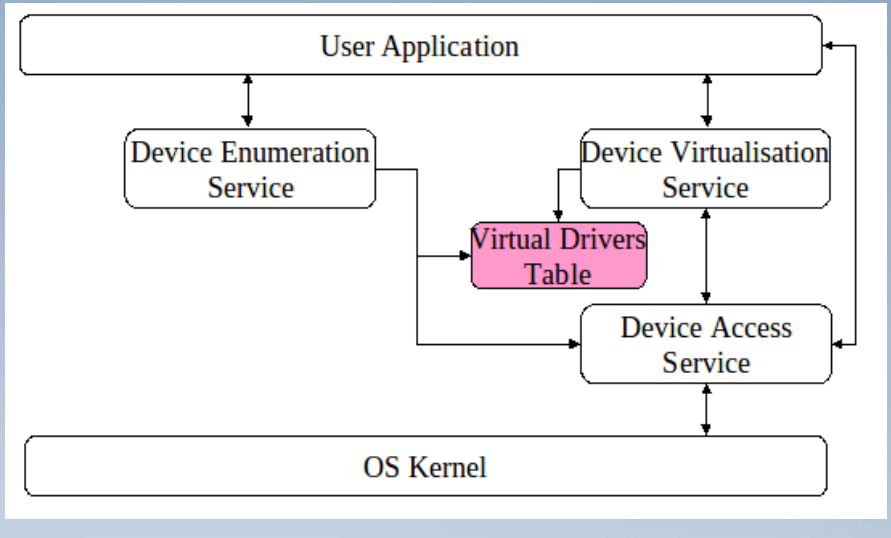
- SKYLAB test hardware management, data communication and control
- Integration and test, simulation and network dimensioning thanks to:
  - virtual routers,
  - RMAP support,
  - IP/Ethernet tunnels,
  - real, emulated or virtual nodes,
  - software implementation of SpW link analyzers,
  - communication statistics,
  - etc…

Outline

1. Introduction
2. TraffiController^{4SpW} DVS
3. TraffiController^{4SpW} API
4. TraffiController^{4SpW} GUI
5. Conclusion and questions

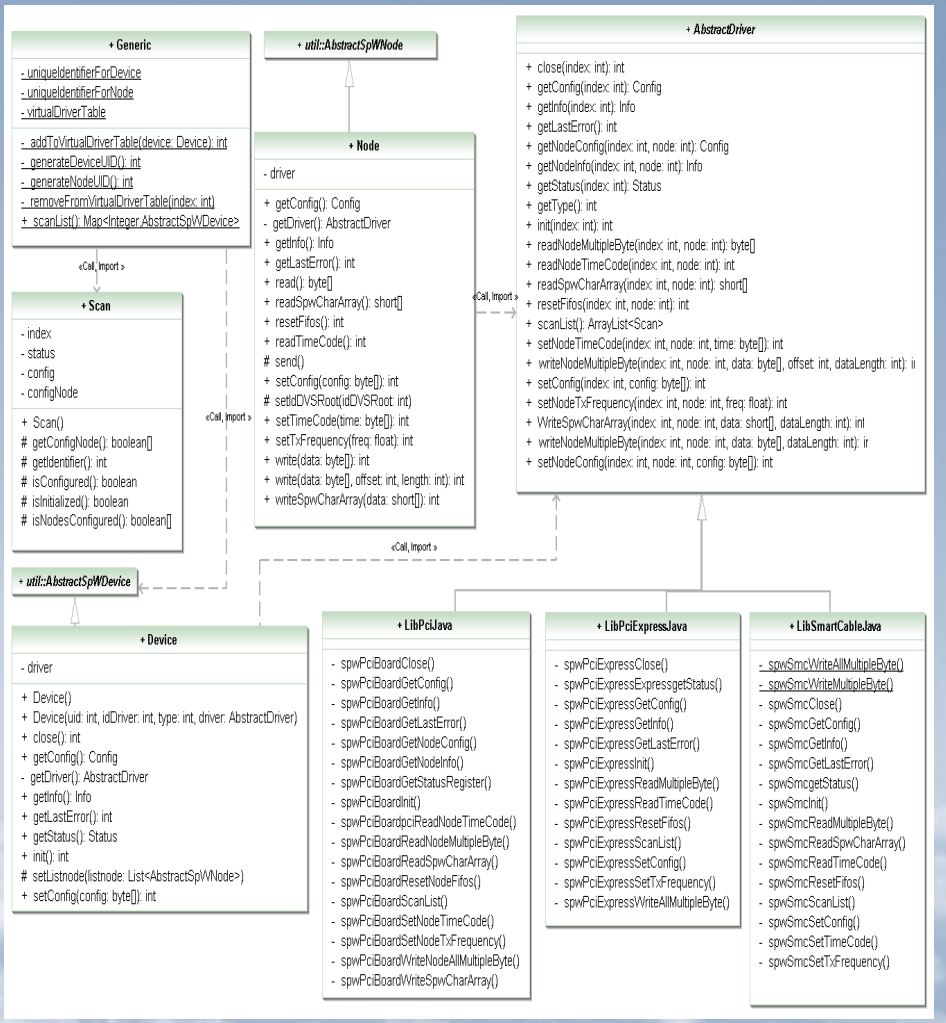
SKYLAB Industries

DES/DVS/DAS (Devices Virtualization Service) overview



DVS main functions

- Enumeration and update of hardware
- Management of SpW devices, nodes
- Driver(s) routines access service
- *ID_DVS* listing



DVS object-oriented architecture

DVS list of routines

DVS_4SpW_scanlist.
DVS_4SpW_get_device.
DVS_4SpW_Init.
DVS_4SpW_close.
DVS_4SpW_set_config.
DVS_4SpW_get_Info.
DVS_4SpW_get_config.
DVS_4SpW_get_status_register.
DVS_4SpW_get_node. DVS_4SpW_set_node_config.
DVS_4SpW_set_node_tx_frequency.
DVS_4SpW_set_node_time_code.
DVS_4SpW_reset_node_fifo.
DVS_4SpW_get_node_Info.
DVS_4SpW_read_node_time_code.
DVS_4SpW_read_SIP. DVS_4SpW_read_SpW.
DVS_4SpW_write_SIP. DVS_4SpW_write_SpW

DVS scan example

```
import util.AbstractSpWDevice;
import util.AbstractSpWNode;
import dvs.Generic;
public class Test {
public static Map<Integer, AbstractSpWDevice> listDevice = new TreeMap<Integer, AbstractSpWDevice>();
public static void main(String[] args) {
listDevice = Generic.scanList();
boolean exist = false;
System.out.println("|");
System.out.println("|__[List of SMARTCABLE devices]");
for (AbstractSpWDevice device : listDevice.values())
if (device.getType() == 1) {
 exist = true;
 System.out.printf("|   |__SMC %d : IDDVS [0x%04x] : Initialized [%b] : Configured [%b]\n",
      device.getIdDriver(), device.getUID(), device.isInitialized(), device.isConfigured());
 if (device.getListnode() != null)
      for (AbstractSpWNode node : device.getListnode())
System.out.printf("|   |    |__Node %d : IDDVS [0x%04x] : Initialized [%b] : Configured [%b]\n",
           node.getNumNode(), node.getUidDvs(),node.isInitialized(), node.isConfigured());
}
```

DVS console

```
[Test Device]
        ¦__(e) Enumerate Devices
           (i) Init
           (c) Config
           (g) Get Info
           (d) Command Devices
           (x) Exit
Enter your choice :
```

```
¦__[List of SMARTCABLE devices]
¦    ¦__SMC 0 : IDDVS [0x0000] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 0 : IDDVS [0x0000] : Initialized [false] : Configured [false]
¦    ¦__SMC 1 : IDDVS [0x0001] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 0 : IDDVS [0x0001] : Initialized [false] : Configured [false]
¦    ¦__SMC 2 : IDDVS [0x0002] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 0 : IDDVS [0x0002] : Initialized [false] : Configured [false]
¦    ¦__SMC 3 : IDDVS [0x0003] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 0 : IDDVS [0x0003] : Initialized [false] : Configured [false]
¦    ¦__SMC 4 : IDDVS [0x0004] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 0 : IDDVS [0x0004] : Initialized [false] : Configured [false]
¦    ¦__SMC 5 : IDDVS [0x0005] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 0 : IDDVS [0x0005] : Initialized [false] : Configured [false]
¦
¦__[List of PCI devices]
¦    ¦
¦    ¦__PCI 0  : IDDVS [0x0006] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 0 : IDDVS [0x0006] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 1 : IDDVS [0x0007] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 2 : IDDVS [0x0008] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 3 : IDDVS [0x0009] : Initialized [false] : Configured [false]
¦    ¦
¦    ¦__PCI 1  : IDDVS [0x0007] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 0 : IDDVS [0x000a] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 1 : IDDVS [0x000b] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 2 : IDDVS [0x000c] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 3 : IDDVS [0x000d] : Initialized [false] : Configured [false]
¦    ¦
¦    ¦__PCI 2  : IDDVS [0x0008] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 0 : IDDVS [0x000e] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 1 : IDDVS [0x000f] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 2 : IDDVS [0x0010] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 3 : IDDVS [0x0011] : Initialized [false] : Configured [false]
¦    ¦
¦    ¦__PCI 3  : IDDVS [0x0009] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 0 : IDDVS [0x0012] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 1 : IDDVS [0x0013] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 2 : IDDVS [0x0014] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 3 : IDDVS [0x0015] : Initialized [false] : Configured [false]
¦    ¦
¦    ¦__PCI 4  : IDDVS [0x000a] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 0 : IDDVS [0x0016] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 1 : IDDVS [0x0017] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 2 : IDDVS [0x0018] : Initialized [false] : Configured [false]
¦    ¦     ¦__Node 3 : IDDVS [0x0019] : Initialized [false] : Configured [false]
¦
¦__[List of PCI Express devices]
¦    ¦__NONE
```

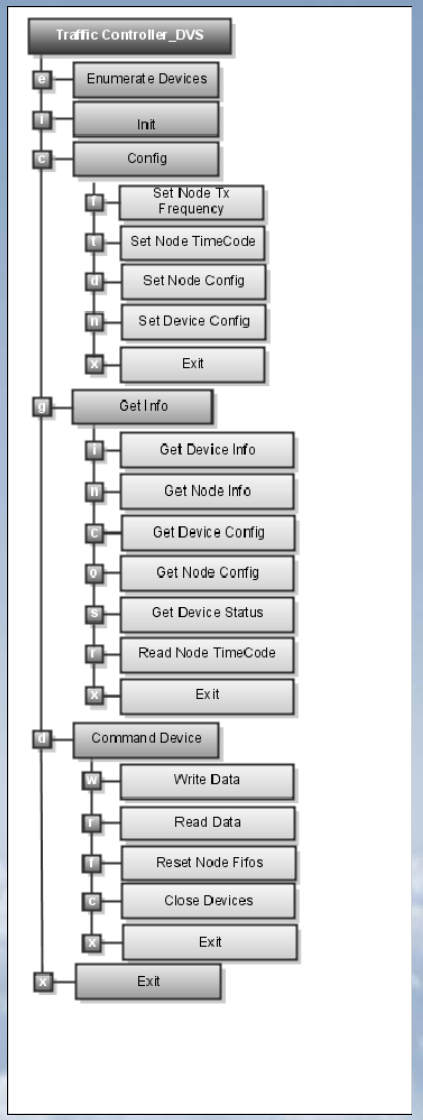
```
[Test Device]
       ¦__[Config]
                ¦__(f) Set Node Tx Frequency
                   (t) Set Node TimeCode
                   (n) Set Node Config
                   (d) Set Device Config
                   (x) Exit
Enter your choice : f

For which node do you want to set Frequency?[node ID --> IDDVS]
12

============================== Result ========================================
Enter SpW CODEC transmission frequency value between 2Mhz and 200Mhz for node 0x0012 :
150

Setup SpW node 0 frequency of PCI Board 3 to 150.0MHz.
NO ERROR

==============================================================================
```



Outline

1. Introduction
2. TraffiController^{4SpW} DVS
3. TraffiController^{4SpW} API
4. TraffiController^{4SpW} GUI
5. Conclusion and questions

SKYLAB Industries

Application Programming Interface (API) overview

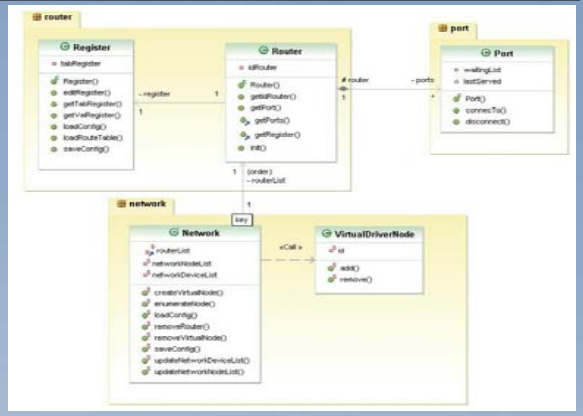
- SpW networking emulation and command control based on the following objects:
  - Router, node, ports objects
  - RMAP engine, multiple instances
  - *Ethernet communication for API dialogues* engine, multiple instances
  - Link streams for Virtual protocol analyzer engine, multiple instances

API main functions

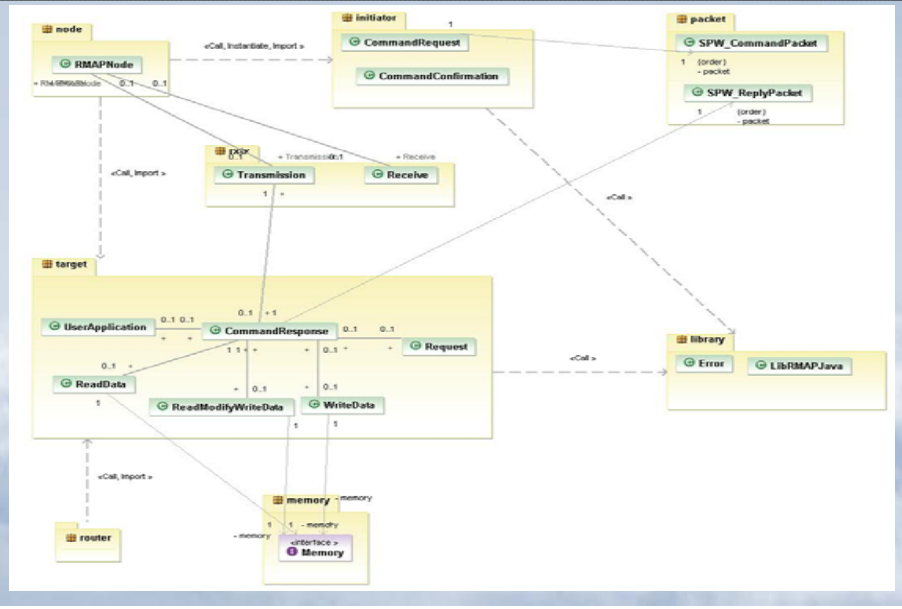
- SpW networking using virtual routers, and virtual, real or emulated nodes
- RMAP support (ECSS-E-ST-50-11C)
- IP/Ethernet tunneling
- Virtual protocol analyzer

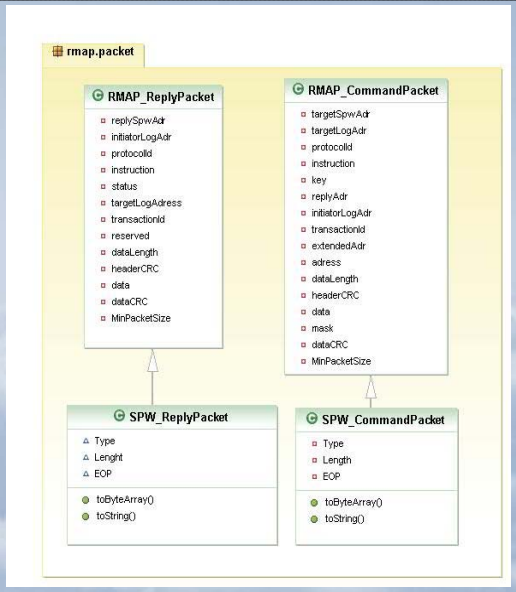
API Router

- 2 to 31 ports Wormhole routing,
- configurable via RMAP port 0
- path and logical adressing using routing table
- 3 arbitration modes:
    - packet priority, last served, lowest port number
- implementation inspired from SpW-10x router



API RMAP module

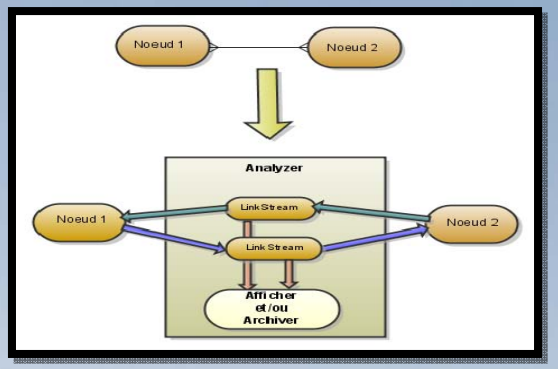




API: IP/Ethernet tunneling

- API interconnexion through IP/Ethernet
- local PC / local area / intranet or internet networking modes
- using Java RMI. Tested with VPN and direct client/server connexion.

API: Protocol analyzer



- between real or emulated nodes (PCI/PCI Express, smartCable), router ports and virtual nodes
- activable (on/off) for each direction of any link
- 4 configuration types: debug, blocking buffered, unblocking buffered, continuous
- 2 outputs: display or archive
- simple setMode() and getData() routines for easy mode config and data retrieves

API console

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                    Traffic Controller 4 SpW
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
=========Main Menu============
          (c) Config Network
          (r) Test RMAP
          (n) Test Network
          (x) Exit
Enter your choice : _
```

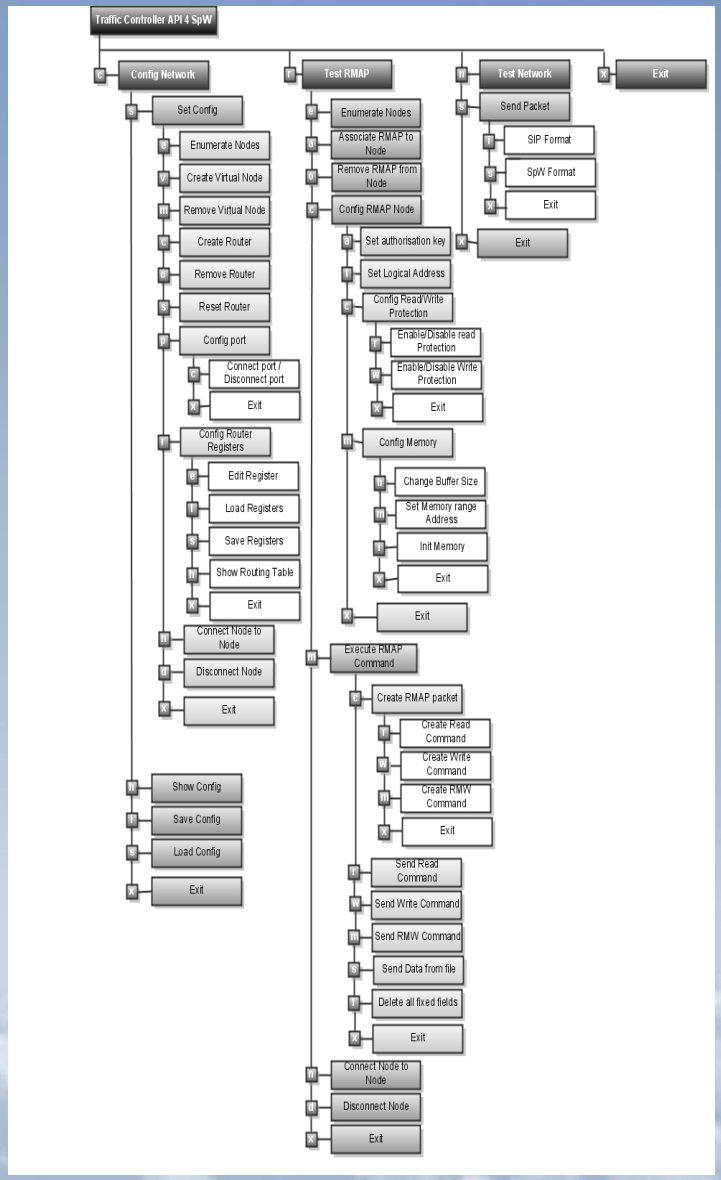
```
[Config Network]
          |__ (s) Set Config
              (h) Show Config
              (a) Save Config
              (l) Load Config
              (x) Exit
Enter your choice : _
```

```
[Config Network]
        |__[Set Config]
                |__(e) Enumerate Nodes
                   (v) Create Virtual Node
                   (m) Remove Virtual Node
                   (c) Create Router
                   (o) Remove Router
                   (s) Reset Router
                   (p) Config port
                   (r) Config Router Registers
                   (n) Connect Node to Node
                   (d) Disconnect node from node
                   (x) Exit
Enter your choice :
```

```
[Config Network]
       |__[Set Config]
              |__(e) Enumerate Nodes
                 (v) Create Virtual Node
                 (m) Remove Virtual Node
                 (c) Create Router
                 (o) Remove Router
                 (s) Reset Router
                 (p) Config port
                 (r) Config Router Registers
                 (n) Connect Node to Node
                 (d) Disconnect node from node
                 (x) Exit
Enter your choice : p
number of router in network [2]
        |__Router Id: [0x00]
        |__Router Id: [0x01]
Enter the Id of router
0
Router Id [0x00]
      |__number of port in this router: 5
          |__Port 1  [IdNode:0x0000] :  connected with virtual Node 0 [IdNode:0x0008]
          |__Port 2  [IdNode:0x0001] :  Not connected
          |__Port 3  [IdNode:0x0002] :  connected with virtual Node 1 [IdNode:0x0009]
          |__Port 4  [IdNode:0x0003] :  Not connected
          |__Port 5  [IdNode:0x0004] :  Not connected
        Enter port number
```

```
[Config Network]
       |__[Set Config]
              |__[Config Router Registers]
                        |__ (e)  Edit Register
                            (l)  Load Registers
                            (s)  Save Registers
                            (h)  Show Routing Table
                            (x)  Exit
Enter your choice : h
       |__Routing Table
         Address        |       Priority|       Delete Header   |       Ports

                32      |          0    |          0            |          2
                33      |          0    |          0            |          3
                34      |          0    |          0            |          4
                35      |          0    |          0            |          5
                36      |          0    |          0            |          6
                37      |          0    |          0            |          7
                38      |          0    |          0            |          8
                39      |          0    |          0            |          9
                40      |          0    |          0            |          1
                41      |          0    |          0            |          2
        Press ENTER to continue or X to Exit
```



Outline

1. Introduction
2. traffiController^{4SpW} DVS
3. traffiController^{4SpW} API
4. traffiController GUI
5. Conclusion and questions

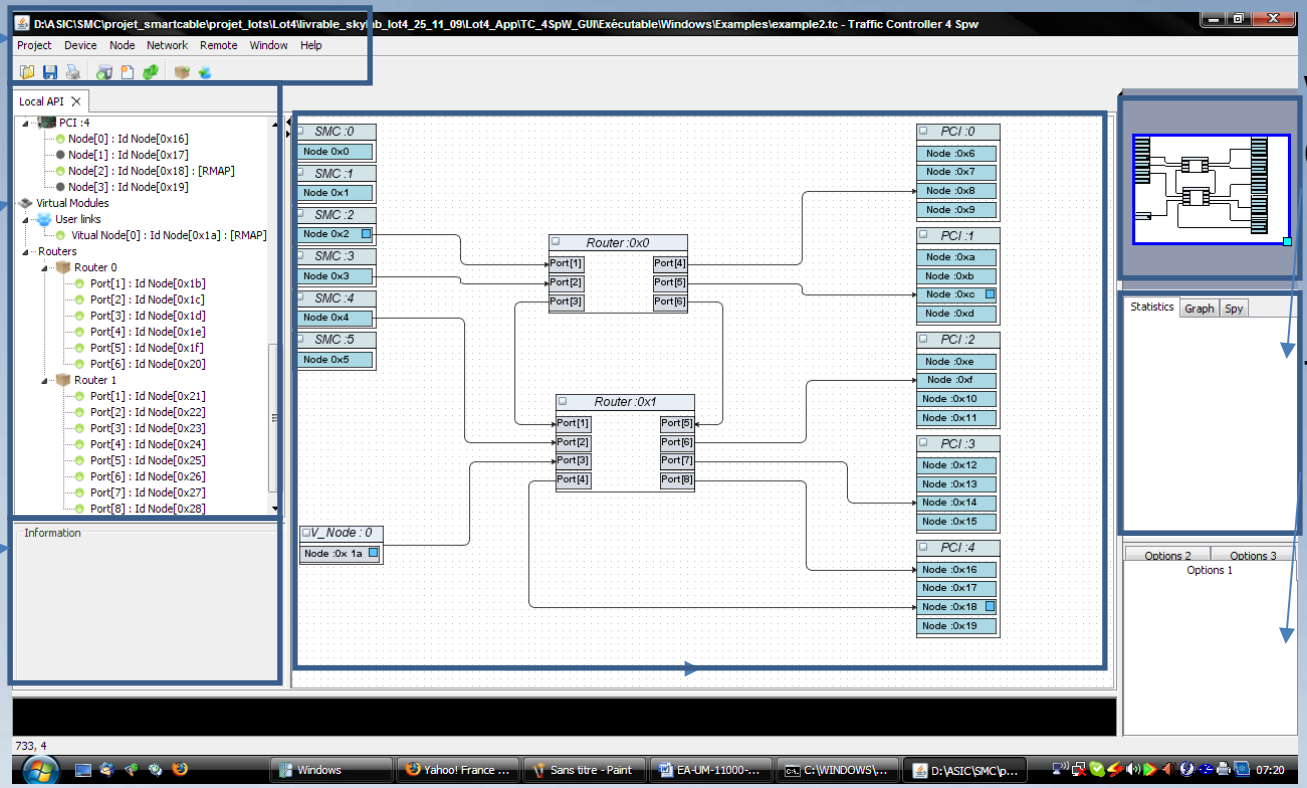
SKYLAB Industries

Graphical User Interface (overview)

Rich Menu Toolbar

Local API Hardware ressources

log

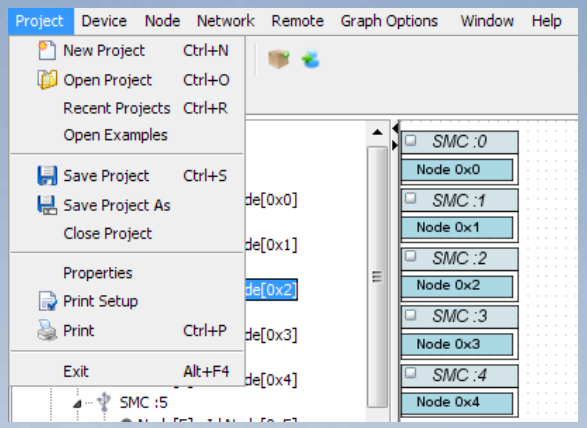


Work area dimensionning

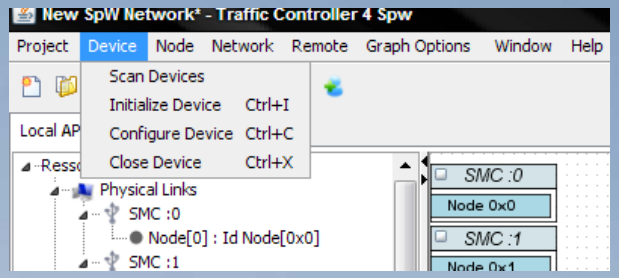
Traffic statistics
Eye diagrams

Main working area panel with intuitive functions acces
(right click or keyboard shortcuts)

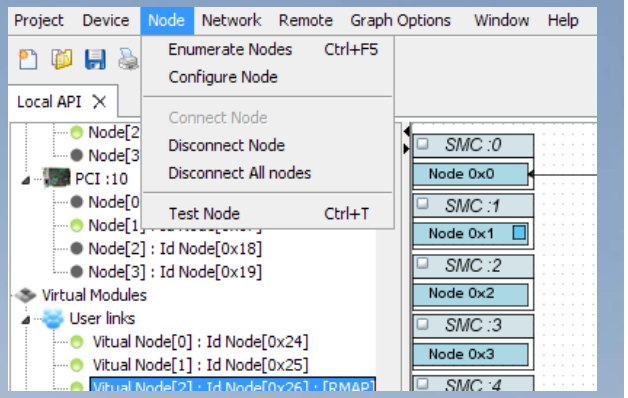
Graphical User Interface (basic functions)



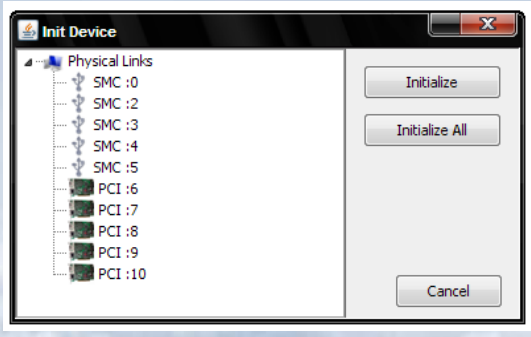
Project management



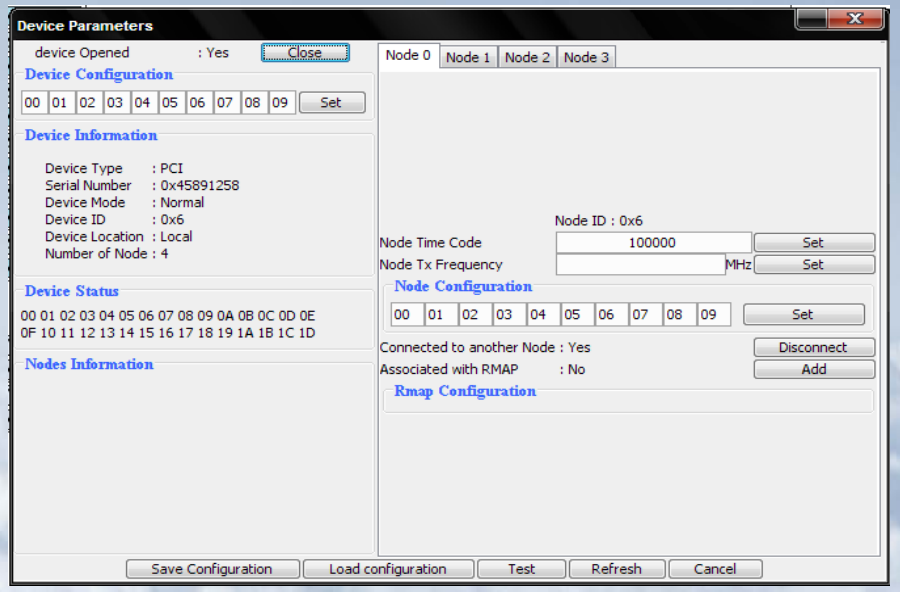
Device access



Node management

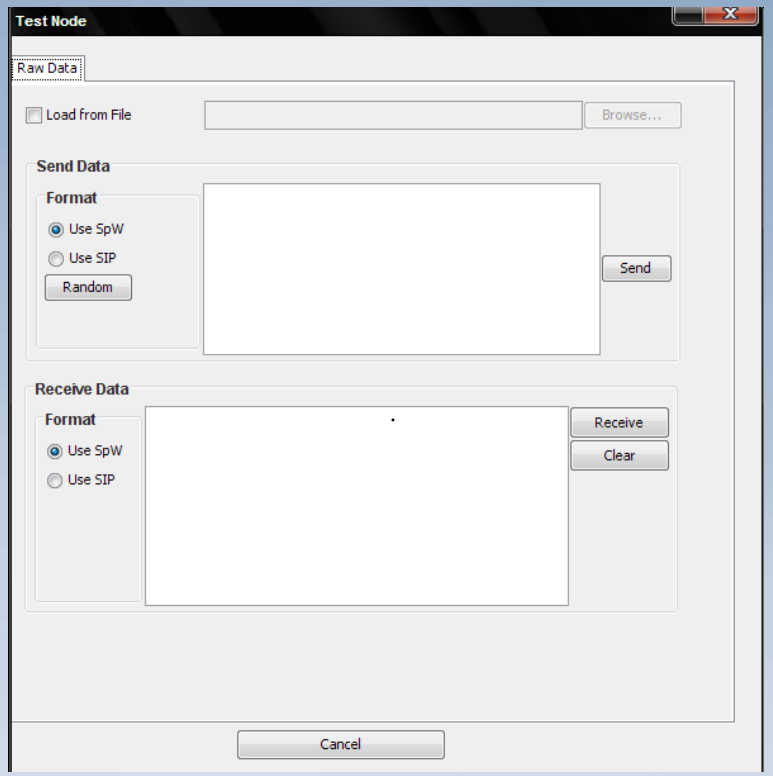


Peripheral initialization

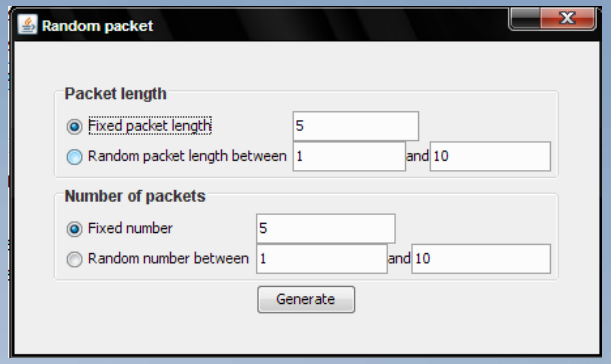


Peripheral configuration (ex: PCI4SpW)

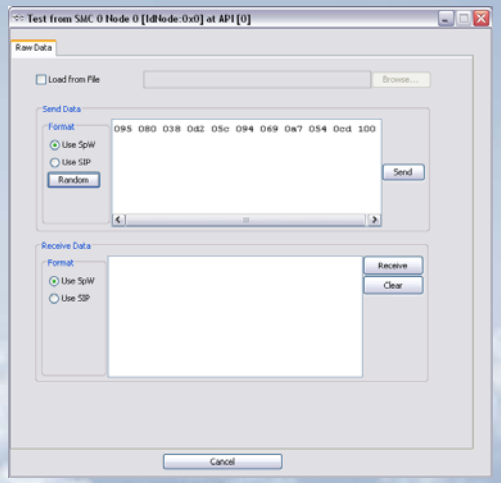
Graphical User Interface (SpW read/write functions)

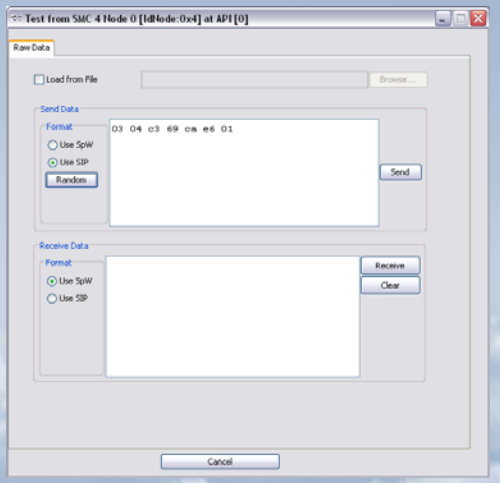


SpW or SIP* Send/Receive



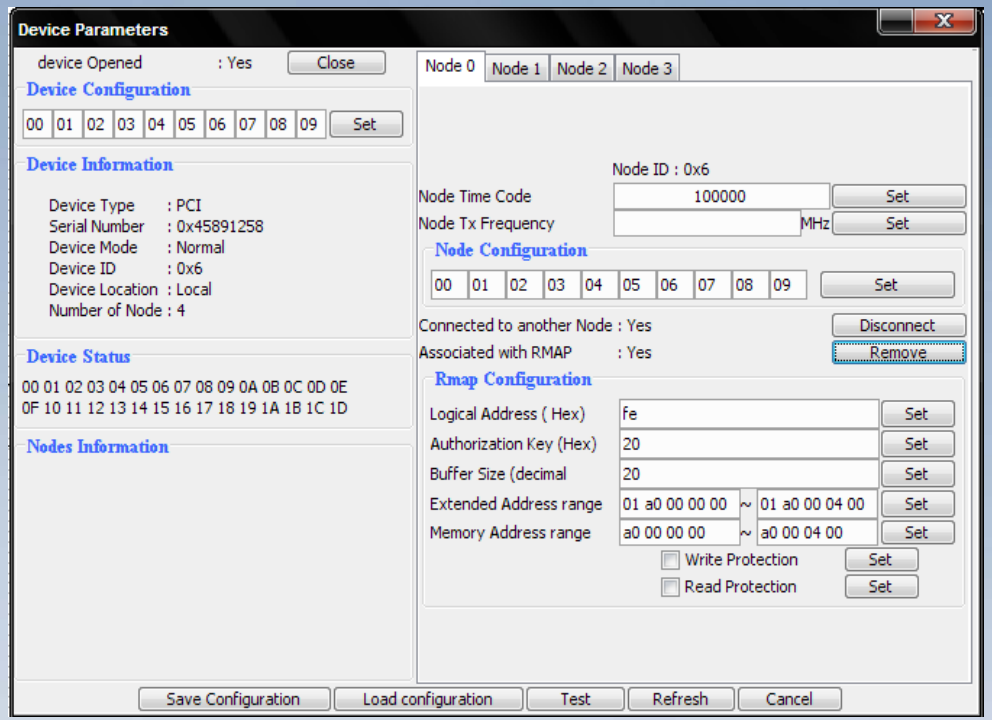
Packet generator

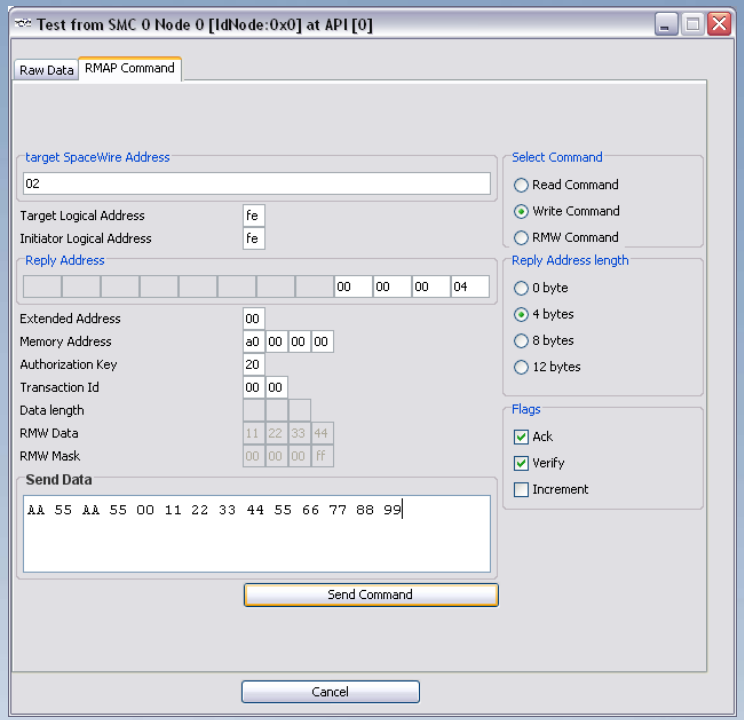




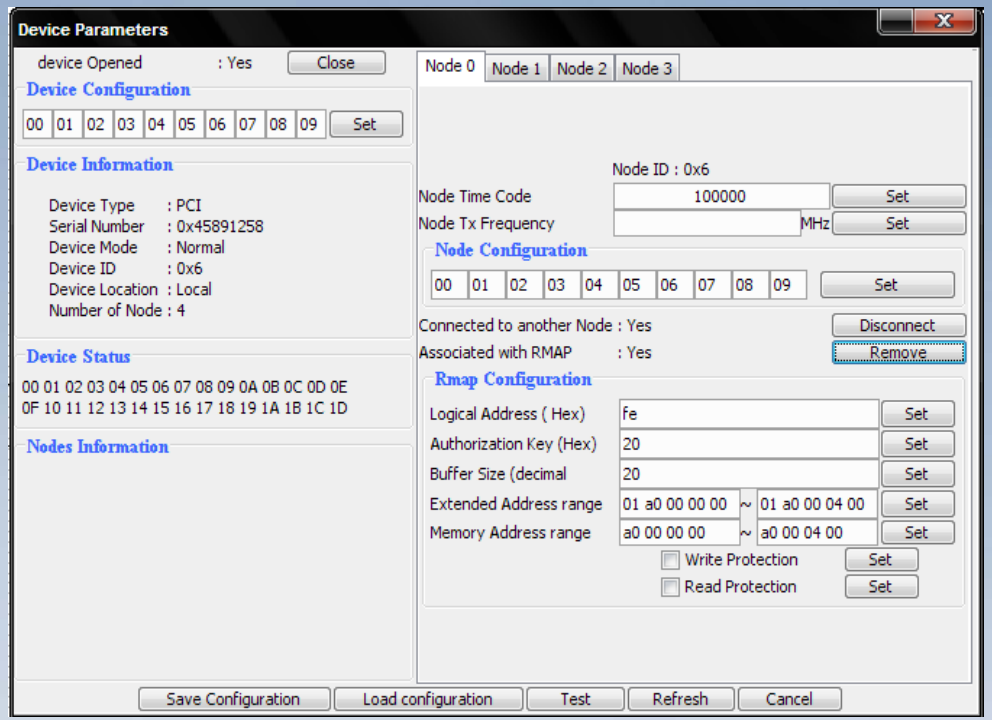
*SIP is SKYLAB **S**paceWire **I**nterpreted **P**rotocol, a 32bit ‘large cargos’ efficient processing format.

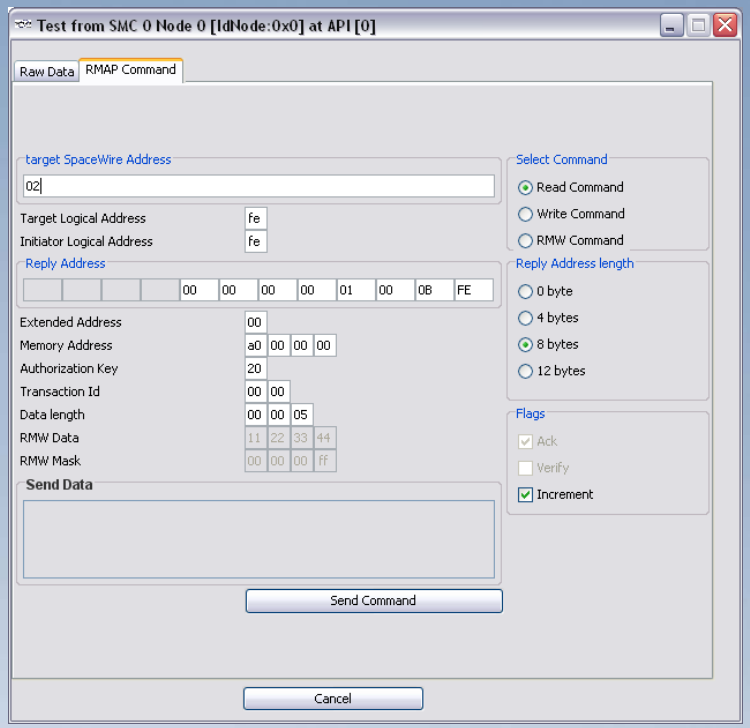
Graphical User Interface (RMAP Write function)



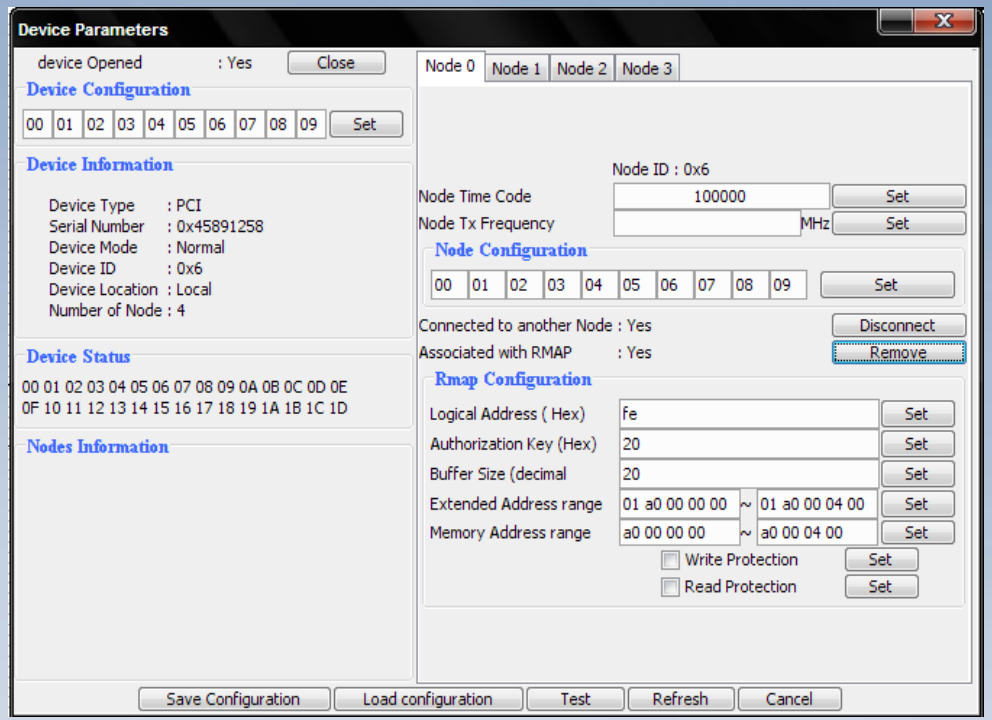


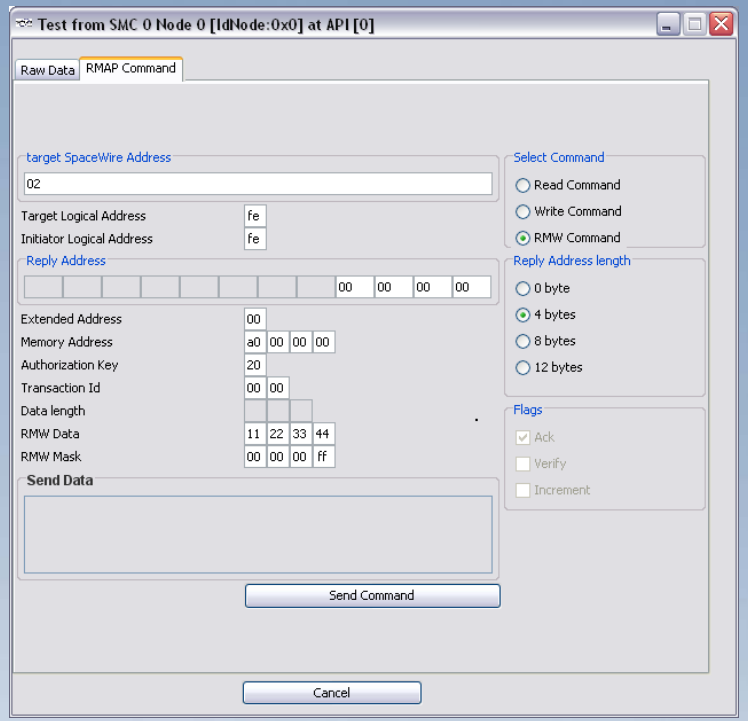
Graphical User Interface (RMAP Read function)



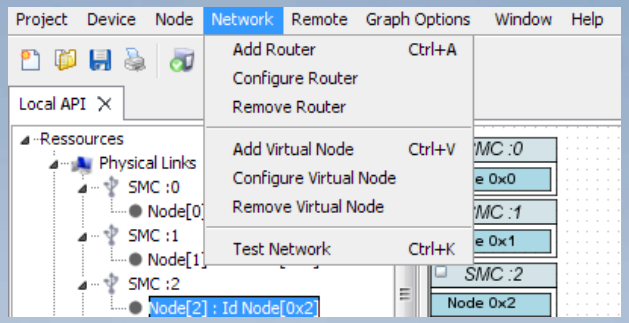


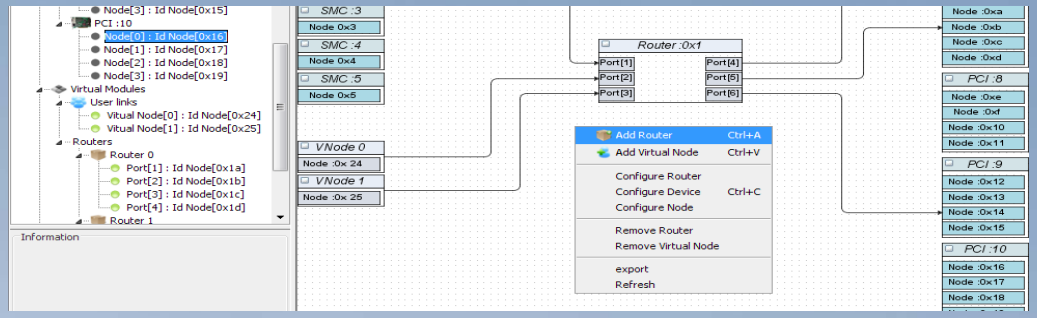
Graphical User Interface (RMAP RMW function)

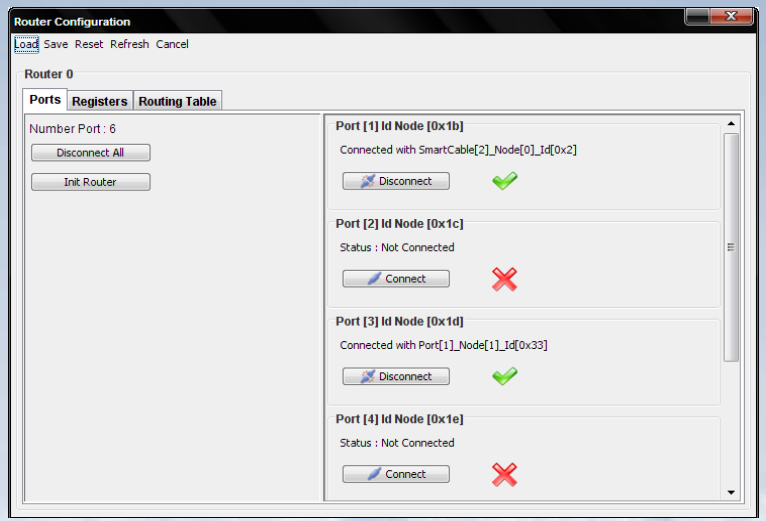


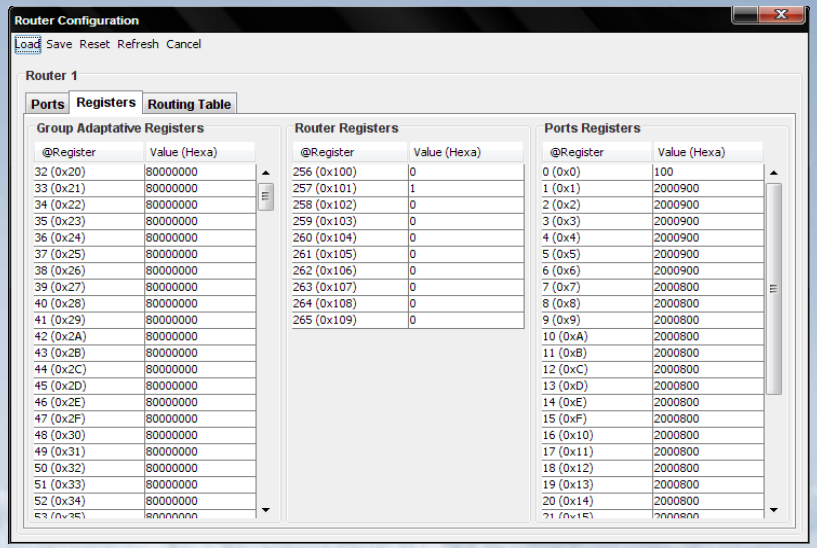


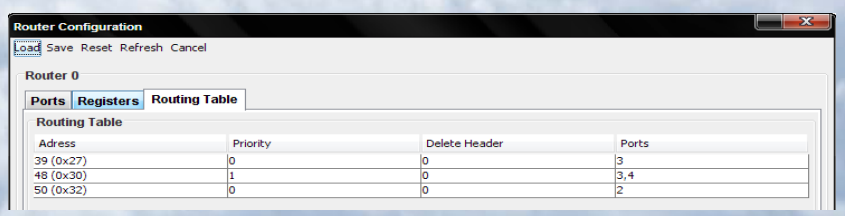
Graphical User Interface (Router functions)



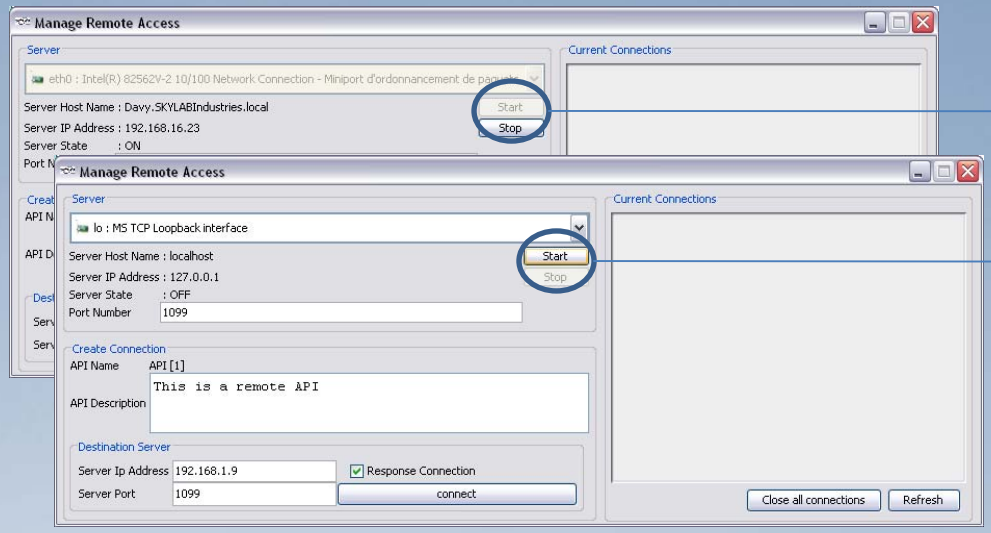






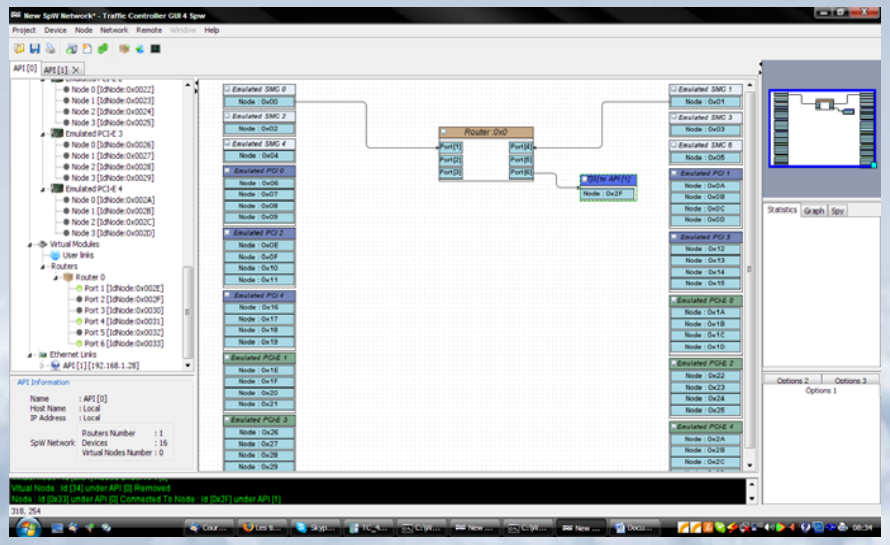


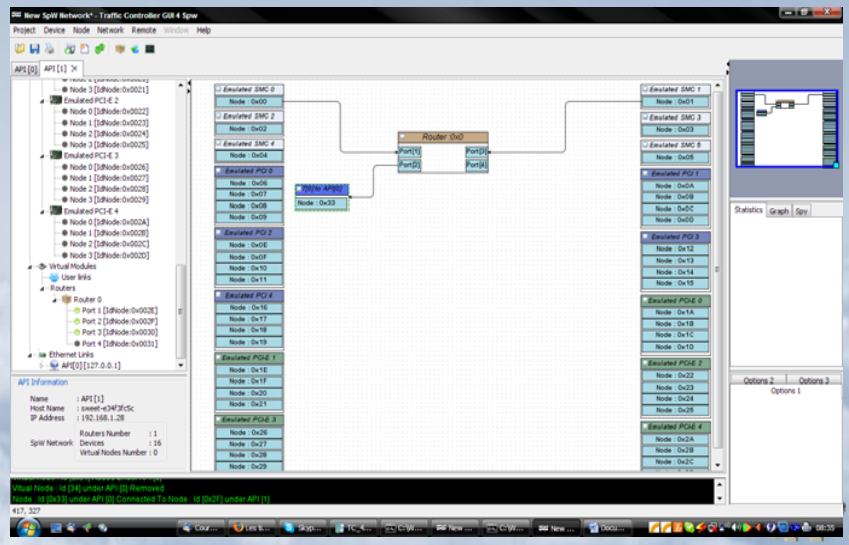
Graphical User Interface (IP tunnelling function)



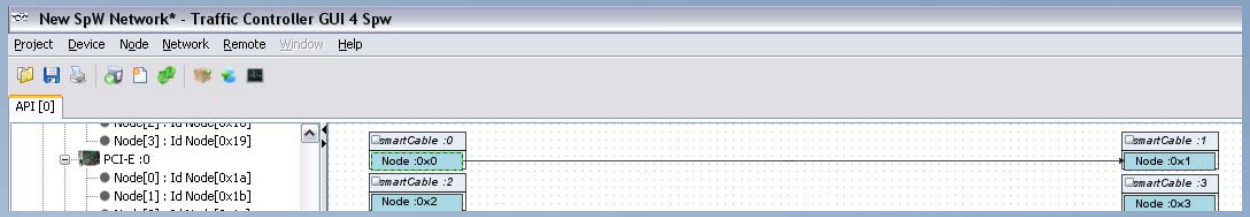
PC#1:API[0]

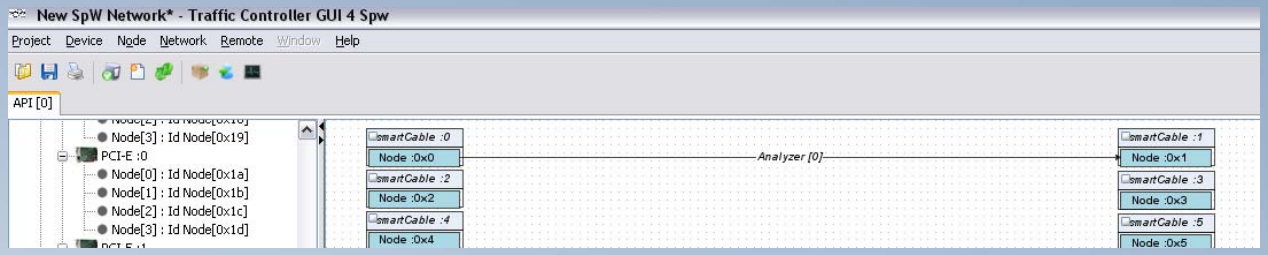
PC#2:API[1]

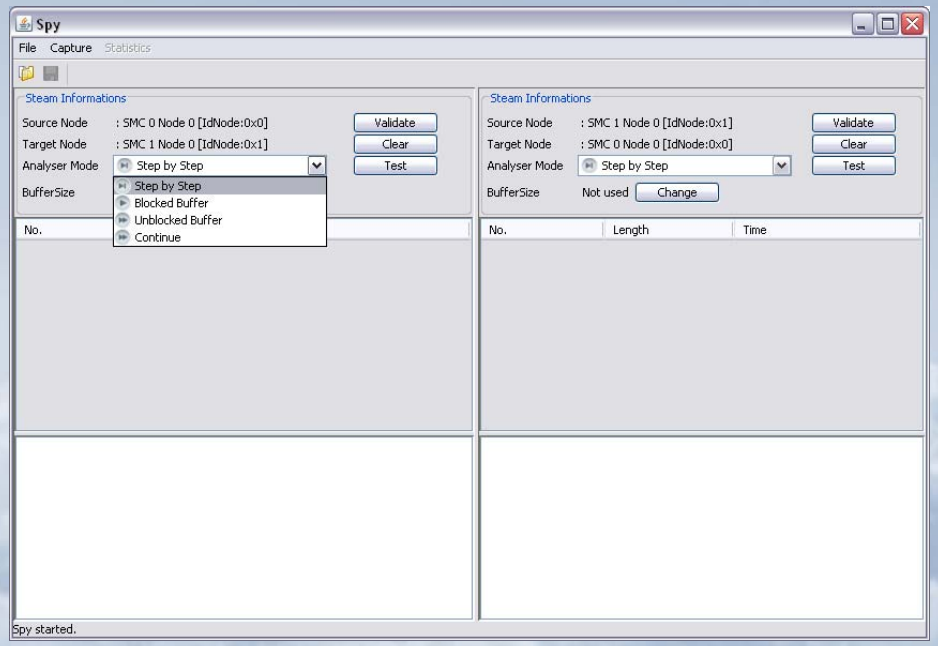




Graphical User Interface (virtual analyzer function)







Outline

1. Introduction
2. TraffiController^{4SpW} DVS
3. TraffiController^{4SpW} API
4. TraffiController^{4SpW} GUI
5. Conclusion and questions

SKYLAB Industries

traffiController^{4SpW}

- **DVS**
- **API**
- **GUI**

are the 3 levels of traffiController architecture, which implements extensive software features useful for network emulation, integration and test.

It is provided with user manual, javadoc, test use cases
TraffiController is included all hardware equipment commercialized by SKYLAB
It is also available in stand alone (without hardware, using emulated nodes)

There is a Skylab4SpW Googlegroup community for user feedback and support

SKYLAB plans to open the platform to competitive hardwares if demand strong enough.
We would like to hear about your potential additionnal requirements to be implemented.

Questions?

Thank you for your attention !

SKYLAB Industries SAS

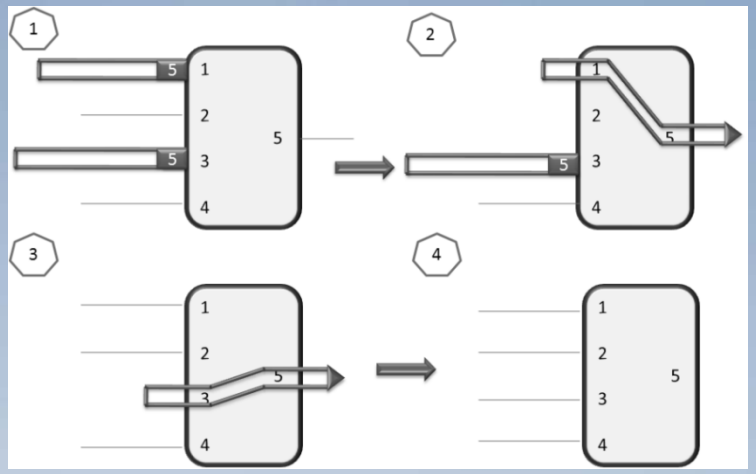
42, av. du general de Croutte
31100 Toulouse - France

Tel. +33 (0) 5.61.41.77.03
Fax +33 (0) 5.61.41.63.56

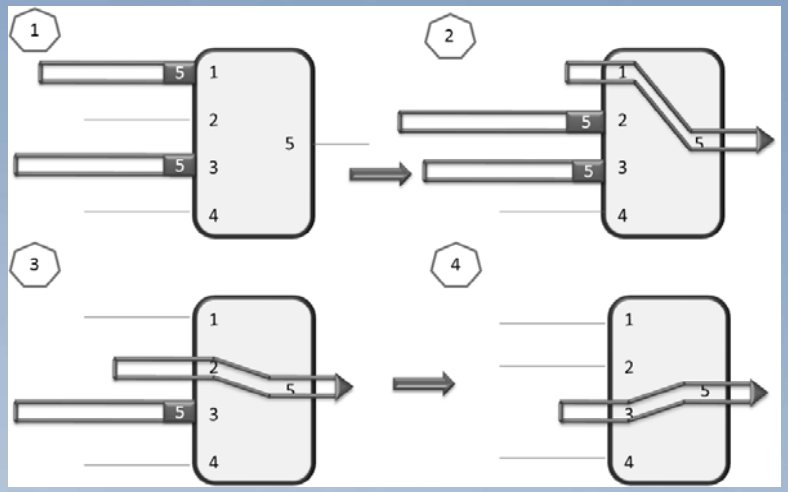
spacewire@skylab-corporate.com
www.skylab-corporate.com

SKYLAB
Industries

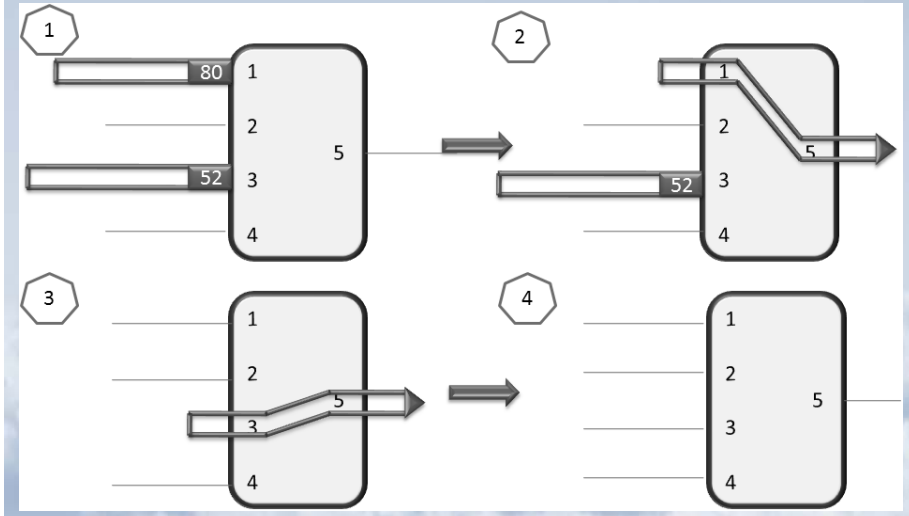
Spare Slide 1



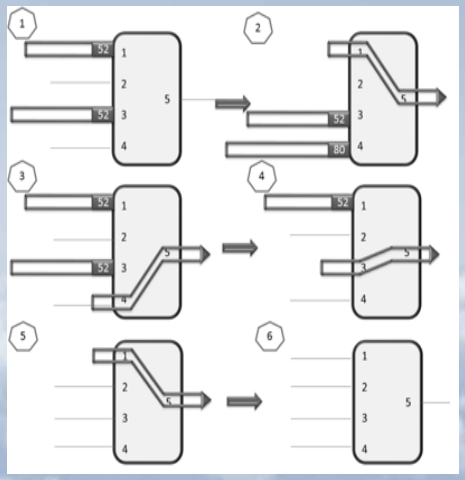
Same priority arbitration



Same priority arbitration



Different priorities arbitration



Different priorities arbitration